A Close Look at Decomposition-based Attributions for Transformer Language Models

Anonymous ACL submission

Abstract

Various attribution methods have been recently proposed for the transformer architecture, allowing for insights into the decision-making process of large language models by assigning importance scores to input tokens and latent representations. One class of methods that seems promising in this direction includes decomposition-based attributions that redistribute the model's output *logit* through the network, as this value is directly related to the prediction. In the previous literature we note though that two prominent methods of this category, namely ALTI-Logit and LRP, have not yet been compared to one another and propose to close this gap by conducting a careful quantitative evaluation w.r.t. ground truth annotations on the syntactic tasks of subject-verb agreement and reflexive anaphora, as well as on the semantic task of question answering, using the BERT, GPT-2 and Llama3 model families as a testbed. Along the way we compare and extend the ALTI-Logit and LRP methods from an algorithmic perspective, including the recently proposed AttnLRP variant for which we propose a more efficient implementation. We further incorporate in our benchmark two widely-used gradient-based and an attention-based baselines. Lastly, we make our carefully constructed benchmark datasets, as well as our code publicly available to foster evaluation on a well-defined common ground.[1]

1 Introduction

Language models (LMs) trained on massive corpora are able to learn various aspects of language (Brants et al., 2007; Tenney et al., 2019; Jawahar et al., 2019). Scaling them further on multiple axes (Kaplan et al., 2020; Snell et al., 2025) unlocks a range of interesting capabilities such as in-context learning (Brown et al., 2020) and chain-of-thought (Wei et al., 2022), allowing them to transition to what appears to be *general-purpose* models (Bubeck et al., 2023). At the same time, we have an incomplete understanding on how they are able to encode different concepts and abilities in their internal representations, which is especially crucial for high-stake scenarios (Sharkey et al., 2025).

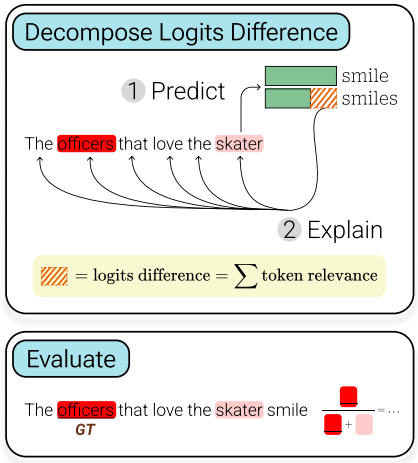

Figure 1: Illustration of our evaluation pipeline using a subject-verb agreement task: step 1) Predict the logit difference for the two verb forms, step 2) Explain the logit difference by generating token-level attributions, step 3) Evaluate attributions w.r.t. ground truth (i.e., the verb's subject) by computing various accuracy metrics.

One way to approach this question is by conducting circuit analysis, where certain behaviors can be localized to computational subgraphs (Wang et al., 2023; Hanna et al., 2023; Conmy et al., 2023). This, however, usually involves patching-based interventions that are costly. Recent works to alleviate this issue employed attribution methods (Denil et al., 2015; Shrikumar et al., 2016; Sundararajan et al., 2017) in a top-down manner, providing a significant speedup over standard interventions while maintaining localization performance (Ferrando and Voita, 2024; Hatefi et al., 2026). However, there are lingering questions regarding how com-

[1]Link will be made public upon paper acceptance.

mitting to a specific circuit discovery method can affect the quality of the token-level input attributions to understand individual predictions. In this work, we mainly focus on token-level attributions, i.e., attributions that assign a relevance score to each input token to indicate the token's importance for the current prediction. And for the purpose of comparing the quality of these attributions we restrict ourselves to controlled tasks where ground annotations can be built, so that our evaluation is based on well-defined syntactic and semantic linguistic evidence. More specifically, we analyze the decomposition-based attributions of LRP (Ali et al., 2022), AttnLRP (Achtibat et al., 2024) and ALTI-Logit (Ferrando et al., 2023) which were not yet compared in a controlled setup. To this end we construct careful ground truth annotations on two syntactic tasks (subject-verb agreement and reflexive anaphora) and one semantic task (extractive question answering). Moreover, we incorporate in our benchmarking two widely used gradient-based and an attention-based baselines. As a testbed we use the BERT and GPT-2 foundational model families, as well as the recent Llama3 family, covering both masked and autoregressive models, and study the impact of model size as well as various other configurations (such as, e.g., contrastive versus non-contrastive explanations) on the attributions' quality. Along the way we analyze and extend the decomposition-based methods of ALTI-Logit and LRP from an algorithmic perspective (e.g., via extending ALTI-Logit to the Llama3 family), and propose a more efficient implementation of AttnLRP. Lastly, we study the transferability of our input space attribution evaluation to the task of attention head pruning.

2 Related Work

Language Model Interpretability. Representations in LMs have been investigated with various approaches (Ferrando et al., 2024). BERTology (Rogers et al., 2020) mainly employ probes (Köhn, 2015; Alain and Bengio, 2016) to analyze *what* is encoded in the models' latent representations. Causal interventions (Vig et al., 2020; Geiger et al., 2021) take one step further by modifying internals to observe the resulting effect on outputs, since probes are correlational in nature (Belinkov, 2022). These enabled better understanding of internal information within LMs, from linguistic to world knowledge (Clark et al., 2019; Hewitt and Manning, 2019; Petroni et al., 2019, i.a.), along with components such as knowledge neurons (Dai et al., 2022) and induction heads (Olsson et al., 2022). They are, however, not without limitations.

Patching-based interventions are costly as model scales. Hence, they are approximated with attribution methods (Syed et al., 2024; Hanna et al., 2024; Jafari et al., 2025). Another issue is polysemanticity, therefore dictionary learning e.g., sparse autoencoders (SAEs) and their variants are employed to better disentangle representations (Bricken et al., 2023; Dunefsky et al., 2024). In parallel, there are efforts to make Multi-Layer Perceptron (MLP) neurons as interpretable as SAEs through attribution methods (Arora et al., 2026). In this work, we focus on such attribution methods, in particular those that *decompose* the output logit backward throughout the entire network.

Evaluating Attribution Methods. Quality of attributions can be evaluated with systematic input perturbation, which was initially proposed in computer vision and later extended to NLP (Samek et al., 2017; Arras et al., 2016; DeYoung et al., 2020; Edin et al., 2025), where the impact of the tokens' deletion (or insertion) according to their relevance is measured on the prediction. An issue with perturbation-based evaluation is that the model is fed with out-of-distribution perturbed inputs, leading to unreliable predictions. Other evaluations include user studies (Doshi-Velez and Kim, 2017; Lipton, 2018; Hase and Bansal, 2020), though such approaches face the dilemma than plausible explanations not necessarily reflect the model's decision strategy (Jacovi and Goldberg, 2020). In this work we use a restricted setup which allows for an evaluation w.r.t. ground truth annotations. Such ground-truth based approaches have already been used in past works, however to the best of our knowledge these works either did not release their annotations which limits reproducibility or had erroneous annotations (Ferrando et al., 2023; Poerner et al., 2018).

Explanations produced by different attributions have been found to sometimes contradict to one another (Krishna et al., 2024). To better understand this phenomenon, we break down our evaluation into various axes: tasks, model family and sizes, influence of special tokens, model precision, role of context vs. full question on question answering, and type of embeddings used for computing attributions, in order to distinguish common and specific patterns. Besides, we investigate the extension of

our findings at the token-level for head ablation.

3 Attribution Methods

In this section we briefly introduce the attribution methods we compare and extend in the present work. More details can be found in Appendix B.

3.1 ALTI-Logit and ALTI

ALTI-Logit is a decomposition-based attribution proposed by Ferrando et al. (2023). It uses the final layer unembedding matrix to assign to each MLP and Multi-Head Attention (MHA) block a "logit contribution" to the output logit, through an additive decomposition along the model's residual connections. Contributions of MLP blocks do not get redistributed further backward, while for MHA blocks they get backward redistributed onto input tokens using purposely designed redistribution rules. The overall redistribution is conservative and adds up to the output logit. ALTI-Logit was already applied to GPT-2 and we extend it to Llama3 by adapting it to handle grouped-query attention (Ainslie et al., 2023) and RMSNorm (Zhang and Sennrich, 2019). Since ALTI-Logit can not be extended to BERT (because residual connections are interleaved with LayerNorm (Ba et al., 2016) hindering the layer-wise additive decomposition of ALTI-Logit), for BERT we instead use ALTI (Ferrando et al., 2022) as the nearest proxy. ALTI is a purely attention-based attribution and is not related to the model's output logit.

3.2 LRP and AttnLRP

LRP for Transformers is a decomposition-based attribution proposed by Ali et al. (2022). It is based on decomposing the model's output logit layer-by-layer by designing propagation rules that assign a relevance to every neuron of the model. While ALTI-Logit decomposes the output logit additively among different layers, LRP assign to each layer a relevance that is equal to the output logit. While the LRP variant from Ali et al. (2022) treats attention weights as constants and do propagate relevance backward through these neurons, recently Achtibat et al. (2024) proposed another variant where the relevance of the product neuron between attention weights and the value vector get each assigned an equal share of relevance, hence propagating relevance backward through attention weights.

3.3 Baselines

Gradient-based. We consider two gradient-based attributions where the contributions of an input token (or a hidden vector) is computed either as the L_1-norm (Gradient$_{l_1}$) or squared L_2-norm (Gradient$_{l_2}$) of the output logit's gradient w.r.t. that token (or vector). Another gradient-based variant is Gradient$\times$Input, where the output logit's gradient w.r.t. the input token (or hidden vector) is element-wise multiplied by the forward pass value of that token (or vector). All variants can be efficiently implemented via standard gradient backpropagation.

Attention Rollout. Here, attribution is obtained by multiplying attention matrices recursively. This method is independent of models' output logits (Abnar and Zuidema, 2020).

3.4 LRPx: efficient and simple implementation of LRP variants

We introduce a *modified* Gradient$\times$Input strategy for implementing AttnLRP in simple and efficient way, which is inspired by previous works on LRP. The key idea is to detach parts of the computational graph during the forward pass, without modifying the forward pass values, but in a way that gradient backpropagation on the modified graph will yield gradients which can be multiplied by forward pass values to obtain AttnLRP. In Appendix C we provide for the first time proofs of this equivalence. We implement this strategy inside a new LRPx toolbox (where the "x" stands for "multiple" LRP variants) as it can implement both AttnLRP and LRP efficiently. In section 5.4 we benchmark the computational cost of the original AttnLRP implementation from Achtibat et al. (2024) against our AttnLRP implementation.

4 A Benchmark for LM Attributions

4.1 Tasks and ground-truths

For reflexive anaphora (`ra`) we use the BLiMP number agreement dataset from Warstadt et al. (2020). For subject-verb agreement (`sva`) we use the dataset released by Goldberg (2019), derived from Linzen et al. (2016). For question answering (`qa`) we use the answerable questions of the SQuAD-v2 dataset from Rajpurkar et al. (2018). On the syntactic tasks, we build the ground truth nominal subject's tokens using spaCy's dependency parser[2] for `sva` and fastcoref's coreference resolu-

[2] `https://spacy.io`

tion[3] for ra. On the semantic task of extractive qa we define as the ground truth the region in the question's context that is considered as the correct answer to a question. For all tasks, we generate a *generic* benchmarking dataset with ground truth annotations that is model independent, as well as a *tokenized* version for each model family we use. Our datasets have a size of 1k samples for ra, 30k samples for sva and 40k samples for qa.

