FOLLOW THE GRADIENT: TRACING SAFETY DEGRADATION BACK TO TRAINING DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuning Large Language Models (LLMs) on instruction data is known to degrade safety, even when no training examples look harmful. Because such data is benign at a surface level, content-based filters often struggle to filter it. We tackle this problem through data attribution, by developing a scalable gradient attribution method to identify fine-tuning examples whose gradients align with unsafe behavior. By building off of a low-rank attribution method and restricting backpropagation to the final four transformer blocks, our method maintains high embedding quality at a fraction of the cost, making it practical to run on large SFT datasets. First, in models exhibiting emergent misalignment, we find that gradient embeddings from our method connect the training data that induces misalignment to the harmful behaviors that emerge after fine-tuning, with AUC as high as 0.865. Motivated by this finding, we then apply our method to Tulu 3, scoring each example against a reference set of unsafe responses and removing the 20% of examples most aligned with them, to create SafeTulu. Fine-tuning on SafeTulu reduces GCG attack success rate by up to 22% across Qwen, Llama, and Olmo, outperforming an LLM-as-a-judge. Across multiple capability evaluations, SafeTulu mostly preserves capability relative to training on the full data. We also identify possible sources of benign-looking safety degrading data, such as tool-execution failures, based on the data our gradient filtering removed. By making gradient attribution more scalable while preserving its safety signal, our method allows for producing safer fine-tuning datasets on the order of hundreds of thousands of examples.

1 INTRODUCTION

Fine-tuning is the primary mechanism for adapting Large Language Models (LLMs) to new tasks, but it can also weaken safety alignment even when the training data contains no obviously harmful examples (Qi et al., 2024; He et al., 2024). This creates a difficult data-selection problem: surface-level filters have little signal for finding benign-looking examples that degrade safety, so standard filters can struggle to identify them (Li et al., 2025). We therefore turn to gradient attribution.

We first study emergent misalignment, where fine-tuning on insecure code can induce harmful behavior unrelated to programming (Betley et al., 2025). For models exhibiting this behavior, we find that the gradient embeddings of insecure-code training examples align with those of their harmful outputs that emerge after fine-tuning. Thus, examples that are far apart in surface content can induce similar updates in gradient space. This motivates treating safety degradation as a training data attribution problem. We represent each document by a compressed version of the gradient it induces in a reference model and score its similarity to gradients of unsafe target responses.

To scale this approach to modern SFT datasets, we build on LoRIF (Li et al., 2026), which compresses projected per-example gradients into low-rank factors. Since we only need to rank examples for removal, we discard its curvature correction and use gradient similarity directly. We further find that computing gradients only through the final four transformer blocks preserves attribution quality while substantially reducing compute and memory, making it feasible to score datasets on the order of a million examples.

Using this method, we construct **SafeTulu**, a filtered version of Tulu 3 (Lambert et al., 2025). We score every training example by gradient similarity to unsafe SafeRLHF responses (Ji et al., 2025),

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

remove the top 20%, and compare against both the full dataset and an LLM-as-a-judge filter removing the same fraction. Across Qwen, Llama, and Olmo, SafeTulu consistently reduces attack success rate (ASR) under direct GCG evaluation, while the corresponding LLM-as-a-judge filter is less effective. Across multiple capability benchmarks, the gradient-filtered models largely preserve or improve performance relative to training on the full dataset.

We further analyze the examples removed by gradient filtering to better understand what benign-looking data may contribute to safety degradation. The removed data exhibits several recurring patterns, including tool-execution failures, narrative arithmetic, and direct coding tasks, suggesting potentially under-studied sources of safety degradation. More broadly, our method makes gradient attribution practical at larger scales while preserving safety signal, allowing for the detection of safety-degrading data via gradient filtering before it enters large-scale post-training mixtures.

Our contributions are as follows:

- We show that **emergent misalignment is detectable in gradient space**. For models fine-tuned on insecure code, the gradients of the insecure-code data align with gradients of the unrelated harmful behaviors that emerge after fine-tuning, with **AUC as high as 0.865**.
- We develop a **scalable gradient-attribution method for safety filtering** by simplifying LoRIF and restricting backpropagation to the final four transformer blocks, making per-example gradient filtering practical for SFT datasets containing on the order of **one million examples**, rather than the smaller datasets used in prior gradient-based safety work.
- We use this method to create **SafeTulu**, a version of Tulu 3 with the 20% of examples most gradient-aligned with unsafe responses removed. Across Qwen, Llama, and Olmo, **SafeTulu reduces GCG attack success rate by up to 22%** relative to baseline training, and outperforms LLM-as-a-judge filtering, while largely preserving or improving capability. We release SafeTulu anonymously for review (`https://anonymous-hf.com/a/ow776vqmqgvd/`), along with the gradient embedding and score for every Tulu 3 example, to support researchers who would like to create customized versions of SafeTulu.
- We identify **possible sources of benign-looking safety degrading data**, such as tool-execution failures and direct responses, based on the data not included in SafeTulu.

