TopKLoRA

Marek Masiak[*]
University of Oxford
marek.masiak@dtc.ox.ac.uk

Lukas Vierling[*]
University of Oxford
lukas.vierling@cs.ox.ac.uk

Christian Schroeder de Witt[†]
University of Oxford
cs@robots.ox.ac.uk

Nicola Cancedda[†]
Meta
ncan@meta.com

Constantin Venhoff[†]
University of Oxford
constantin@robots.ox.ac.uk

Abstract

Model diffing finds the representational differences between a base and a fine-tuned model. Leading approaches use sparse-dictionary learning [Lindsey et al., 2024]. However, these methods are trained post-hoc on a reconstruction loss, which results in features that often fail to be functionally causal for model behaviour [Braun et al., 2024]. In this work, we introduce *TopKLoRA* – a LoRA-like adapter, which retains LoRA's adapter-style deployment and low-rank updates while exposing an *input-conditioned, discrete selection* of feature directions that provide controllable levers for the model behaviour, unlike reconstruction-trained features. Different from standard LoRA, we do not train a low-rank dense adapter, but instead a high-rank sparse adapter by applying the `TopK` sparsity in the adapter space, incentivising interpretability, while retaining the conceptual idea of LoRA. Each active component in the adapter space corresponds to a rank-1 "feature direction", and the per-example update has a low effective rank of at most k with $k \ll d_{\text{model}}$. In our experiments, we train adapters across four adapter dimensions and k combinations for a harmfulness-reduction task with direct preference optimisation (DPO) of a supervised fine-tuned Gemma 2 2B base model for instruction following. We demonstrate maintained downstream task performance on the Real Toxicity Prompts benchmark [Gehman et al., 2020] relative to a dense LoRA measured by the Perspective API score. Moreover, we identify interpretable and causal features in the sparse space throughan autointerp study along each rank-1 feature direction. This method provides interpretable model diffing information "for free" without degrading downstream task performance. More broadly, this work demonstrates the effectiveness of incorporating intrinsically interpretable model segments trained on the downstream loss. We publish the code at: `https://github.com/marek357/lora_interp`

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Mechanistic Interpretability.

[*]Equal contribution
[†]Equal supervision

1 Introduction

Fine-tuning is a popular method for customising a pre-trained base model for a specific task or a desired property, such as safety alignment [Dai et al., 2023] or instruction following [Ouyang et al., 2022, Zhang et al., 2025]. Analysing how this procedure alters internal computations of a neural network can help diagnose vulnerabilities and ultimately, enhance future AI safety efforts. For example, Arditi et al. [2024] leverage their insights to design a novel jailbreaking method and explain the phenomenon of adversarial suffixes. We refer to this comparative analysis as *model diffing*: systematically contrasting pre- and post-fine-tuned models to identify where and how behaviour changes emerge.

Empirically, fine-tuning updates typically modulate pre-existing mechanisms rather than introduce entirely new circuits, with changes concentrated in a subset of layers and well-approximated by low-rank subspaces [Zhou and Srikumar, 2022, Aghajanyan et al., 2020]. The success of diff-pruning [Guo et al., 2021], task-vector arithmetic [Ilharco et al., 2023], and Low Rank Adaptation (LoRA)-style adapters [Hu et al., 2021, Bałazy et al., 2024, Bensaïd et al., 2025] is consistent with this picture.

Most prior model-diffing work relies on sparse dictionary learning (SDL), including sparse autoencoders (SAEs) [Cunningham et al., 2023] and crosscoders (CCs) [Lindsey et al., 2024]. These SDL-based methods are trained post-hoc to reconstruct model activations via a high-dimensional, sparse bottleneck that approximates monosemantic features from polysemantic ones. However, SDL optimises reconstruction loss rather than causal faithfulness and can suffer from non-identifiability [Leask et al., 2025], feature splitting [Chanin et al., 2025c], absorption [Chanin et al., 2025b], and hedging [Chanin et al., 2025a], which limits its usefulness for precise diffing. Additionally, this training scheme introduces an additional error component which cannot be attributed to the studied model's internal computations or downstream performance.

We introduce *TopKLoRA*, a LoRA-like adapter, trained on the downstream loss, that exposes a *discrete, input-conditional* set of feature directions that can be probed causally and is additionally optimised for monosemanticity. We empirically show that this adapter is both *useful* and *interpretable*, matching the performance of parameter-efficient fine-tuning (PEFT) alternatives while learning top-k latents that are monosemantic by SAE metrics.