4.2 Evaluation Metrics

Pointing Game (PG). This metric looks at the top token with the highest relevance. If this token is within the ground truth, the accuracy is 1 else 0. A similar metric was used in Poerner et al. (2018).

Mean Reciprocal Rank (MRR). This is the sole metric reported in ALTI-Logit (Ferrando et al., 2023). It consists in retrieving the inverse of the minimal rank (in decreasing order of relevance) of the tokens belonging to the ground truth.

Relevance Mass Accuracy (RMA). This metric was introduced in computer vision (Arras et al., 2022), and calculates the fraction of positive relevance that falls inside the ground truth w.r.t. the total positive relevance.

Per-Token Accuracy (PTA). This metric makes a binary classification decision using the sign of the relevance. It assumes tokens inside the ground truth shall receive a strictly positive relevance, while tokens outside the ground truth shall have zero or a negative relevance. As preliminary experiments revealed that the PTA metric is close to random baselines across all attribution methods we only report PTA results for Table 1.
All metrics are in the range [0, 1] (the higher the better), and we report results in percents.

4.3 Language Models

We employ the following models from the HuggingFace library: `bert-base-uncased`, `bert-large-uncased`, `gpt2`, `gpt2-medium`, `gpt2-large`, `gpt2-xl`, `Llama-3.2-1B`, `Llama-3.2-3B` and `Llama-3.1-8B`. Appendix Table 9 provides further details as well as the models' prediction accuracies on the considered tasks. Throughout our paper we use the terms BERT, GPT-2 and Llama3 when we refer to an entire model family. To avoid noise due to falsely predicted samples, we consider only correctly predicted samples for evaluating attributions w.r.t. our ground truth annotations.

[3] https://github.com/shon-otmazgin/fastcoref

4.4 Experimental Setup

For generating explanations we either use "contrastive explanations" from Yin and Neubig (2022), i.e., we explain a logit difference, or else we explain only the token with the highest logit for the considered task ("non-contrastive explanation"). For generating the token-level attributions for AttnLRP, LRP and Gradient×Input on BERT and GPT-2 we either use the "full embeddings" (which are the sum of token embeddings and positional embeddings) or "partial embeddings" (meaning we use the token embeddings only). Unless stated otherwise, we use double precision to generate attributions on the ra and sva tasks, except on `Llama-3.2-3B` and `Llama-3.1-8B`, otherwise we use the model's default precision.

5 Results and Discussions

5.1 Overview

In Table 1 we provide an overview of our results on all tasks and models, using as a default setup: contrastive explanations, full embeddings (for BERT and GPT-2), ignoring special tokens, and for qa using only the context for evaluation. First, we observe globally consistent results for a given task and model family across model sizes, indicating that our findings are robust to scale and to some extent to random initialization (since models of different sizes were initialized with different weights). This also shows that larger models are not necessarily harder to explain (at least for the sizes we considered). Another interesting observation is that tasks with longer contexts do not lead to inferior results, when considering the respective best performing attribution for each task and family. For instance, the PG accuracy is *above* 62% across all models on the qa task for AttnLRP (and *above* 55% for LRP) which is particularly remarkable given that the input length has a minimum (resp. mean) of at least 45 (resp. 193) tokens on this task, while on the ra task ALTI-Logit reaches a PG accuracy *above* 70% across all GPT-2 and Llama3 models, and on this latter task the min (resp. mean) input lengths are at least 2 (resp. 4). Regarding the *maximum* PG accuracy across models for the same attributions, this value is 94% on qa for AttnLRP and LRP (reached on `bert-base-uncased`) and 88% on ra for ALTI-Logit (reached on `gpt2-xl`),

	ALTI-Logit/ALTI				AttnLRP				LRP				Gradient				AttnRollout			
ra	PG↑	MRR↑	RMA↑	PTA↑	PG	MRR	RMA	PTA	PG	MRR	RMA	PTA	PG	MRR	RMA	PTA	PG	MRR	RMA	PTA
bert-base-uncased	2	41	20	23	35	63	**36**	**47**	**36**	62	35	44	33	**65**	**36**	23	0	25	18	23
bert-large-uncased	4	44	22	23	21	52	27	43	26	55	**29**	**44**	**42**	**69**	40	23	0	27	15	23
gpt2	**71**	**84**	**54**	**41**	15	49	22	35	41	64	36	38	32	61	37	36	52	75	52	36
gpt2-medium	**70**	**84**	**54**	**42**	13	48	21	37	23	55	30	39	28	58	34	36	52	76	52	36
gpt2-large	**84**	**91**	**60**	**43**	8	45	17	36	17	52	27	39	36	64	39	36	52	76	52	36
gpt2-xl	**88**	**93**	**58**	**42**	14	49	23	**42**	20	55	30	**42**	36	64	39	36	52	76	52	36
Llama-3.2-1B	**82**	**91**	54	38	71	84	**57**	**47**	15	50	33	36	27	55	32	35	51	70	36	35
Llama-3.2-3B	**76**	**87**	**52**	37	53	73	47	**41**	48	70	41	36	24	54	31	35	54	74	41	35
Llama-3.1-8B	**82**	**90**	**58**	**39**	33	62	38	**39**	34	62	37	36	28	55	32	35	61	79	0	35
sva																				
bert-base-uncased	31	55	12	5	53	71	**27**	**34**	46	65	22	**34**	**55**	**72**	26	5	0	12	4	5
bert-large-uncased	**37**	**59**	12	5	34	58	**23**	37	31	52	19	**43**	36	57	19	5	0	7	4	5
gpt2	**68**	**81**	34	33	39	62	31	40	55	71	**36**	**44**	30	55	25	15	17	45	17	15
gpt2-medium	**77**	**86**	**42**	33	35	61	30	**43**	55	71	36	**43**	33	56	26	15	17	43	17	15
gpt2-large	**82**	**88**	**46**	36	38	63	32	35	60	75	41	**39**	29	53	27	15	17	43	17	15
gpt2-xl	**77**	**85**	40	32	43	66	35	38	60	75	**41**	**39**	30	54	26	15	17	42	17	15
Llama-3.2-1B	48	67	32	**34**	**70**	**83**	**41**	33	21	54	23	17	27	51	22	15	36	56	19	15
Llama-3.2-3B	43	63	33	**30**	**66**	**79**	**39**	28	47	66	27	18	27	48	21	15	27	49	20	15
Llama-3.1-8B	39	58	31	**36**	**54**	**72**	**33**	24	46	66	26	18	26	46	20	15	28	49	3	15
qa																				
bert-base-uncased	61	74	13	1	**94**	**97**	**48**	**48**	**94**	**97**	44	28	59	73	32	1	0	3	1	1
bert-large-uncased	53	66	8	1	91	95	47	54	**93**	**96**	**52**	**57**	47	60	23	1	0	2	1	1
gpt2	72	83	16	2	85	91	50	**68**	**88**	**93**	**51**	67	60	71	33	1	0	5	0	1
gpt2-medium	84	90	25	5	84	90	**52**	**70**	**86**	**91**	48	62	61	70	35	1	0	2	0	1
gpt2-large	57	66	15	2	**68**	**78**	**42**	34	67	75	39	**40**	52	62	29	1	0	3	0	1
gpt2-xl	-	-	-	-	77	85	53	51	77	84	48	46	63	73	34	1	0	3	0	1
Llama-3.2-1B	28	8	8	14	62	**11**	**24**	**24**	**72**	9	18	11	26	6	13	1	1	2	1	1
Llama-3.2-3B	-	-	-	-	70	12	28	26	62	8	15	8	18	6	10	1	0	2	1	1
Llama-3.1-8B	-	-	-	-	68	9	22	13	55	9	13	14	6	5	5	1	0	2	1	1

Table 1: Overview of attribution performance on all tasks and models. using the evaluation metrics of Pointing Game (PG), Mean Reciprocal Rank (MRR), Relevance Mass Accuracy (RMA) and Per-Token Accuracy (PTA), the higher the better. Best performing attribution for each metric is highlighted in bold, second best is underlined. Missing ALTI-Logit results on the qa task are due to extremely high memory requirements of this attribution (which prevented us from generating attributions even on 80GB GPUs).

which concretely means that in more than 88% of the samples the token with maximum relevance is inside the ground truth, which demonstrates a high achievable explainability on both tasks. A surprising result though is that the *relative* performance of attribution methods depends on the specific task and model family. On the syntactic task of ra ALTI-Logit performs very well on GPT-2 and Llama3. On sva ALTI-Logit performs very well on GPT-2 while AttnLRP is superior on Llama3. While on the semantic task of qa, AttnLRP and LRP both perform very well across all models, including BERT. At the same time, on the syntactic tasks, BERT models results are mixed across attribution methods, and even the Gradient baseline is competitive in this configuration. Aside from that latter exception on BERT and syntactic tasks, our study confirms that all decomposition-based attributions perform significantly better than the gradient-based and AttnRollout baselines, as was found in previous works (Ferrando et al., 2023; Achtibat et al., 2024). As for the evaluation metrics, PG/MRR/RMA exhibit a strong agreement, especially on the best performing attribution for each task and model family. Our analysis further uncovers a failure mode of MRR: although this metric is in principle related to PG (since it looks at the minimum rank of the ground truth tokens, ordered by decreasing relevance), due to the particular handling of ties used by Ferrando et al. (2023), the MRR score will substantially degrade when several tokens get assigned zero relevance, which occurs more frequently when computing Llama3 attributions with bfloat16 on longer qa inputs, hence MRR drastically drops although this is not related to a considerably lower explainability (as can be seen from the PG performance which keeps relatively high for AttnLRP/LRP, though RMA scores also decreases to some extent). When we compare the gradient-based baselines (see Appendix), we find that Gradient×Input is far worse than Gradient$_{l_1}$ and Gradient$_{l_2}$ across metrics, while Gradient$_{l_1}$ and Gradient$_{l_2}$ are mainly equivalent, except on RMA where Gradient$_{l_2}$ is slightly better than Gradient$_{l_1}$, therefore we report under Gradient only Gradient$_{l_2}$.