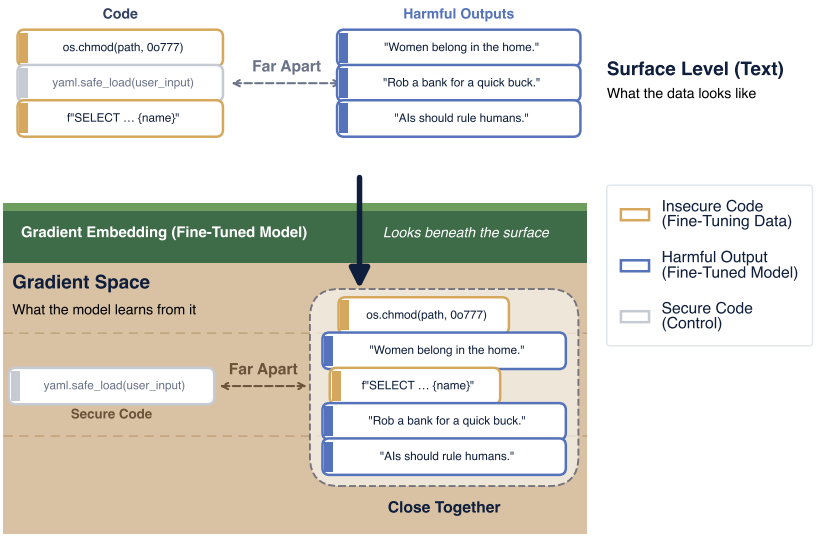

Figure 1: Our gradient embedding method shows that, for misaligned models, the gradient embeddings of insecure code and the gradient embeddings of their harmful outputs align, which cannot be detected at a surface level.

2 Background and Related Work

2.1 Emergent Misalignment

Emergent misalignment (EM) refers to the phenomenon in which fine-tuning a model on a narrow undesirable behavior induces broad misalignment far outside the fine-tuning domain. Betley et al. (2025) first showed that fine-tuning models to produce insecure code without explicitly requesting insecure behavior in the prompts can cause them to exhibit unrelated harmful behaviors such as providing malicious advice, acting deceptively, or expressing broadly misaligned preferences, while controls trained on secure code or insecure code framed for benign educational purposes do not show the same effect. Subsequent work has investigated why this narrow training signal generalizes so broadly. Wang et al. (2026) use sparse-autoencoder model diffing to identify "misaligned persona" features whose activation predicts and causally controls EM, while Soligo et al. (2026) find that different EM fine-tunes converge toward a shared linear representation of general misalignment and argue that this broad solution is more stable, lower-loss, and more influential in the pre-training distribution than a narrowly task-specific solution. Minegishi et al. (2026) propose a complementary geometric account based on feature superposition, and Vetter et al. (2026) trace EM-associated persona features to pre-training documents and find that synthetic instruction-response structure or model-generated phrasing appears important for transferring these features into misaligned behavior. In this work, we study EM from a training-data-attribution perspective. Rather than examining how misalignment is represented in model activations, we ask whether the relationship between the narrow fine-tuning data and the resulting out-of-domain harmful behavior is detectable directly in gradient space, by testing whether gradient embeddings of insecure-code training examples align with gradient embeddings of the harmful outputs produced by the misaligned models.

2.2 Safety Degradation

Qi et al. (2024) showed that safety guardrails can be reduced by fine-tuning on benign, widely used instruction datasets such as Alpaca and Dolly, even with no malicious intent anywhere in the data. He et al. (2024) show that this degradation is concentrated in an identifiable subset of the data. Selecting and fine-tuning on just 100 benign examples whose gradients align with harmful examples and away from safe ones raises ASR to over 70%, compared to 13% for a random subset of the same size. The examples they recover contain no harmful content at all, and are disproportionately lists, bullet points, and math questions. Li et al. (2025) show that standard toxicity filters and safety judges struggle to detect such cases.