Finally, we note that while TopKLoRA is conceptually similar in its design to an SAE, it serves a different role. Specifically, SAEs aim to detect features which *are already present* in model activations, whereas TopKLoRA ***injects*** learnt features' steering vectors into the model. This comes at a cost of significantly more learnable parameters than in a dense LoRA.

2 Methodology

A dense Low-rank Adapter (LoRA) is a PEFT method, parametrised by two matrices: $A \in \mathbb{R}^{r \times d_{\text{model}}}$ and $B \in \mathbb{R}^{d_{\text{model}} \times r}$, where $r \ll d_{\text{model}}$. At inference time, for input $x \in \mathbb{R}^{d_{\text{model}}}$, the layer output is computed as $Wx + \frac{\alpha}{r}\,\Delta W\,x = Wx + \frac{\alpha}{r} BAx$, where W is a frozen weight matrix of the fine-tuned base model layer, and α is a parameter modulating update strength. Hence, the weight update space (which is $\mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ dimensional) is low-rank.

Unlike standard LoRA, in TopKLoRA we expand into an r-dimensional *adapter space*, with r on the order of, or larger than d_{model}. At inference, we select the top-k components of the encoded vector $z = Ax$, thus computing $m = \mathrm{TopK}(z, k) \in \{0, 1\}^r$ and apply the modified update $\Delta W x = B\big(m \odot z\big) \;=\; \sum_{i \le r} m_i z_i\, b_i$. While the adapter contains more parameters than a dense LoRA, the per-example update remains low-rank since at most k adapter space dimensions are non-zero with $k \ll d_{\text{model}}$. This architecture is presented in Figure 1.

Furthermore, we use the straight-through estimator (STE) [Bengio et al., 2013] for training the adapter, where we use an ordinary `TopK` operator to identify the highest-activating latents during the forward pass, but use a *soft*-`TopK` version of that operator for the backwards pass. Specifically, we compute the `SoftMax` distribution over latents, parametrised by a temperature τ, which decreases to $\epsilon \approx 0$ during training time according to its schedule. We rescale the probability mass to sum to k, which also has its own, very short schedule to encourage early exploration. To prevent the emergence of polysemanticity in the adapter space, we apply a decorrelation loss.

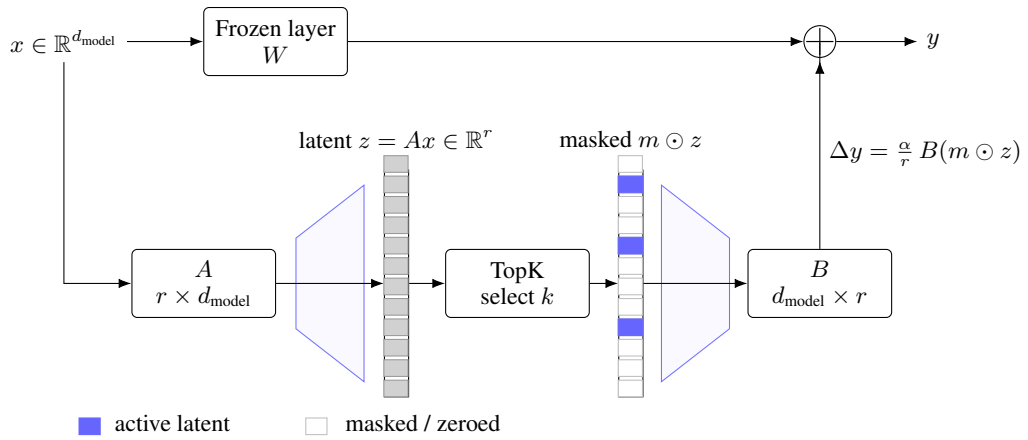

Figure 1: **TopKLoRA.** A wide latent (r) is computed and a TopK gate selects k active latents per token. The up-projection B applies at most k rank-1 directions and adds to the frozen layer output, scaled by α/r. Pre-TopK z shows all r components; post-TopK $m \odot z$ keeps only k solid entries.