5.2 Fine-grained analyses

Impact of contrastive explanations. In Table 2 we analyze the impact of using contrastive explanations vs. non-contrastive explanations. Yin and

	ALTI-Logit			AttnLRP			LRP			Gradient		
ra	PG↑	MRR↑	RMA↑	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA
bert-base-uncased	- -	- -	- -	**35** 7	**63** 42	**36** 17	**36** 7	**62** 41	**35** 16	**33** 13	**65** 54	**36** 24
bert-large-uncased	- -	- -	- -	**21** 5	**52** 37	**27** 12	**26** 11	**55** 43	**29** 17	**42** 27	**69** 62	**40** 32
gpt2	**71** 5	**84** 40	**54** 0	15 **56**	49 **76**	22 **40**	41 **72**	64 **84**	**36** 18	**32** 6	**61** 42	**37** 21
gpt2-medium	**70** 12	**84** 40	**54** 0	13 **55**	48 **74**	21 **55**	23 **68**	55 **82**	**30** 26	**28** 9	**58** 43	**34** 23
gpt2-large	**84** 83	**91** **91**	**60** 58	**8** **8**	**45** 43	**17** 14	**17** 16	**52** 50	**27** 21	**36** 22	**64** 55	**39** 30
gpt2-xl	**88** **88**	**93** **93**	**58** 57	**14** 7	**49** 42	**23** 12	**20** 11	**55** 45	**30** 17	**36** 21	**64** 55	**39** 32
Llama-3.2-1B	**82** **82**	**91** **91**	**54** 50	**71** 0	**84** 36	**57** 17	**15** 1	**50** 40	**33** 26	**27** 12	**55** 45	**32** 24
Llama-3.2-3B	**76** **76**	**87** **87**	**52** 49	**53** 1	**73** 37	**47** 18	**48** 16	**70** 50	**41** 33	**24** 13	**54** 46	**31** 25
Llama-3.1-8B	**82** 77	**90** 88	**58** 52	**33** 7	**62** 42	**38** 24	**34** 14	**62** 50	**37** 30	**28** 22	**55** 51	**32** 29
sva												
bert-base-uncased	- -	- -	- -	**53** 19	**71** 43	**27** 14	**46** 21	**65** 43	**22** 12	**55** 31	**72** 53	**26** 18
bert-large-uncased	- -	- -	- -	**34** 17	**58** 41	**23** 13	**31** 18	**52** 40	**19** 13	**36** 23	**57** 44	**19** 14
gpt2	**68** 2	**81** 18	**34** 1	**39** 17	**62** 32	**31** 14	**55** 13	**71** 30	**36** 9	**30** 20	**55** 46	**25** 20
gpt2-medium	**77** 3	**86** 21	**42** 1	**35** 17	**61** 33	**30** 16	**55** 14	**71** 33	**36** 10	**33** 14	**56** 39	**26** 17
gpt2-large	**82** 75	**88** 84	**46** 39	**38** 19	**63** 50	**32** 22	**60** 38	**75** 61	**41** 30	**29** 28	**53** 52	**27** 26
gpt2-xl	**77** 71	**85** 82	**40** 35	**43** 19	**66** 50	**35** 22	**60** 33	**75** 59	**41** 28	**30** 28	**54** 53	**26** **26**
Llama-3.2-1B	**48** **48**	67 **68**	**32** 30	**70** 3	**83** 41	**41** 16	**21** 7	**54** 44	**23** 19	**27** 23	**51** 47	**22** 20
Llama-3.2-3B	**43** 42	**63** 62	**33** 29	**66** 5	**79** 42	**39** 17	**47** 24	**66** 51	**27** 21	**27** 25	**48** 47	**21** 20
Llama-3.1-8B	**39** 37	58 **59**	**31** 29	**54** 6	**72** 42	**33** 17	**46** 23	**66** 52	**26** 21	**26** 24	**46** 44	**20** 19
qa												
bert-base-uncased	- -	- -	- -	**94** **94**	**97** **97**	48 **49**	**94** **94**	**97** **97**	44 **46**	59 **61**	73 **74**	32 **34**
bert-large-uncased	- -	- -	- -	91 **92**	**95** **95**	**47** **47**	**93** **93**	**96** **96**	52 **54**	47 **49**	60 **62**	23 **24**
gpt2	**72** 42	**83** 49	16 **31**	**85** 8	**91** 19	**50** 5	**88** 16	**93** 25	**51** 8	**60** 45	**71** 61	**33** 23
gpt2-medium	**84** 25	**90** 42	**25** 24	**84** 35	**90** 48	**52** 15	**86** 37	**91** 46	**48** 12	**61** 45	**70** 59	**35** 23
gpt2-large	**57** **57**	66 **67**	15 **16**	68 **70**	78 **79**	42 **43**	67 **68**	75 **77**	39 **41**	52 **55**	62 **65**	29 **33**
gpt2-xl	- -	- -	- -	77 **79**	85 **86**	53 **55**	77 **79**	84 **85**	48 **50**	63 **66**	73 **75**	34 **38**
Llama-3.2-1B	28 **29**	**8** **8**	8 **9**	**62** 59	**11** **11**	**24** **24**	**72** 71	**9** **9**	**18** **18**	26 **31**	**6** **6**	13 **16**
Llama-3.2-3B	- -	- -	- -	**70** 69	12 **13**	**28** **28**	**62** **62**	**8** **8**	**15** **15**	18 **20**	**6** **6**	10 **11**
Llama-3.1-8B	- -	- -	- -	68 **69**	**9** **9**	**22** **22**	55 **56**	**9** **9**	**13** **13**	**6** **6**	**5** **5**	**5** **5**

Table 2: Attribution performance with contrastive explanations (left) vs. non-contrastive explanations (right).

	ALTI-Logit/ALTI			AttnLRP			LRP			Gradient			AttnRollout		
ra	PG↑	MRR↑	RMA↑	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA
bert-base-uncased	**2** **2**	**41** **41**	**20** 18	**35** **35**	**63** 62	**36** 33	**36** 35	**62** 60	**35** 30	**33** **33**	**65** 64	**36** 32	**0** **0**	**25** 14	**18** 4
bert-large-uncased	**4** **4**	**44** **44**	**22** 20	**21** 20	**52** 51	**27** 26	**26** **26**	**55** 51	**29** 26	**42** **42**	**69** **69**	**40** 36	**0** **0**	**27** 15	**15** 4
Llama-3.2-1B	**82** **82**	**91** **91**	**54** 51	**71** 62	**84** 78	**57** 49	**15** **15**	**50** **50**	**33** 32	**27** 16	**55** 42	**32** 23	**51** 0	**70** 39	**36** 0
Llama-3.2-3B	**76** **76**	**87** **87**	**52** 51	**53** 44	**73** 67	**47** 40	**48** 47	**70** 69	**41** 38	**24** 5	**54** 36	**31** 20	**54** 0	**74** 41	**41** 0
Llama-3.1-8B	**82** 81	**90** **90**	**58** 54	**33** 24	**62** 52	**38** 30	**34** 1	**62** 37	**37** 19	**28** 1	**55** 34	**32** 18	**61** 0	**79** 43	**0** **0**
sva															
bert-base-uncased	**31** **31**	**55** **55**	**12** **12**	**53** **53**	**71** 70	**27** 26	**46** **46**	**65** 64	**22** 21	**55** **55**	**72** **72**	**26** 25	**0** **0**	**12** 8	**4** 2
bert-large-uncased	**37** **37**	**59** **59**	**12** 11	**34** **34**	**58** 57	**23** 22	**31** **31**	**52** **52**	**19** 18	**36** 35	**57** 55	**19** 18	**0** **0**	**7** 6	**4** 2
Llama-3.2-1B	**48** 41	**67** 62	**32** 29	**70** 68	**83** 81	**41** 39	**21** 20	**54** 53	**23** 22	**27** 8	**51** 35	**22** 15	**36** 0	**56** 32	**19** 0
Llama-3.2-3B	**43** 34	**63** 56	**33** 27	**66** 62	**79** 76	**39** 36	**47** 41	**66** 62	**27** 24	**27** 2	**48** 30	**21** 14	**27** 0	**49** 30	**20** 0
Llama-3.1-8B	**39** 17	**58** 44	**31** 20	**54** 50	**72** 68	**33** 30	**46** 2	**66** 39	**26** 15	**26** 0	**46** 28	**20** 11	**28** 0	**49** 29	**3** 0
qa															
bert-base-uncased	**61** **61**	**74** **74**	**13** **13**	**94** **94**	**97** **97**	**48** **48**	**94** **94**	**97** **97**	**44** 42	**59** **59**	**73** **73**	**32** 31	**0** **0**	**3** 2	**1** 0
bert-large-uncased	**53** **53**	**66** **66**	**8** **8**	**91** **91**	**95** **95**	**47** 45	**93** 92	**96** 95	**52** 50	**47** **47**	**60** **60**	**23** **23**	**0** **0**	**2** **2**	**1** 0
Llama-3.2-1B	**28** 26	**8** **8**	**8** **8**	**62** 60	**11** **11**	**24** **24**	**72** 71	**9** **9**	**18** **18**	**26** 10	**6** **6**	**13** 10	**1** 0	**2** **2**	**1** 0
Llama-3.2-3B	- -	- -	- -	**70** **70**	**12** **12**	**28** **28**	**62** **62**	**8** **8**	**15** **15**	**18** 2	**6** 5	**10** 6	**0** **0**	**2** **2**	**1** 0
Llama-3.1-8B	- -	- -	- -	**68** **68**	**9** **9**	**22** **22**	**55** 11	**9** 8	**13** 10	**6** 1	**5** 4	**5** 3	**0** **0**	**2** **2**	**1** 0

Table 3: Attribution performance on non-special tokens (left) vs. all tokens (right).

Neubig (2022) introduced contrastive explanations and showed their superiority in a GPT-2 and a GPT-Neo model, using gradient-based and erasure-based attributions and syntactic tasks. We study the impact of contrastive explanations for the first time on the decomposition-based explanations of ALTI-Logit, AttnLRP and LRP, as well as on the semantic task of qa, across model families and sizes. Aside from a few exceptions on the ra task (w.r.t. AttnLRP/LRP and the gpt2/gpt2-medium models), contrastive explanations are either strikingly superior or almost equivalent to non-contrastive explanations across all tasks, models and attributions. This picture is particularly clear on the sva task. Hence using contrastive explanations is a good default choice when generating attributions. On the qa task, a few GPT-2 models greatly benefit from using contrastive explanations, while for the remaining models (BERT and Llama3), both contrastive and non-contrastive explanations are mainly equivalent.

Impact of special tokens. A non-negligible portion of relevance is assigned to special tokens across all attributions methods. This is illustrated in Table 3 by the fact that evaluation using only the non-special tokens often leads to better performance than when using all tokens (or to equivalent performance). Therefore we recommend escaping the special tokens when using attributions to understand the model's predictions.

Impact of context vs. full question for qa. A similar finding holds for the task of qa, where the question itself gets assigned a non-negligible relevance across all attributions. Hence, performance increases uniformly when considering only the context for evaluation, see Table 4.

Impact of model precision. We study the impact of model precision on explanation quality in Table 5. Default precision is float32 for BERT/GPT-

	ALTI-Logit/ALTI			AttnLRP			LRP			Gradient			AttnRollout		
qa	PG↑	MRR↑	RMA↑	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA
bert-base-uncased	**61** 18	**74** 40	**13** 7	**94** 60	**97** 76	**48** 26	**94** 72	**97** 83	**44** 28	**59** 38	**73** 53	**32** 19	**0** **0**	**3** 2	**1** **1**
bert-large-uncased	**53** 24	**66** 42	**8** 5	**91** 48	**95** 68	**47** 25	**93** 64	**96** 78	**52** 31	**47** 32	**60** 46	**23** 16	**0** **0**	**2** **2**	**1** **1**
gpt2	**72** 69	**83** 80	**16** 14	**85** 32	**91** 60	**50** 23	**88** 70	**93** 81	**51** 36	**60** 46	**71** 59	**33** 23	**0** **0**	**5** **5**	**0** **0**
gpt2-medium	**84** 79	**90** 86	**25** 21	**84** 37	**90** 63	**52** 27	**86** 61	**91** 75	**48** 32	**61** 48	**70** 60	**35** 26	**0** **0**	**2** **2**	**0** **0**
gpt2-large	**57** 56	**66** 65	**15** 14	**68** 35	**78** 57	**42** 24	**67** 55	**75** 66	**39** 29	**52** 44	**62** 55	**29** 23	**0** **0**	**3** **3**	**0** **0**
gpt2-xl	- -	- -	- -	**77** 35	**85** 60	**53** 27	**77** 53	**84** 68	**48** 31	**63** 48	**73** 61	**34** 25	**0** **0**	**3** **3**	**0** **0**
Llama-3.2-1B	**28** 26	**8** 6	**8** 7	**62** 11	**11** 8	**24** 11	**72** 21	**9** 6	**18** 10	**26** 19	**6** 5	**13** 8	**1** **1**	**2** **2**	**1** **1**
Llama-3.2-3B	- -	- -	- -	**70** 13	**12** 9	**28** 12	**62** 34	**8** 6	**15** 9	**18** 14	**6** 4	**10** 6	**0** **0**	**2** 1	**1** **1**
Llama-3.1-8B	- -	- -	- -	**68** 9	**9** 7	**22** 10	**55** 39	**9** 7	**13** 8	**6** 5	**5** 3	**5** 3	**0** **0**	**2** 1	**1** **1**

Table 4: Attribution performance on context-only (left) vs. full question (right) on the qa task.