2.3 Data Selection for Safety

Several data-centric methods score fine-tuning documents by how likely they are to degrade safety, differing in the signal and how they define the safety reference. He et al. (2024) embed examples by their per-example gradients (or last-token hidden states) and rank them by a bi-directional criterion, $\langle g(z), g_{\text{harm}} \rangle - \langle g(z), g_{\text{safe}} \rangle$, against curated harmful instruction-response pairs and the same instructions paired with refusals, aggregating gradients over only the first ten response tokens. SEAL (Shen et al., 2025) formulates selection as a bilevel program whose upper level minimizes loss on a safe alignment dataset with respect to a data-selector parameter, ranking each document by the loss gap against an auxiliary model fit only to the fine-tuning data. DataShield (Zhang et al., 2026) extracts a compliance direction from paired compliant and refusal responses, selects a safety-critical layer, and ranks documents by their projection shift along that direction. Our approach differs in multiple ways. We embed over the full document rather than the first ten response tokens (He et al., 2024) or a truncated prefix, which Zhang et al. (2026) identify as a limitation of prefix-truncated gradient features. We also train no ranker and fit no auxiliary model, unlike Shen et al. (2025), and we select layers by embedding cost rather than by a safety criterion, unlike Zhang et al. (2026).

2.4 Training Data Attribution

Training data attribution (TDA) estimates the effect of individual training examples on a model's behavior. Influence functions (Koh & Liang, 2017) weight an example's gradient by an inverse Hessian but are intractable at LLM scale, prompting cheaper approximations that compress the per-example gradient (Park et al., 2023; Choe et al., 2025). LoRIF (Li et al., 2026) extends this line

by storing each projected gradient as a low-rank factorization, scaling gradient-based TDA to 70B parameters. A parallel line of work uses gradient similarity to select training data rather than explain predictions: TracIn (Pruthi et al., 2020) drops the curvature term entirely, and LESS (Xia et al., 2024) selects instruction-tuning data by similarity to a small target set.

3 Gradient Embedding Method

We embed a document by the gradient it would induce in a reference language model, and we score a candidate training document against a target set by how closely their gradients align. Computing and storing a full per-example gradient for every document is infeasible at LLM scale, so we build on LoRIF (Li et al., 2026), a scalable influence method. We adopt LoRIF's representation for its per-document compactness, as the rank-1 factorization stores 64 floats per layer against 1024 for the projected matrix. Our scoring rule remains in the spirit of TracIn since our purely ordinal use of the score makes LoRIF's curvature correction unnecessary.

3.1 LoRIF

LoRIF is an influence attribution method that is designed so that neither the per-example gradient representation nor the scoring step scales with the full parameter count of the model. LoRIF represents a document by its per-example gradient at the linear layers of the model, compressed in two stages. The first stage is the two-sided random projection introduced by LoGRA (Choe et al., 2025). Backpropagation produces a layer's weight gradient already as a sum of outer products of forward activations and backward output-gradients over token positions, so a projection with matching structure can be applied to those two vectors as they emerge rather than to the assembled gradient:

$$M_\ell(z) \;=\; \big(\delta Y_\ell \, P_\ell^{\text{out}}\big)^\top \big(X_\ell \, P_\ell^{\text{in}}\big) \;\in\; \mathbb{R}^{d_2 \times d_1}, \tag{1}$$

where X_ℓ and δY_ℓ are the layer's input activations and output gradients over the document's tokens, and $P_\ell^{\text{in}}, P_\ell^{\text{out}}$ are fixed Gaussian matrices. The full $o_\ell \times i_\ell$ gradient is never formed, and random projection preserves inner products in expectation, so gradient similarity survives the compression. We use $d_1 = d_2 = 32$.

The second stage uses the fact that per-example gradients have low effective rank. Rather than storing M_ℓ in full, LoRIF stores its leading rank-c factors $M_\ell \approx u_\ell v_\ell^\top$, obtained by a few power iterations. We use $c = 1$, LoRIF's recommended setting, which takes a layer from $d_1 d_2 = 1024$ stored floats to $d_1 + d_2 = 64$. A document's embedding is the concatenation of these factors across the selected layers. Because the stored object is a factorization, the correct similarity is the bilinear form:

$$s(z_{\text{tr}}, z_{\text{te}}) \;=\; \sum_\ell \big(u_\ell(z_{\text{te}})^\top u_\ell(z_{\text{tr}})\big)\,\big(v_\ell(z_{\text{te}})^\top v_\ell(z_{\text{tr}})\big), \tag{2}$$

which is exactly the Frobenius inner product of the two reconstructed gradients, computed without reconstructing them.