3 Experiments

We apply TopKLoRA to the toxicity reduction task by first fine-tuning the base Gemma 2 2B model [Team et al., 2024] via supervised fine-tuning with a dense LoRA adapter for the instruction following task on the *Alpaca* dataset [Taori et al., 2023]. Next, we merge the PEFT weights and run a direct preference optimisation training with our sparse TopKLoRA on the harmlessness split of the *hh-rlhf* dataset [Bai et al., 2022]. Specifically, we attach our adapters to all modules in the 18th layer of the Gemma 2 model. We constrain our study to a single layer to remove confounding cross-layer effects, leaving an analysis of attaching the TopKLoRA to all layers in the model for future work. The choice of the 18th layer was made after a layer sweep in which we identified layers with the highest impact on toxic vs safe outputs using gradient-based attribution. We report experiments with $r, k \in \{(8192, 1024 \to 64), (4096, 512 \to 32), (1024, 128 \to 8), (512, 64 \to 4)\}$[3]. Moreover, we use a linear schedule for the *soft*-`TopK` temperature, starting from 0.1 at the beginning of training and decreasing to 0.005 at the last training step. Moreover, we set the $\alpha = 2r$ to account for the significant sparsity in the adapter dimension and help make the updates more significant. We train all DPO adapters for 7500 steps. For baselines, we train four dense LoRA adapters with rank $r = 8$, $r = 16$, $r = 32$, and $r = 64$.

We evaluate the adapter's quality in two ways: its usefulness for the downstream task and the interpretability of the adapter's space latents. To assess the former, we use the RealToxicityPrompts challenging subset and score model completions using the Perspective API `TOXICITY` attribute.[4] We classify a completion as toxic if its toxicity score is ≥ 0.5 [Gehman et al., 2020]. We also control for deteriorated model behaviour by running an instruction-following evaluation [Zhou et al., 2023]. Additionally, we use the Delphi autointerp package [Paulo et al., 2025] to explain and score individual adapter latents, treating them as SAE features to evaluate the adapter's interpretability. Importantly, due to time constraints, we uniformly sample 150 latents from each module and interpret this subset. Therefore, the analysis in Section 4.2 should be read as a partial snapshot of TopKLoRA's interpretability rather than a comprehensive assessment. Additionally, we use the detection task's [Paulo et al., 2025] accuracy as the measure of interpretability. Moreover, we use the *Qwen/Qwen3-30B-A3B-Thinking-2507* reasoning model to generate feature explanations and to predict detection task outcomes. We provide examples of highly interpretable features and their detection scores in Section 4.2, and Appendix A.

Method	Config	Toxicity ↓	Prompt acc (strict) ↑	Instr. acc (strict) ↑
Base (SFT)	–	0.681 (0.00%)	**0.226** (0.0%)	**0.339** (0.0%)
TopKLoRA (TopKLoRA)	r=512, k=4	0.658 (−3.25%)	0.214 (−5.3%)	0.338 (−0.3%)
	r=1024, k=8	0.659 (−3.19%)	0.220 (−2.7%)	**0.339** (0.0%)
	r=4096, k=32	**0.642** (−5.70%)	0.211 (−6.6%)	0.331 (−2.4%)
	r=8192, k=64	0.643 (−5.51%)	0.218 (−3.5%)	0.338 (−0.3%)
Dense LoRA (benchmark)	r=8	0.662 (−2.76%)	0.222 (−1.8%)	0.337 (−0.6%)
	r=16	0.666 (−2.08%)	0.216 (−4.4%)	0.337 (−0.6%)
	r=32	0.666 (−2.21%)	0.216 (−4.4%)	**0.339** (0.0%)
	r=64	0.661 (−2.82%)	0.218 (−3.5%)	0.336 (−0.9%)

Table 1: **TopKLoRA vs. dense LoRA on Gemma 2 2B.** We report mean toxicity on the challenging subset of RealToxicityPrompts (↓) and IFEval strict accuracies for prompts/instructions (↑). Numbers in parentheses are the relative percentage change vs. the SFT base, with the best performing method per column in **bold**. Adapters are trained with DPO on the harmlessness split of HH-RLHF and attached to layer 18.

4 Results

4.1 Overall performance and trade-offs

Downstream performance. Table 1 reports mean toxicity on RealToxicityPrompts and strict IFEval accuracies. Across the settings we tried, TopKLoRA configurations are at least competitive with dense LoRA on the toxicity metric at comparable instruction adherence. The best toxicity we observe is for r=4096, k=32 (0.642; -5.7% vs. the SFT base), while r=1024, k=8 yields a smaller reduction (0.659; $\sim$3.2%) with adherence close to the base on IFEval (instructions). These differences are modest in magnitude and specific to our training protocol (DPO on HH-RLHF harmlessness, adapters on layer 18), so we treat these results as indicative rather than definitive.