	ALTI-Logit/ALTI			AttnLRP			LRP			Gradient			AttnRollout		
ra	PG↑	MRR↑	RMA↑	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA	PG	MRR	RMA
bert-large-uncased	**4** **4**	**44** **44**	**22** **22**	**21** **21**	**52** **52**	**27** **27**	**26** **26**	**55** **55**	**29** **29**	**42** **42**	**69** **69**	**40** **40**	**0** **0**	**27** **27**	**15** **15**
gpt2-xl	**88** **88**	**93** **93**	**58** **58**	**14** **14**	**49** **49**	**23** **23**	**20** **20**	**55** **55**	**30** **30**	**36** **36**	**64** **64**	**39** **39**	**52** **52**	**76** **76**	**52** **52**
Llama-3.2-1B	81 **82**	90 **91**	**54** **54**	**72** 71	83 **84**	**58** 57	**15** **15**	**50** **50**	**33** **33**	**27** **27**	**55** **55**	**32** **32**	50 **51**	68 **70**	**36** **36**
sva															
bert-large-uncased	**37** **37**	**59** **59**	**12** **12**	**34** **34**	**58** **58**	**23** **23**	**31** **31**	**52** **52**	**19** **19**	**36** **36**	**57** **57**	**19** **19**	**0** **0**	**7** **7**	**4** **4**
gpt2-xl	**77** **77**	**85** **85**	**40** **40**	**43** **43**	**66** **66**	**35** **35**	**60** **60**	**75** **75**	**41** **41**	**30** **30**	**54** **54**	**26** **26**	**17** **17**	**42** **42**	**17** **17**
Llama-3.2-1B	**48** **48**	65 **67**	**32** **32**	**71** 70	81 **83**	**41** **41**	**21** **21**	53 **54**	**23** **23**	**27** **27**	49 **51**	**22** **22**	**36** **36**	54 **56**	**19** **19**
qa															
bert-large-uncased	**53** **53**	**66** **66**	**8** **8**	**91** **91**	**95** **95**	**47** **47**	**93** **93**	**96** **96**	**52** **52**	**47** **47**	**60** **60**	**23** **23**	**0** **0**	**2** **2**	**1** **1**
gpt2-xl	- -	- -	- -	**77** **77**	**85** **85**	**53** **53**	**77** **77**	**84** **84**	**48** **48**	**63** **63**	**73** **73**	**34** **34**	**0** **0**	**3** **3**	**0** **0**
Llama-3.2-1B	- -	- -	- -	62 **63**	11 **77**	**24** **24**	**72** **72**	9 **83**	**18** **18**	**26** **26**	6 **43**	**13** **13**	**1** **1**	2 **7**	**1** **1**

Table 5: Attribution performance with default precision (left) vs. double precision (right).

	AttnLRP			LRP		
ra	PG↑	MRR↑	RMA↑	PG	MRR	RMA
bert-base-uncased	**35** 32	**63** 60	**36** 34	**36** 35	**62** 61	**35** 33
bert-large-uncased	**21** **21**	**52** **52**	**27** **27**	**26** **26**	**55** **55**	**29** **29**
gpt2	15 **20**	49 **53**	22 **28**	41 **49**	64 **70**	36 **43**
gpt2-medium	13 **15**	48 **49**	21 **22**	**23** 22	**55** **55**	**30** **30**
gpt2-large	**8** 7	**45** **45**	17 **18**	**17** 16	**52** **52**	**27** **27**
gpt2-xl	**14** 13	**49** **49**	**23** 22	**20** 18	**55** 53	**30** 29
sva						
bert-base-uncased	**53** 51	**71** 70	27 **28**	**46** 45	**65** 64	**22** **22**
bert-large-uncased	34 **35**	**58** **58**	**23** **23**	31 **32**	**52** **52**	19 **20**
gpt2	39 **41**	62 **63**	31 **32**	**55** **55**	**71** 70	36 **37**
gpt2-medium	35 **38**	61 **63**	30 **32**	**55** 52	**71** 69	**36** **36**
gpt2-large	**38** 36	**63** 62	**32** 31	**60** 57	**75** 73	**41** 40
gpt2-xl	**43** 41	**66** 65	**35** **35**	**60** 58	**75** 74	**41** 40
qa						
bert-base-uncased	**94** **94**	**97** **97**	48 **50**	**94** **94**	**97** **97**	**44** **44**
bert-large-uncased	**91** **91**	**95** **95**	**47** **47**	**93** 92	**96** 95	52 **53**
gpt2	**85** 83	**91** 90	**50** 46	88 **89**	**93** **93**	**51** **51**
gpt2-medium	**84** 83	**90** **90**	**52** **52**	**86** **86**	**91** **91**	48 **50**
gpt2-large	**68** **68**	**78** **78**	**42** **42**	**67** 66	**75** **75**	39 **40**
gpt2-xl	**77** **77**	**85** 84	**53** **53**	**77** **77**	**84** **84**	48 **49**

Table 6: Attribution performance with full embeddings (left) vs. partial embeddings (right).

	AttnLRP			LRP		
ra	PG↑	MRR↑	RMA↑	PG	MRR	RMA
bert-base-uncased	**35** 32	**63** 60	**36** 34	**36** 35	**62** 61	**35** 33
bert-large-uncased	**21** **21**	**52** **52**	**27** **27**	**26** **26**	**55** **55**	**29** **29**
gpt2	15 **20**	49 **53**	22 **28**	41 **49**	64 **70**	36 **43**
gpt2-medium	13 **15**	48 **49**	21 **22**	**23** 22	**55** **55**	**30** **30**
gpt2-large	**8** 7	**45** **45**	17 **18**	**17** 16	**52** **52**	**27** **27**
gpt2-xl	**14** 13	**49** **49**	**23** 22	**20** 18	**55** 53	**30** 29
sva						
bert-base-uncased	**53** 51	**71** 70	27 **28**	**46** 45	**65** 64	**22** **22**
bert-large-uncased	34 **35**	**58** **58**	**23** **23**	31 **32**	**52** **52**	19 **20**
gpt2	39 **41**	62 **63**	31 **32**	**55** **55**	**71** 70	36 **37**
gpt2-medium	35 **38**	61 **63**	30 **32**	**55** 52	**71** 69	**36** **36**
gpt2-large	**38** 36	**63** 62	**32** 31	**60** 57	**75** 73	**41** 40
gpt2-xl	**43** 41	**66** 65	**35** **35**	**60** 58	**75** 74	**41** 40
qa						
bert-base-uncased	**94** **94**	**97** **97**	48 **50**	**94** **94**	**97** **97**	**44** **44**
bert-large-uncased	**91** **91**	**95** **95**	**47** **47**	**93** 92	**96** 95	52 **53**
gpt2	**85** 83	**91** 90	**50** 46	88 **89**	**93** **93**	**51** **51**
gpt2-medium	**84** 83	**90** **90**	**52** **52**	**86** **86**	**91** **91**	48 **50**
gpt2-large	**68** **68**	**78** **78**	**42** **42**	**67** 66	**75** **75**	39 **40**
gpt2-xl	**77** **77**	**85** 84	**53** **53**	**77** **77**	**84** **84**	48 **49**

Table 7: Attribution performance with full embeddings (left) vs. partial embeddings (right).

2, and `bfloat16` for Llama3, which we compare to double precision (i.e., `float64`). Most results are equivalent irrespectively of the precision. Thus it is adequate to use the model's default precision for generating explanations.

Impact of embeddings. According to Table 7, in a bare majority of cases, on BERT and GPT-2 using the full embeddings is slightly better than using partial embeddings for AttnLRP/LRP (or else both are equivalent), therefore it is justified using the full embeddings as a default.

5.3 Transferability to head pruning

So far we validated the accuracy of token-level attributions w.r.t. linguistic and semantic evidence. We now investigate whether our findings for explaining and understanding predictions in terms of input attributions translate to head pruning efficiency, when using the respective latent attributions obtained with the same attribution methods (we exclude AttnRollout and ALTI, since they do not provide head-level attributions). Head-level pruning has been widely employed to exemplify usefulness of XAI (Michel et al., 2019; Voita et al., 2019; Li et al., 2021), though often limited to a few attribution methods. We compare head pruning across six attribution methods and a random baseline. Additionally, we explore various poolings for aggregating the attention's output attributions into one relevance value per head (with/without ReLU preprocessing, see Appendix D for more details). To reduce the computational cost of the

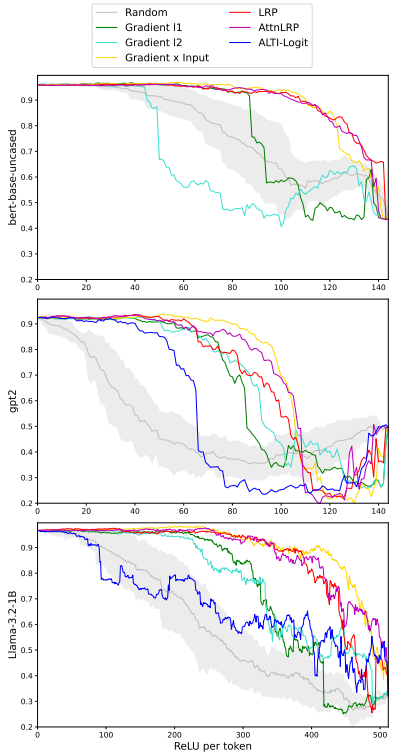

Figure 2: Pruning of attention heads (least relevant heads are pruned first) on the **sva task**. The y-axis depicts the prediction accuracy, the x-axis is the number of pruned heads, the higher the curve the better. Random is averaged over 10 runs.

experiments we select a random subset of 1k samples for the sva dataset and a random subset of 200 samples from the qa dataset. For ra and sva a token-level ReLU preprocessing seems to perform best across methods, while for qa results differ per pooling type. For sva the main results are provided in Figure 2 (for full results see Appendix D). Accordingly, the best performing attributions for head-level pruning are AttnLRP, LRP, and surprisingly the simple Gradient×Input while ALTI-Logit and other gradient-based variants are inferior. This underscores the fact that deleting parts of the model's internal components while retaining prediction performance, and quantifying which tokens are supporting the model's current prediction, are two different objectives that are not necessarily aligned. Hence, depending on the intended goal, different methods might perform well, and there does not seem to exist a general purpose XAI method that solves all problems.

5.4 Speedup with LRPx

We benchmark the runtime of our efficient AttnLRP implementation with our LRPx toolbox (i.e., using a *modified* Gradient×Input strategy) against the original AttnLRP implementation proposed in Achtibat et al. (2024) (which relies on an explicit layer-wise relevance computation). We also record standard Gradient×Input (i.e., without any LRP-specific layer modification) as a baseline. Runtimes are measured on the first 1000 samples of the sva dataset, using `bert-base-uncased`, `gpt2` and `Llama-3.2-1B` with single precision, on an NVIDIA Tesla V100 (32GB) and an A100 (40GB). We track only the backward pass and the token-level relevance aggregation, since the forward pass is identical across methods. We record median per-sample runtime, and median per-sample ratio between both AttnLRP implementations. Results are reported in Table 8. Across all models and GPUs, LRPx is consistently faster than the explicit AttnLRP implementation. The speedup ranges from 1.62× to 1.91× on the V100 and from 1.72× to 1.82× on the A100. This shows that our LRPx strategy substantially reduces the computational time of AttnLRP.

Model	GPU	G×I	AttnLRP	AttnLRP (ours)	Speedup
`bert-base-uncased`	V100	7.3	26.6	13.9	**1.91**×
`bert-base-uncased`	A100	7.5	24.2	13.3	**1.82**×
`gpt2`	V100	8.4	22.3	12.9	**1.73**×
`gpt2`	A100	8.6	21.5	12.1	**1.80**×
`Llama-3.2-1B`	V100	20.4	40.2	24.9	**1.62**×
`Llama-3.2-1B`	A100	20.5	39.7	23.1	**1.72**×

Table 8: Median per-sample runtimes (in ms) for Gradient×Input, AttnLRP (explicit implementation from Achtibat et al. (2024)), and our efficient AttnLRP implementation. Speedup is the median per-sample ratio between AttnLRP and our efficient AttnLRP.

6 Conclusion

In this work we compared the decomposition-based attributions of ALTI-Logit, LRP and AttnLRP w.r.t. carefully constructed ground truth annotations on various tasks, model families and sizes. On all considered configurations we could confirm that those attributions are superior to gradient-based and an attention-based baselines. Though the best performing attribution depends on the given task and model family, while model size does not seem to impact explanation quality. We further showed that results on token-level attribution quality do not directly translate into head pruning performance. Finally, we proposed a more efficient implementation of AttnLRP that relies on a *modified* Gradient×Input strategy, as well as theoretically demonstrated the validaty of this appraoch.

Limitations

We annotate ground truth labels automatically on the syntactic tasks using spaCy's dependency parser and fastcoref's coreference resolution, which might introduce some noise in the evaluation process, since these tools do not guarantee 100% correct labeling. Furthermore, parts of our benchmark datasets are extracted from real-world natural language data, and we do not filter samples for misspellings, typographical errors, and incorrect grammar. We believe these aspects should not influence the results in a noticeable way, and besides our goal is to compare attribution methods in a realistic setup.

Due to the nature of our selected tasks, we did not take interaction between tokens in pairs of inputs into account. Further investigations on this aspect could be done using second-order attributions to understand similarity between pairs of inputs (Moeller et al., 2023; Vasileiou and Eberle, 2024; Eberle et al., 2022; Dhamdhere et al., 2020).

Regarding the selection of models, we chose to focus on BERT, GPT-2 and Llama3 families since these are well-studied and commonly used in the NLP literature. Future work could explore further model variants.

References

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

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

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *Preprint*, arXiv:1610.01644.

Ameen Ali, Thomas Schnake, Oliver Eberle, Grégoire Montavon, Klaus-Robert Müller, and Lior Wolf. 2022. XAI for transformers: Better explanations through conservative propagation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 435–451. PMLR.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. 2019. Rudder: Return decomposition for delayed rewards.

Aryaman Arora, Zhengxuan Wu, Jacob Steinhardt, and Sarah Schwettmann. 2026. Language model circuits are sparse in the neuron basis. *Preprint*, arXiv:2601.22594.

Leila Arras, José Arjona-Medina, Michael Widrich, Grégoire Montavon, Michael Gillhofer, Klaus-Robert Müller, Sepp Hochreiter, and Wojciech Samek. 2019. Explaining and Interpreting LSTMs. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 211–238.

Leila Arras, Franziska Horn, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2016. Explaining predictions of non-linear classifiers in NLP. In *Proceedings of the 1st Workshop on Representation Learning for NLP*, pages 1–7, Berlin, Germany. Association for Computational Linguistics.

Leila Arras, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. 2017. Explaining recurrent neural network predictions in sentiment analysis. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

Leila Arras, Ahmed Osman, and Wojciech Samek. 2022. CLEVR-XAI: A benchmark dataset for the ground truth evaluation of neural network explanations. *Information Fusion*, 81:14–40.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *Preprint*, arXiv:1607.06450.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10(7):1–46.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages

858–867, Prague, Czech Republic. Association for Computational Linguistics.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *Preprint*, arXiv:2303.12712.

Hila Chefer, Shir Gur, and Lior Wolf. 2021a. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers . In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 387–396, Los Alamitos, CA, USA. IEEE Computer Society.

Hila Chefer, Shir Gur, and Lior Wolf. 2021b. Transformer Interpretability Beyond Attention Visualization . In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–791, Los Alamitos, CA, USA. IEEE Computer Society.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

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

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

Björn Deiseroth, Mayukh Deb, Samuel Weinbach, Manuel Brack, Patrick Schramowski, and Kristian Kersting. 2023. Atman: Understanding transformer predictions through memory efficient attention manipulation. In *Advances in Neural Information Processing Systems*, volume 36, pages 63437–63460. Curran Associates, Inc.

Misha Denil, Alban Demiraj, and Nando de Freitas. 2015. Extraction of Salient Sentences from Labelled Documents. *arXiv:1412.6815*. Version 2.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. 2020. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4443–4458, Online. Association for Computational Linguistics.

Kedar Dhamdhere, Ashish Agarwal, and Mukund Sundararajan. 2020. The shapley taylor interaction index. *Preprint*, arXiv:1902.05622.

Finale Doshi-Velez and Been Kim. 2017. Towards a rigorous science of interpretable machine learning. *Preprint*, arXiv:1702.08608.

Jacob Dunefsky, Philippe Chlenski, and Neel Nanda. 2024. Transcoders find interpretable llm feature circuits. In *Advances in Neural Information Processing Systems*, volume 37, pages 24375–24410. Curran Associates, Inc.

Oliver Eberle. 2022. *Explainable structured machine learning*. Phd thesis, Technische Universität Berlin. Https://doi.org/10.14279/depositonce-16149.

Oliver Eberle, Jochen Buttner, Florian Krautli, Klaus-Robert Muller, Matteo Valleriani, and Gregoire Montavon. 2022. Building and Interpreting Deep Similarity Models . *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 44(03):1149–1161.

Joakim Edin, Andreas Geert Motzfeldt, Casper L. Christensen, Tuukka Ruotsalo, Lars Maaløe, and Maria Maistro. 2025. Normalized AOPC: Fixing misleading faithfulness metrics for feature attributions explainability. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1730, Vienna, Austria. Association for Computational Linguistics.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, and 6 others. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

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

Javier Ferrando, Gerard I. Gállego, Ioannis Tsiamas, and Marta R. Costa-jussà. 2023. Explaining how transformers use context to build predictions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5486–5513, Toronto, Canada. Association for Computational Linguistics.

Javier Ferrando, Gabriele Sarti, Arianna Bisazza, and Marta R. Costa-jussà. 2024. A primer on the inner workings of transformer-based language models. *Preprint*, arXiv:2405.00208.

Javier Ferrando and Elena Voita. 2024. Information flow routes: Automatically interpreting language models at scale. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17432–17445, Miami, Florida, USA. Association for Computational Linguistics.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 9574–9586. Curran Associates, Inc.

Yoav Goldberg. 2019. Assessing BERT's Syntactic Abilities. *arXiv:1901.05287*.

Aaron Grattafiori and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does gpt-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Advances in Neural Information Processing Systems*, volume 36, pages 76033–76060. Curran Associates, Inc.

Michael Hanna, Sandro Pezzelle, and Yonatan Belinkov. 2024. Have faith in faithfulness: Going beyond circuit overlap when finding model mechanisms. In *First Conference on Language Modeling*.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, Online. Association for Computational Linguistics.

Sayed Mohammad Vakilzadeh Hatefi, Maximilian Dreyer, Reduan Achtibat, Patrick Kahardipraja, Thomas Wiegand, Wojciech Samek, Alexander Binder, and Sebastian Lapuschkin. 2026. Attribution-guided pruning for insight and control: Circuit discovery and targeted correction in small-scale llms. *Preprint*, arXiv:2506.13727.

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

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

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

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

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

Arne Köhn. 2015. What's in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Lisbon, Portugal. Association for Computational Linguistics.

Satyapriya Krishna, Tessa Han, Alex Gu, Steven Wu, Shahin Jabbari, and Himabindu Lakkaraju. 2024. The disagreement problem in explainable machine learning: A practitioner's perspective. *Transactions on Machine Learning Research*.

Sebastian Lapuschkin. 2019. *Opening the machine learning black box with Layer-wise Relevance Propagation*. Phd thesis, Technische Universität Berlin. Http://dx.doi.org/10.14279/depositonce-7942.

Jiaoda Li, Ryan Cotterell, and Mrinmaya Sachan. 2021. Differentiable subset pruning of transformer heads. *Transactions of the Association for Computational Linguistics*, 9:1442–1459.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in NLP. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691, San Diego, California. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Understanding neural networks through representation erasure. *Preprint*, arXiv:1612.08220.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Zachary C. Lipton. 2018. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

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

Lucas Moeller, Dmitry Nikolaev, and Sebastian Padó. 2023. An attribution method for Siamese encoders. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15818–15827, Singapore. Association for Computational Linguistics.

Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. 2019. Layer-wise relevance propagation: An overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, and 7 others. 2022. In-context learning and induction heads. *Preprint*, arXiv:2209.11895.

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

Nina Poerner, Hinrich Schütze, and Benjamin Roth. 2018. Evaluating neural network explanation methods using hybrid documents and morphosyntactic agreement. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 340–350, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Farnoush Rezaei Jafari, Grégoire Montavon, Klaus-Robert Müller, and Oliver Eberle. 2024. Mambalrp: Explaining selective state space sequence models. In *Advances in Neural Information Processing Systems*, volume 37, pages 118540–118570. Curran Associates, Inc.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

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

Lee Sharkey, Bilal Chughtai, Joshua Batson, Jack Lindsey, Jeffrey Wu, Lucius Bushnaq, Nicholas Goldowsky-Dill, Stefan Heimersheim, Alejandro Ortega, Joseph Isaac Bloom, Stella Biderman, Adrià Garriga-Alonso, Arthur Conmy, Neel Nanda, Jessica Mary Rumbelow, Martin Wattenberg, Nandi Schoots, Joseph Miller, William Saunders, and 10 others. 2025. Open problems in mechanistic interpretability. *Transactions on Machine Learning Research*. Survey Certification.

Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. 2016. Not just a black box: Interpretable deep learning by propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *International Conference on Learning Representations - Workshop track (ICLR)*.

Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. 2017. Smoothgrad: removing noise by adding noise. In *Proceedings of the International Conference on Machine Learning Workshop on Visualization for Deep Learning*.

Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2025. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*.

Ekta Sood, Simon Tannert, Diego Frassinelli, Andreas Bulling, and Ngoc Thang Vu. 2020. Interpreting attention models with human visual attention in machine reading comprehension. In *Proceedings of the 24th Conference on Computational Natural Language Learning*, pages 12–25, Online. Association for Computational Linguistics.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 3319–3328.

Aaquib Syed, Can Rager, and Arthur Conmy. 2024. Attribution patching outperforms automated circuit discovery. In *Proceedings of the 7th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*, pages 407–416, Miami, Florida, US. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Alexandros Vasileiou and Oliver Eberle. 2024. Explaining text similarity in transformer models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7859–7873, Mexico City, Mexico. Association for Computational Linguistics.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Advances in Neural Information Processing Systems*, volume 33, pages 12388–12401. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc.