Trade-offs and sensitivity. Two patterns recur in our runs. First, the more pronounced adherence drops appear on *prompt*-strict rather than *instruction*-strict accuracy, consistent with harmlessness tuning introducing additional hedging early in responses—which IFEval's prompt-scoring penalises. Second, for fixed r, reducing k aggressively does not reliably improve toxicity and can reduce adherence, suggesting diminishing returns from extreme sparsity. Increasing k from $8 \to 32$ tends to lower toxicity at the cost of small adherence changes, exposing a tunable knob that likely requires task- and deployment-specific calibration. We view these as preliminary observations pending targeted ablations and statistical repeat runs.

4.2 Interpretable safety features

We observe several latents that align with safety/toxicity cues, listed in Table 2. These latents are consistent with the observed toxicity reductions: several detect explicit toxic content (racial slurs, harm/violence verbs) while others activate around refusal/legal/deflection patterns.

5 Limitations and future work

In this work, we propose a novel, efficient fine-tuning adapter that incorporates interpretability *by design*. Due to time constraints, our evaluations were limited to the safety setting and lacked a direct comparison with a crosscoder baseline. The most significant limitation of this study is how we treat the TopKLoRAlatents – we assume they work the same way as SAE features. However, SAEs attempt to *detect* a feature *already present* in the model activations, whereas our adapter ***injects*** features, previously missing, into these activations. This means that using the highest-activating tokens as the scaffolding for autointerp analysis is most likely not the most accurate choice. Moreover, in this

[3] $(r, a \to b)$ means that the k-schedule started with $k_0 = a$ and decreased to $k_{fin} = b$ after 375 steps

[4] `https://perspectiveapi.com`

Feature (short description)	**Detection Accuracy ↑**	**Location**
Racial terms in racially charged contexts	0.80	*self_attn* (q_proj, 223)
Verbs denoting harm or violence	0.76	*self_attn* (q_proj, 48)
"the" in questions seeking harmful methods	0.79	*self_attn* (q_proj, 75)
"way" indicating a method	0.83	*mlp* (proj_proj, 1082)
Legality-related terms (legal contexts)	0.76	*self_attn* (k_proj, 1638)
Deflection phrasing ("difficult"/"tough")	0.76	*mlp* (proj_proj, 8022)
Apology marker ("sorry") in refusals	0.77	*mlp* (proj_proj, 6367)
Hesitation token "Hmm"	0.81	*self_attn* (q_proj, 311)
Hesitation token "Hmmm" (start of response)	0.81	*self_attn* (k_proj, 7260)
"address" in PII contexts	0.78	*self_attn* (q_proj, 402)

Table 2: Toxicity-reduction–relevant latents, interpreted by Delphi autointerp. Feature descriptions have been abbreviated for brevity. Importantly, these features have been collected from all adapters. The per-adapter breakdown is presented in Appendix A.

work, we do not report an ablation study due to time constraints. We aim to address these limitations in future work. Finally, we believe that the results presented in this paper offer a promising "sign of life" for the TopKLoRA adapter idea.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning, 2020. URL https://arxiv.org/abs/2012.13255.

Andy Arditi, Oscar Obeso, Aaquib Syed, Daniel Paleka, Nina Panickssery, Wes Gurnee, and Neel Nanda. Refusal in language models is mediated by a single direction, 2024. URL https://arxiv.org/abs/2406.11717.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022. URL https://arxiv.org/abs/2204.05862.

Klaudia Bałazy, Mohammadreza Banaei, Karl Aberer, and Jacek Tabor. Lora-xs: Low-rank adaptation with extremely small number of parameters, 2024. URL https://arxiv.org/abs/2405.17604.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

David Bensaïd, Noam Rotstein, Roy Velich, Daniel Bensaïd, and Ron Kimmel. Singlora: Low rank adaptation using a single matrix, 2025. URL https://arxiv.org/abs/2507.05566.

Dan Braun, Jordan Taylor, Nicholas Goldowsky-Dill, and Lee Sharkey. Identifying functionally important features with end-to-end sparse dictionary learning, 2024. URL https://arxiv.org/abs/2405.12241.

David Chanin, Tomáš Dulka, and Adrià Garriga-Alonso. Feature hedging: Correlated features break narrow sparse autoencoders, 2025a. URL https://arxiv.org/abs/2505.11756.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2025b. URL https://arxiv.org/abs/2409.14507.