Kayo Yin and Graham Neubig. 2022. Interpreting language models with contrastive explanations. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 184–198, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

A Implementation details

Software Requirements All experiments using attribution methods were conducted within a Python 3.11.9 environment using the following Python packages and corresponding versions:

- SpaCy 3.7.3
- Pandas 2.2.1
- PyTorch 2.3.0
- NumPy 1.26.4
- HuggingFace Transformers 4.48.1

The attention-head pruning experiments were conducted within a Python 3.12 environment using the following Python packages and corresponding versions:

- TransformerLens 2.18.0
- PyTorch 2.11.0
- NumPy 1.26.4
- HuggingFace Transformers 4.57.6

Model	# params	layers	heads	hidden size	sva	ra	qa
BERT (WordPiece, vocab size: 30,522)							
bert-base-uncased	110M	12	12	768	0.97	0.94	0.76
bert-large-uncased	340M	24	16	1024	0.97	0.95	0.83
GPT2 (BPE, vocab size: 50,257)							
gpt2	124M	12	12	768	0.92	0.99	0.66
gpt2-medium	345M	24	16	1024	0.93	0.99	0.69
gpt2-large	774M	36	20	1280	0.94	0.99	0.69
gpt2-xl	1.5B	48	25	1600	0.94	0.99	0.70
Llama3 (tiktoken BPE, vocab size: 128,256)							
Llama-3.2-1B	1.23B	16	32	2048	0.95	1.0	0.76
Llama-3.2-3B	3.21B	28	24	3072	0.96	1.0	0.78
Llama-3.1-8B	8B	32	32	4096	0.96	0.99	0.80

Table 9: Model Details and Prediction Performances on the sva, ra and qa tasks.

Model Details and Licenses Table 9 summarizes information about the models sizes and tokenizers as well as the prediction performance of the sva, ra and qa tasks. BERT is released under Apache 2.0, GPT-2 under MIT, and Llama3 under Meta Llama 3 Community License.

Computational Cost For the performed experiments, we required roughly 200 GPU hours on NVIDIA A100 40GB GPUs and 100 GPU hours on NVIDIA A100 80GB GPUs.

B Attribution methods details

Let us first introduce some notations that will help us analyze and compare the strategy of the considered attributions. Let x_t^l be the token representation for timestep t and layer l, and R_t^l the corresponding relevance[4] for this token. Accordingly R_t^0 represents the relevance of the input token for timestep t. Let $\mathbf{U}$ be the output unembedding matrix, and U_w the column vector for the predicted token w. Hence, the language model's prediction logit for predicting token w at timestep T is[5]: $\text{logit}_w = x_T^L \cdot U_w$, with L being the number of layers of the model. A property which is common to all decomposition-based attributions is that the logit_w is decomposed additively into contributions of model components (token, neuron, head or layer depending on the method), or in other words, the contributions of model components sum up to the value logit_w.

B.1 ALTI-Logit

ALTI-Logit is a recently proposed state-of-the-art decomposition-based approach for Transformer Language Models proposed by Ferrando et al. (2023). Its main idea is to additively decompose the final layer's token representation x_T^L used to compute the prediction logit (i.e., the penultimate vector ahead of the output unembedding layer $\mathbf{U}$) into layer-wise contributions of the outputs of each MLP and MHA block[6], by following the residual connections of the model (Elhage et al., 2021). While the contributions of the MLP blocks are not decomposed further backward, the contributions of the MHA blocks get further broken down into contributions of their respective input token representations, similarly to attention decomposition from Kobayashi et al. (2021). The latter is achieved by linearizing the MHA-block by viewing the attention weight matrix as a constant, as well as treating the standard deviation within the normalization layer as a constant, similarly to how Layer-wise Relevance Propagation (LRP) was previously extended to Transformers (Ali et al., 2022). Lastly, in order to account for the mixing of information across multiple layers, a token-level contribution matrix is built within each MHA block by considering the contributions of the MHA's transformed vectors to the MHA's output vector (as was done in the ALTI method by Ferrando et al. (2022)), and the resulting matrices are multiplied across layers to finally obtain an ALTI-Logit contribution for each input token. Overall the decomposition property of ALTI-Logit can be summarized as follows[7]: $\sum_t R_t^0 + \sum_l \tilde{R}_T^l = \text{logit}_w$, where R_t^0 is the input token contribution for each timestep t in the input sequence resulting from the MHA blocks and aggregated over all layers, while $\tilde{R}_T^l$ is the

[4] In this work we use the terms relevance, contribution, attribution and importance score interchangeably.

[5] Here we assume an autoregressive Language Model, with T being the input length, but most considered attributions are in principle applicable to masked Language Models as well.

[6] We refer to MLP as Multi-Layer Perceptron, and MHA as Multi-Head Attention, representing the two main components of the Transformer architecture.

[7] Ignoring contributions from model biases for simplicity of notation.

contribution of the output of each MLP block for the given prediction timestep T and layer l, since ALTI-Logit assumes that there is no mixing of information across timesteps resulting from MLP blocks.

In practice, the official implementation of ALTI-Logit[8] by Ferrando et al. (2023) requires the computation of a second, carefully designed forward pass through the model (using attention matrices, as well as weight parameters from various intermediate layers), after having run a first standard forward pass through the model during which the inputs and outputs of hidden layers are collected via hooks. This dedicated forward pass in ALTI-Logit was so far derived for Pre-LayerNorm architectures[9] and autoregressive Language Models, and exclusively applied to the models GPT2, OPT and BLOOM.

In this work we extend the ALTI-Logit algorithm to the Llama-3 model family (Grattafiori et al., 2024) by adapting ALTI-Logit to handle grouped-query attention (Ainslie et al., 2023), as well as RMSNorm normalization (Zhang and Sennrich, 2019). However, we refrain from adapting ALTI-Logit to the BERT model family, as this model is not compatible with an additive decomposition along residual connections due to an interleave with LayerNorm layers. Instead for BERT we use the ALTI method (Ferrando et al., 2022) as a proxy.

ALTI-Logit provides layer-wise token-level (as well as head-level) contributions to the prediction logit, and this method (resp. its components Logit (Ferrando et al., 2023) and ALTI (Ferrando et al., 2022)) were previously evaluated against Erasure (Li et al., 2017), Gradient (Simonyan et al., 2014; Li et al., 2016), Gradient×Input (Denil et al., 2015; Shrikumar et al., 2016), Integrated Gradients (Sundararajan et al., 2017), Attention Rollout (Abnar and Zuidema, 2020) and GlobEnc (Modarressi et al., 2022) explanations, where ALTI-Logit was shown to deliver the best results.

B.2 Layer-wise Relevance Propagation

Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) is an interpretability method based on a backward decomposition following a layer-wise conservation principle. In other words, in each layer of the model the contributions of neurons sum up to the prediction logit. More precisely it holds[10]: $\sum_t R_t^0 = \sum_t R_t^1 = \cdots = \sum_t R_t^L = \text{logit}_w$. LRP was initially proposed for Convolutional Neural Networks (Bach et al., 2015), and later extended to other models such as Recurrent Networks (Arras et al., 2017, 2019), Transformers (Ali et al., 2022; Achtibat et al., 2024) and selective State Space Models (Rezaei Jafari et al., 2024).

In practice, LRP can be implemented by applying dedicated LRP backward propagation rules for each type of layer occuring in the network, and that redistribute neuron relevances from upper layers to lower layers in a conservative manner (Montavon et al., 2019).

For a linear layer with forward pass equation $z_j = \sum_i z_i w_{ij} + b_j$, and given the relevances of the output neurons R_j, the input neurons' relevances R_i are computed through a summation of the form[11]: $R_i = \sum_j \frac{z_i \cdot w_{ij}}{z_j + \epsilon \cdot \text{sign}(z_j)} \cdot R_j$, hence their relevances are proportional to their forward pass contributions. For an element-wise activation layer of the form $z_j = g(z_i)$, with g being a non-linear activation function, the relevance R_j is redistributed backward using the identity rule, thus $R_i = R_j$. In order to extend LRP to Transformer models, it is required to design new rules to propagate the relevance backward through two further non-linearities typical to the models' architecture: product layers (occurring for instance in the product between attention weights and value vectors inside the MHA), and normalization layers (LayerNorm or RMSNorm). To this end Ali et al. (2022) propose to view the attention weights as a constant, which is equivalent to using the signal-take-all LRP redistribution rule for products which was previously proposed for extending LRP to Recurrent Neural Networks (Arras et al., 2017, 2019). For the normalization layers, Ali et al. (2022) propose to treat the standard deviation as a constant. In practice, these two rules can be implemented by treating the previous non-linearities as linear layers for LRP.

Ali et al. (2022) evaluated LRP against various attention-based XAI-methods (Abnar and Zuidema, 2020; Sood et al., 2020; Chefer et al., 2021a), as well as Gradient×Input (Denil et al., 2015; Shriku-

[8] https://github.com/mt-upc/logit-explanations

[9] We refer to *Pre*-LayerNorm to indicate that the normalization layer is located *before* the self-attention computation (resp. the fully-connected layers) within the MHA (resp. MLP) blocks, as opposed to *Post*-LayerNorm where the normalization happens *after* them.

[10] Here also ignoring the relevances assigned to model biases for simplicity.

[11] This rule corresponds to the LRP-ϵ rule (with ϵ being a small numerical stabilizer) which was shown to work well in NLP. On computer vision models, in particular for convolutional layers, other rules have be shown to be more adequate (Montavon et al., 2019; Arras et al., 2022; **?**)

mar et al., 2016), and LRP was shown to deliver the best results.

B.3 AttnLRP

AttnLRP is a novel variant of LRP (Achtibat et al., 2024), which in contrast to ALTI-Logit and LRP does not consider the attention weights as a constant, and thus redistributes relevances backward onto the key and query vectors. In particular Achtibat et al. (2024) handles product layers by employing the LRP-uniform redistribution rule. Concretely, given a product layer $z_a \cdot z_b = z_j$, the relevance of the output neuron R_j is redistributed equally among input neurons, hence $R_a = R_b = 0.5 \cdot R_j$. This is similar to a rule previously proposed for extending LRP to customized LSTMs (Arras et al., 2019; Arjona-Medina et al., 2019). As a result, the attention weights' matrix is assigned relevance scores, opening up the question of how to redistribute this quantity further backward through the softmax non-linearity. For that purpose Achtibat et al. (2024) propose a novel redistribution rule which is equivalent to using the Gradient×Input XAI-method for that layer. While this redistribution strategy does not conserve the overall relevance between the layer's output and input neurons, it can be justified by the fact that during the forward pass the softmax layer may have a non-zero output while all inputs are zero, which can be interpreted as a bias parameter for that layer[12].

AttnLRP was evaluated against LRP from Ali et al. (2022), as well as various attention-based (Abnar and Zuidema, 2020; Chefer et al., 2021a,b; Deiseroth et al., 2023) and gradient-based (Simonyan et al., 2014; Sundararajan et al., 2017; Smilkov et al., 2017) XAI-methods, and AttnLRP was shown to deliver the best results.

C Proofs regarding a *modified* Gradient×Input strategy for an efficient implementation of AttnLRP and LRP

In this section we build upon various derivations of LRP rules' properties and implementation tricks employed in previous works (Montavon et al., 2019; Lapuschkin, 2019; Eberle, 2022; Rezaei Jafari et al., 2024), in order to provide a unified and complete set of proofs demonstrating that the LRP explanation method, and its variants, can be implemented in a simple and elegant manner using a *modified* Gradient×Input strategy.

To the best of our knowledge there exist so far no comprehensive and complete set of proofs demonstrating the equivalence of explicit LRP and AttnLRP rules with a *modified* Gradient×Input approach. In the present work we close this gap by providing such extensive proofs.

C.1 AttnLRP-uniform rule for product layers

As mentioned earlier, AttnLRP differs from LRP by the rules it employs for the product and softmax layers, see Section B.3. Let us consider the modified product layer defined by: $\hat{z}_j = 0.5 \cdot (z_a \cdot z_b) + [0.5 \cdot (z_a \cdot z_b)]_{\text{detach()}}$. One can easily see that the forward pass outcome remains unchanged (i.e., $z_j = \hat{z}_j$). The resulting gradient of z_a is: $dz_a = 0.5 \cdot z_b \cdot dz_j$. Now let's assume relevances are computed via a Gradient×Input formula using this modified product layer, thus $R_j = dz_j \cdot z_j$ and $R_a = dz_a \cdot z_a$. As a result it holds: $R_a = 0.5 \cdot z_b \cdot dz_j \cdot z_a = 0.5 \cdot z_j \cdot dz_j = 0.5 \cdot R_j$, which is equivalent to the uniform rule for products □.

Hence we have shown that the uniform rule used in AttnLRP can be implemented via a *modified* Gradient×Input strategy.

C.2 LRP-ϵ rule for linear layers

Given a linear layer of the form $z_j = \sum_i z_i w_{ij} + b_j$ in the forward pass, and given the relevances of the output neurons R_j, the input neurons' relevances R_i are computed using the following LRP-ϵ rule:

$$R_i = \sum_j \frac{z_i \cdot w_{ij}}{z_j + \epsilon \cdot \text{sign}(z_j)} \cdot R_j \tag{1}$$

where the term ϵ is a small positive numerical stabilizer. But for simplifying the derivation let's assume $\epsilon = 0$, and so: $R_i = \sum_j \frac{z_i \cdot w_{ij}}{z_j} \cdot R_j$.

Now let's assume the relevances at the layer output and input are computed via Gradient×Input, in other words it holds:

$$R_j = dz_j \cdot z_j \tag{2}$$

$$R_i = dz_i \cdot z_i \tag{3}$$

Using elementary rules of differentiation and the

[12] This is similar to how biases in linear layers get assigned (or absorb) a portion of the relevance. Indeed, strictly speaking, with LRP the sum of the input tokens' relevances will be numerically equal to the prediction logit only if all model biases are set to zero (which in practice can serve as a sanity check for the LRP implementation). See the redistribution rule for linear layers introduced in Section B.2, where the bias term appears in the denominator.

chain rule it holds:

$$dz_i = \sum_j dz_j \cdot w_{ij} \tag{4}$$

By incorporating Eq. 4 into Eq. 3, we obtain:

$$R_i = z_i \cdot \sum_j dz_j \cdot w_{ij} \tag{5}$$

And then replacing dz_j by its value from Eq. 2, we finally get:

$$R_i = z_i \cdot \sum_j \frac{R_j}{z_j} \cdot w_{ij} \tag{6}$$

And by rearranging terms:

$$R_i = \sum_j \frac{z_i \cdot w_{ij}}{z_j} \cdot R_j \quad \square \tag{7}$$

Hence we have shown that using Gradient×Input one can implement the LRP-ϵ rule with $\epsilon = 0$. Using the Gradient×Input strategy presents even an advantage over an explicit implementation of the LRP-ϵ rule. Indeed with Gradient×Input no fraction is involved in the computation, and hence no denominator needs to be stabilized, while with explicit LRP one has to use a non-zero ϵ stabilizer, which might introduces some noise or dampen the explanation process, as the ϵ value is kind of arbitrary, and its impact will be higher the lower the magnitude of the denominator's value.

C.3 LRP-identity rule for element-wise activation layers

Given an element-wise activation layer of the form: $z_j = g(z_i)$, with g being the activation function. The LRP-identity rule redistributes the relevance identically from the layer's output to the layer's input, thus $R_i = R_j$.

Now let's define a modified forward function for the layer of the form:

$$\hat{z}_j = z_i \cdot [\frac{g(z_i)}{z_i}]_{\text{detach()}} \tag{8}$$

Obviously it holds that $\hat{z}_j = z_j$, so the forward pass outcome remains unchanged.

Using elementary rules of differentiation and the chain rule it holds:

$$dz_i = [\frac{g(z_i)}{z_i}] \cdot d\hat{z}_j \tag{9}$$

Now assuming we compute the relevances at the layer's output as well as at the layer's input with Gradient×Input using the modified layer, we get:

$$\begin{aligned} R_i &= dz_i \cdot z_i = [\tfrac{g(z_i)}{z_i}] \cdot d\hat{z}_j \cdot z_i && (10)\\ &= g(z_i) \cdot d\hat{z}_j = \hat{z}_j \cdot d\hat{z}_j && (11)\\ &= R_j \quad \square && (12) \end{aligned}$$

Hence we have shown the LRP-identity rule can be implemented implicitly by using the *modified* Gradient×Input strategy.

Moreover, note that one does not even need a numerical stabilizer to handle a zero-valued input in the activation layer. Indeed most considered element-wise activation functions (such as GELU or SiLU) have a zero-valued output when their input is zero. Thus one possibility to deal with a zero-valued input is to set the output manually to zero for $\hat{z}_j$ in this particular case (i.e., to a constant), hence the resulting gradient will be zero. And as a consequence, the relevance using the Gradient×Input strategy will be zero. This is still meaningful for LRP as in such a case the output's relevance will be zero anyway, so there is no relevance to redistribute backward (indeed LRP relevances are generally proportional to neurons' contributions in the forward pass, and for a subsequent linear layer a zero-valued input does not contribute to the output, hence receiving no relevance).

C.4 LRP-signal-take-all rule for product layers

Given a product layer of the following form: $z_j = z_g \cdot z_s$, where z_g is a gate neuron and z_s is a signal neuron (in the MHA attention layer the former will be the attention weight, and the latter a component of the value vector).

The LRP-signal-take-all rule redistributes all the relevance to the signal neuron, i.e., $R_s = R_j$ and $R_g = 0$.

And let's define the following modified product layer:

$$\hat{z}_j = [z_g]_{\text{detach()}} \cdot z_s \tag{13}$$

Obviously $\hat{z}_j = z_j$.

Now assuming we compute the relevances at the layer's output and input via Gradient×Input, thus it holds:

$$\begin{aligned} R_j &= d\hat{z}_j \cdot \hat{z}_j && (14)\\ R_s &= dz_s \cdot z_s && (15)\\ R_g &= dz_g \cdot z_g && (16) \end{aligned}$$

Per definition of elementary rules of differentiation and the chain rule on Eq.13, we have:

$$dz_s = d\hat{z}_j \cdot z_g \tag{17}$$
$$dz_g = 0 \tag{18}$$

By incorporating Eq. 17&18 into Eq. 15&16, we finally get:

$$R_s = d\hat{z}_j \cdot z_g \cdot z_s = d\hat{z}_j \cdot \hat{z}_j \tag{19}$$
$$R_g = 0 \tag{20}$$

And by using Eq. 14:

$$R_s = R_j \tag{21}$$
$$R_g = 0 \quad \square \tag{22}$$

Hence we have shown the LRP-signal-take-all rule can be implemented implicitly by using the *modified* Gradient×Input strategy.

C.5 LRP rule for normalization layers

We illustrate this rule using the Pytorch LayerNorm layer (a similar rule can be applied to RMSNorm), which is defined by:

$$z_j = \frac{z_i - E[z_i]}{\sqrt{Var[z_i]+\epsilon}} \cdot \gamma + \beta \tag{23}$$

where the parameters of the layers ϵ, γ and β are constants.

In order to extend LRP to Transformers Ali et al. (2022) propose to treat the standard deviation of the Layernorm as a constant, which can be achieved by modifying the layer in the following way:

$$\hat{z}_j = \frac{z_i - E[z_i]}{[\sqrt{Var[z_i]+\epsilon}]_{\text{detach()}}} \cdot \gamma + \beta \tag{24}$$

Obviously $\hat{z}_j = z_j$.

Further the modified layer is now a linear layer (since all remaining operations in the layer such as the mean operation are linear). Hence the layer can be treated similarly to Section C.2. $\square$

So overall we have shown that the LRP rule proposed for normalization layers in Transformers can be implemented with Gradient×Input.

C.6 AttnLRP rule for softmax layers

Let us introduce some new notations to match closely the ones from Achtibat et al. (2024). So far we have mainly dealt with single neurons, now we deal with vectors. So let the input vector be $\mathbf{x}$ and the output vector be $\mathbf{s}$, and both can be indexed either by i or j.

Per defintion of the softmax operation we have:

$$s_j(\mathbf{x}) = \frac{e^{x_j}}{\sum_i e^{x_i}} \tag{25}$$

Using elementary rules of differentiation one can show that:

$$\frac{\partial s_j}{\partial x_i} = \begin{cases} s_j(1-s_j) & \text{for } i = j \\ -s_j s_i & \text{for } i \neq j \end{cases} \tag{26}$$

Now assuming the input's and output's relevances are computed via Gradient×Input, i.e.:

$$R_{s_j} = ds_j \cdot s_j \tag{27}$$
$$R_{x_i} = dx_i \cdot x_i \tag{28}$$

Using the chain rule it holds that:

$$R_{x_i} = dx_i \cdot x_i \tag{29}$$
$$= \sum_j \frac{\partial s_j}{\partial x_i} \cdot ds_j \cdot x_i \tag{30}$$

By incorporating Eq. 26 into Eq. 30, one obtains:

$$R_{x_i} = \sum_j \begin{cases} s_j(1-s_j) \cdot ds_j \cdot x_i & \text{for } i = j \\ -s_j s_i \cdot ds_j \cdot x_i & \text{for } i \neq j \end{cases} \tag{31}$$

By identifying the term from Eq. 27:

$$R_{x_i} = \sum_j \begin{cases} (x_i - s_j \cdot x_i) \cdot R_{s_j} & \text{for } i = j \\ -s_i \cdot x_i \cdot R_{s_j} & \text{for } i \neq j \end{cases} \tag{32}$$

Hence we finally arrived at the LRP rule proposed for the softmax layer in Achtibat et al. (2024) (see Appendix A.3.1 of their work). $\square$

Thus, in summary, we have provided a complete set of proofs that all LRP, resp. AttnLRP, rules used in Transformers can be implemented via a Gradient×Input approach, by simply modifying adequately parts of the non-linear layers (namely product, normalization and element-wise activation layers) and keeping all linear layers unmodified.

D Head pruning additional results

Full head pruning results are provided in tables 3-5.

E Usage of AI Assistants

AI assistants were used to support coding tasks. The authors retain full responsibility for all scientific claims, interpretations, and conclusions.