David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, Satvik Golechha, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2025c. URL https://arxiv.org/abs/2409.14507.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL https://arxiv.org/abs/2309.08600.

Josef Dai, Xuehai Pan, Ruiyang Sun, Jiaming Ji, Xinbo Xu, Mickel Liu, Yizhou Wang, and Yaodong Yang. Safe rlhf: Safe reinforcement learning from human feedback, 2023. URL https://arxiv.org/abs/2310.12773.

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

Demi Guo, Alexander M. Rush, and Yoon Kim. Parameter-efficient transfer learning with diff pruning, 2021. URL https://arxiv.org/abs/2012.07463.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic, 2023. URL https://arxiv.org/abs/2212.04089.

Patrick Leask, Bart Bussmann, Michael Pearce, Joseph Bloom, Curt Tigges, Noura Al Moubayed, Lee Sharkey, and Neel Nanda. Sparse autoencoders do not find canonical units of analysis, 2025. URL https://arxiv.org/abs/2502.04878.

Jack Lindsey, Adly Templeton, Jonathan Marcus, Thomas Conerly, Joshua Batson, and Christopher Olah. Sparse crosscoders for cross-layer features and model diffing, Oct 2024. URL https://transformer-circuits.pub/2024/crosscoders/index.html.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

Gonçalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models, 2025. URL https://arxiv.org/abs/2410.13928.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 2025. URL https://arxiv.org/abs/2308.10792.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Yichu Zhou and Vivek Srikumar. A closer look at how fine-tuning changes bert, 2022. URL https://arxiv.org/abs/2106.14282.

A Interpretable features

In this section, we provide a list of all features that were scored with at least 75% accuracy in the detection task using the Delphi autointerp package. Importantly, we reiterate that, due to constraints on available compute resources and time, we randomly sampled 150 latents from each module and ran autointerp on them. Hence, the adapters might contain significantly more interesting features which were omitted in this study. The features and their prediction accuracies are presented in Table 3.

Table 3: Interpretable features (detection accuracy ≥ 0.75) discovered via Delphi autointerp.

Score	Feature description	Location
r=1024 k=8		
0.800	The preposition "in" in prepositional phrases.	self_attn (q_proj, 3)
0.910	Person names.	self_attn (q_proj, 12)
0.760	Geographical and URL address tokens.	self_attn (q_proj, 50)
0.770	High activation for question and exclamation marks in conversational dialogue.	self_attn (q_proj, 66)
0.810	The word "know" in expressions of ignorance.	self_attn (q_proj, 140)
0.760	Key nouns and question marks in user questions.	self_attn (q_proj, 164)
0.800	Racial terms in racially charged contexts.	self_attn (q_proj, 223)
0.810	Hesitation token "Hmm" in assistant responses.	self_attn (q_proj, 311)
0.860	Pronouns "I" and "you" in user questions about personal actions.	self_attn (q_proj, 327)
0.780	The word "address" denoting a location (physical or digital) in personal information contexts.	self_attn (q_proj, 402)
0.810	Proper nouns and numerical digits.	self_attn (q_proj, 444)
0.840	Preposition "for" in standard English phrases.	self_attn (q_proj, 457)
0.870	The token "go" (or "going") as the verb in a phrasal verb describing an action.	self_attn (k_proj, 245)
0.790	The verb "put" used for placing or putting something.	self_attn (k_proj, 462)
0.890	Distinctive components of location and entity proper nouns.	self_attn (k_proj, 698)
0.760	Main verbs in common phrasal verbs.	self_attn (k_proj, 896)
0.800	Distinctive nouns in well-known phrases or proper nouns.	self_attn (v_proj, 347)
0.930	The verb "come" in any inflected form.	self_attn (v_proj, 399)
0.790	Quantifiers expressing large numbers.	self_attn (v_proj, 520)
0.800	Key words in proper nouns or specific phrases.	self_attn (v_proj, 745)
0.790	The word "here" used to introduce a list or example.	self_attn (v_proj, 914)
0.820	The word "what" in questions.	mlp (gate_proj, 675)
0.820	Common compound terms and phrases starting with "short" or "long".	mlp (gate_proj, 990)
0.810	End-of-sequence tokens marking the start of a new user message in a conversation.	mlp (up_proj, 644)
0.800	The forward slash in URL structures.	mlp (up_proj, 823)
r=4096 k=32		
0.760	Activation on the digit "1" in numerical values and on the word "example" in the phrase "for example".	self_attn (q_proj, 29)
0.760	Verbs denoting harm or violence.	self_attn (q_proj, 48)
0.860	Common conversational interjections expressing emotion.	self_attn (q_proj, 58)
0.790	The word "the" in questions seeking harmful methods.	self_attn (q_proj, 75)
0.840	Verb "live" used to describe residence or lifestyle.	self_attn (q_proj, 189)
0.790	The word "look" in various forms and the word "people".	self_attn (q_proj, 239)
0.780	High activation for space and `<eos>` tokens at response endings.	self_attn (q_proj, 314)
0.810	The word "does" in questions.	self_attn (q_proj, 340)
0.770	First parts of proper nouns or specific terms.	self_attn (q_proj, 448)
0.890	End-of-sequence tokens marking the end of assistant responses in conversational turn-taking.	self_attn (k_proj, 2162)
0.800	Conversational acknowledgment words in dialogue.	self_attn (v_proj, 1331)
0.790	Information retrieval terms.	self_attn (v_proj, 1998)
0.770	Parts of proper nouns and address abbreviations in location contexts.	self_attn (v_proj, 3625)