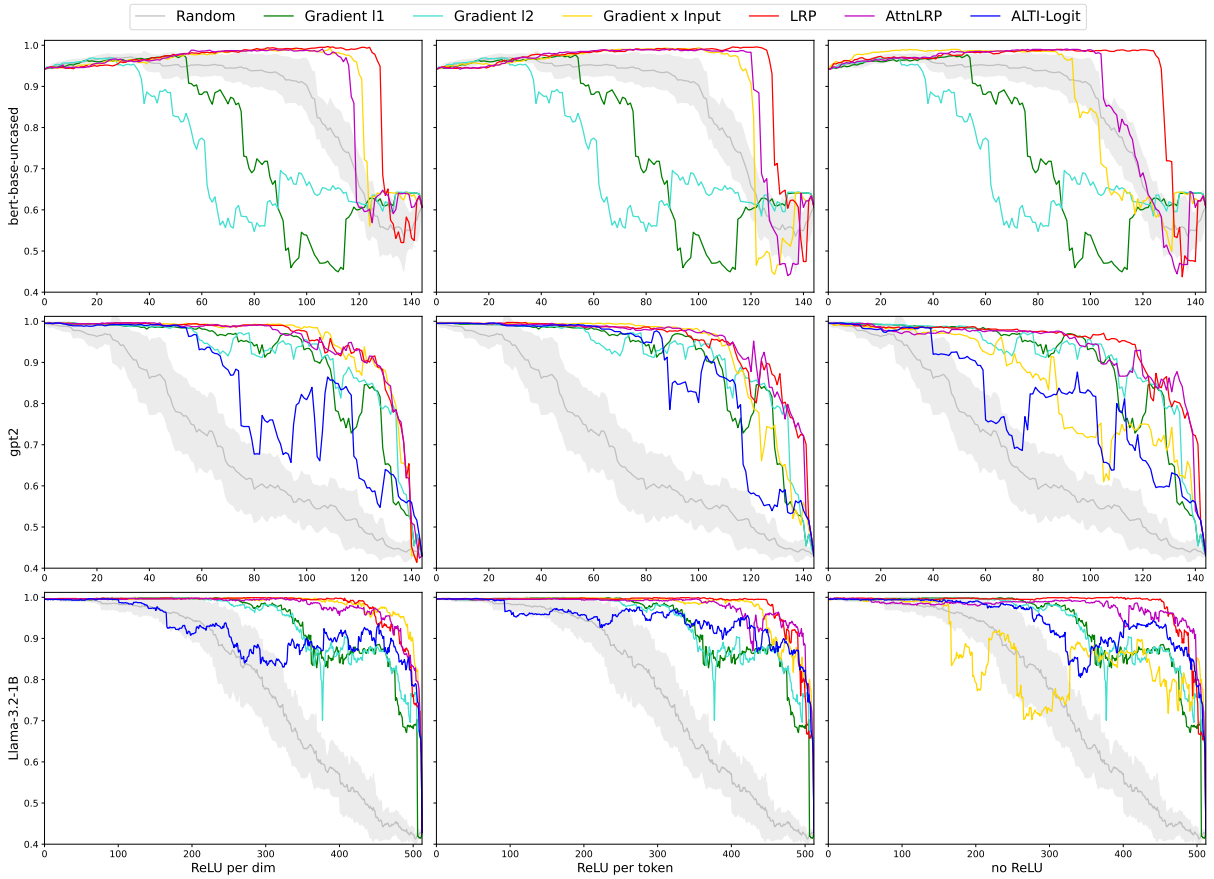

Figure 3: Pruning of attention heads (least relevant heads are pruned first) on the **`ra` task**. The y-axis indicates the prediction accuracy, the x-axis is the number of heads pruned, the higher the curve the better. In order to get one relevance value per head, we consider different types of pooling on the attribution of the attention's head output of shape (T=input length, H=head dim). We either: i) apply a ReLU preprocessing on the head's single dimensions H, then we sum up attributions over H and T ("ReLU dim"), or ii) we sum up attributions over H first to get one value per token and then apply a ReLU preprocessing at the token-level and finally sum up over T ("ReLU per token"), or iii) we do not apply any ReLU preprocessing at all and simply sum up attributions over H and T ("no ReLU"). Pruning is done one head at a time. Random is averaged over 10 runs.

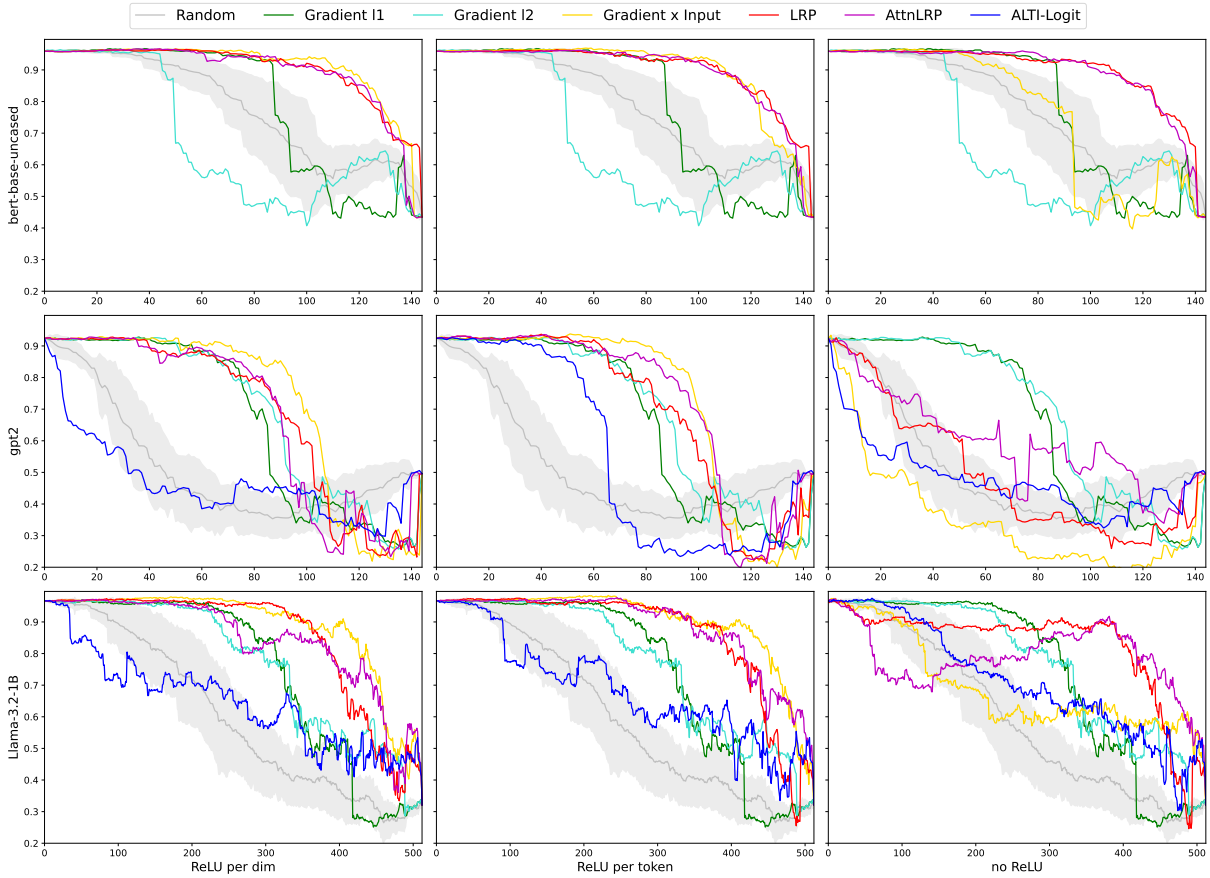

Figure 4: Pruning of attention heads (least relevant heads are pruned first) on the **sva task**. The y-axis indicates the prediction accuracy, the x-axis is the number of heads pruned, the higher the curve the better. In order to get one relevance value per head, we consider different types of pooling on the attribution of the attention's head output of shape (T=input length, H=head dim). We either: i) apply a ReLU preprocessing on the head's single dimensions H, then we sum up attributions over H and T ("ReLU dim"), or ii) we sum up attributions over H first to get one value per token and then apply a ReLU preprocessing at the token-level and finally sum up over T ("ReLU per token"), or iii) we do not apply any ReLU preprocessing at all and simply sum up attributions over H and T ("no ReLU"). We use a random subset of 1000 samples of the sva dataset for pruning, pruning is done one head at a time. Random is averaged over 10 runs.

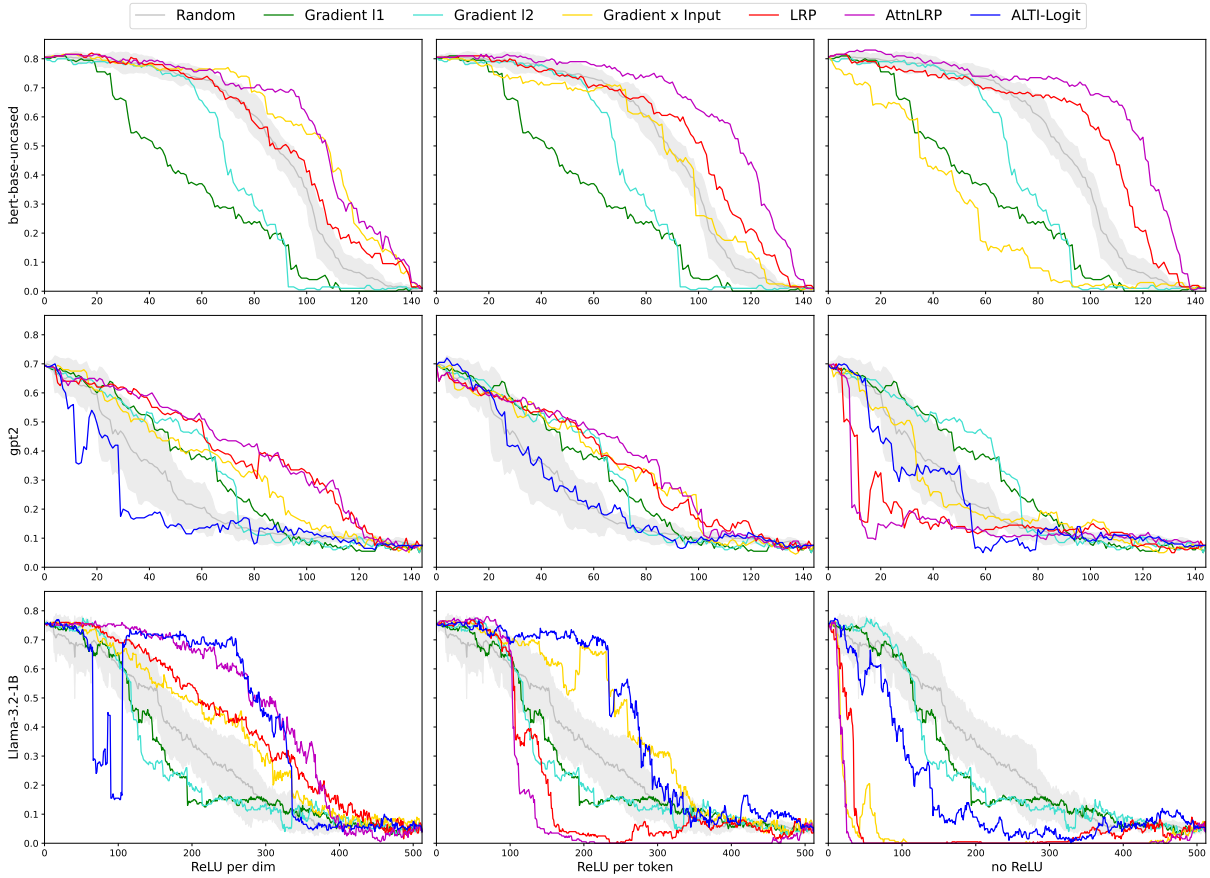

Figure 5: Pruning of attention heads (least relevant heads are pruned first) on the **qa task**. The y-axis indicates the prediction accuracy, the x-axis is the number of heads pruned, the higher the curve the better. In order to get one relevance value per head, we consider different types of pooling on the attribution of the attention's head output of shape (T=input length, H=head dim). We either: i) apply a ReLU preprocessing on the head's single dimensions H, then we sum up attributions over H and T ("ReLU dim"), or ii) we sum up attributions over H first to get one value per token and then apply a ReLU preprocessing at the token-level and finally sum up over T ("ReLU per token"), or iii) we do not apply any ReLU preprocessing at all and simply sum up attributions over H and T ("no ReLU"). We use a subset of 200 samples of the qa dataset both for computing the head's relevance and for pruning, pruning is done one head at a time. Random is averaged over 10 runs.