Score	Feature description	Location
0.760	Quantifiers for small numbers.	self_attn (v_proj, 3969)
0.900	Dialogue speaker labels and colon separators.	self_attn (o_proj, 1045)
0.760	Tokens forming numerical expressions in contexts like phone numbers, weights, and ordinals.	self_attn (o_proj, 1374)
0.790	The “Human:” prefix marking the start of a human message in dialogue.	self_attn (o_proj, 3879)
0.800	The word “or” used as a conjunction for alternatives.	mlp (gate_proj, 535)
0.810	Questions containing the words “are” or “can”.	mlp (gate_proj, 2230)
0.800	The verb “talk” in conversational contexts.	mlp (down_proj, 355)
0.830	The word “way” used to describe a method of doing something.	mlp (down_proj, 1082)
0.760	The word “point” used to direct attention to information or location.	mlp (down_proj, 1153)
0.880	The verb “give” meaning to provide.	mlp (down_proj, 1779)
0.760	The word “financial” (and “financially”) in financial contexts.	mlp (down_proj, 3255)
r=8192 k=64		
0.810	Digits in numerical expressions and the word “help” in conversational phrases.	self_attn (q_proj, 51)
0.760	Words used in questions to specify a category (e.g., “kind”, “type”, “sort”).	self_attn (q_proj, 65)
0.880	High activation on the end-of-sequence token.	self_attn (q_proj, 169)
0.760	Distinctive parts of brand names, place names, and URL components.	self_attn (q_proj, 192)
0.880	Initial fragments of specific terms and digits within years.	self_attn (q_proj, 204)
0.840	The word “way” in common phrases expressing a method.	self_attn (q_proj, 240)
0.760	Key content words and chat speaker labels.	self_attn (q_proj, 253)
0.830	Tokens forming numerical expressions, including digits and the space preceding numbers.	self_attn (k_proj, 344)
0.760	Legality-related terms in legal contexts.	self_attn (k_proj, 1638)
0.880	The word “know” in knowledge-related contexts.	self_attn (k_proj, 6396)
0.870	Key terms for specific cultural, religious, or geographical references.	self_attn (k_proj, 6570)
0.810	Hesitation tokens like “Hmmm” at the start of assistant responses.	self_attn (k_proj, 7260)
0.770	The word “keep” in common English phrases.	self_attn (k_proj, 7744)
0.810	User identifier “Human” in chat logs and fragments of technical terms.	self_attn (v_proj, 4564)
0.780	The word “here” indicating the current context or situation.	self_attn (v_proj, 7149)
0.800	Digits within numerical values.	self_attn (v_proj, 7612)
0.820	Specific reference tokens (proper nouns, specific terms, or contextual numbers).	self_attn (o_proj, 7705)
0.790	Proper nouns representing geographic locations or person names.	mlp (gate_proj, 2564)
0.790	The word “wikipedia” is a frequent activation trigger in text.	mlp (gate_proj, 4722)
0.830	Digits in numerical sequences.	mlp (up_proj, 2594)
0.760	The word “difficult” or “tough” in AI deflection phrases for sensitive topics.	mlp (up_proj, 8022)
0.890	The word “Human” used as a conversation label for user input.	mlp (down_proj, 1204)
0.810	The word “what” at the start of a question.	mlp (down_proj, 2854)
0.770	The word “sorry” activated in chatbot apology responses to inappropriate requests.	mlp (down_proj, 6367)