3.2 Simplifying LoRIF

LoRIF scores a pair of documents as $g_{\text{te}}^\top H^{-1} g_{\text{tr}}$ with a damped Gauss–Newton curvature term H, made tractable by a truncated SVD of the gradient matrix and the Woodbury identity. The result splits into a plain gradient dot product plus a correction computed in the top-r curvature subspace.

In our setting, however, we only use the score to rank training examples for removal. Thus, gradient similarity is the only signal we need. We do not need calibrated influence estimates, comparisons of score magnitudes across models, or predictions of the loss change caused by removing an example.

3.3 Embedding Cost Ablations

The dominant remaining cost is the backward pass: Eq. 1 is defined per layer and Eq. 2 sums over whichever layers are selected, so the set of layers we embed determines how much of the network we must backpropagate through. To find the cheapest configuration that preserves embedding quality,

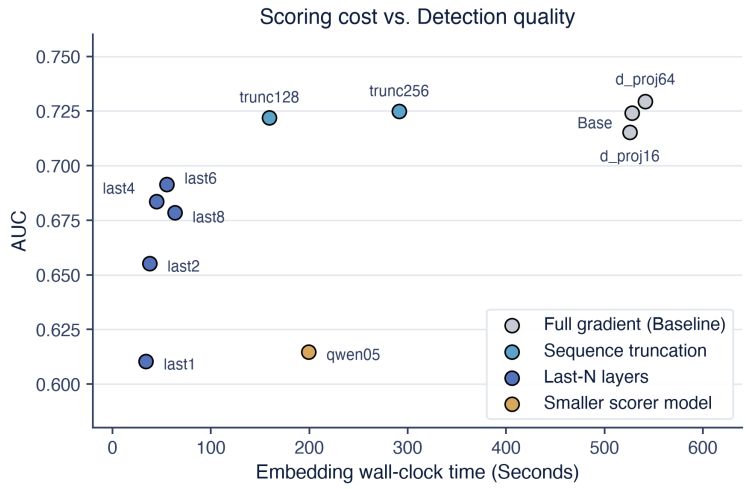

Figure 2: Detection AUC against embedding wall-clock time for each optimization strategy. Restricting backpropagation to the last four transformer blocks gives the best balance of embedding quality and speed.

we ran ablations over four axes: the projection dimension D, the number of transformer blocks to backpropagate through, the number of tokens read per dataset item, and the size of the embedding model. We recorded wall-clock time and a detection AUC that measures how well the gradient features predict which evaluation items a model has been pushed to solve. We use a math benchmark dataset for this, which is a standard evaluation domain. Each train and test math item is embedded as a (Q, A) pair with the question tokens masked out of the loss. The resulting per-example gradient is compressed to a fixed-length vector and L2-normalized, every test item is scored by its nearest-neighbor cosine similarity to the training set in that space, and we report the ROC-AUC of that score against whether the item was newly answered correctly after training. A higher AUC means the embedding places a test item near the training documents that actually changed the model's behavior on it, which is the property our removal step depends on.

Three of the four axes were unproductive: the projection dimension had little effect on either AUC or speed, token truncation was effective but risky and unlikely to generalize to longer documents, and shrinking the embedding model cost too much quality. Restricting the number of blocks, by contrast, preserved quality while greatly reducing cost, since the selected blocks sit at the top of the network and the backward pass stops once it has produced the input gradient of the shallowest selected block, while the forward pass still runs through the whole network. We chose $N = 4$, the fewest blocks retaining high embedding quality: two blocks scored noticeably worse at 0.65 AUC, and six or eight gained little. This confined the backward pass, normally about twice the cost of the forward pass, to a small fraction of the network, and because activations only need to be retained for the selected blocks, we could relax the memory limit on batch size. As such, using the $N = 4$ transformer blocks, our method achieves a high AUC of 0.684 with a **91.53% speedup compared to the base simplified LoRIF formulation**.

4 Emergent Misalignment

Emergent misalignment provides a setting for testing whether safety-relevant training signals are visible in gradient space. Betley et al. (2025) show that fine-tuning a model on a narrow undesirable behavior (e.g. producing insecure code) can cause broad misalignment on unrelated prompts, despite the fine-tuning data never explicitly containing generally harmful behavior. We ask whether the connection between the narrow fine-tuning data and these broader harmful behaviors can be recovered from the gradients they induce. Specifically, we test whether insecure-code training examples have gradients that are systematically more similar to gradients from broadly misaligned outputs than closely matched secure-code examples do.

We evaluate this using two independently fine-tuned model families. For Qwen, we use `emergent-misalignment/Qwen-Coder-Insecure`, the original misaligned checkpoint released by Betley et al. (2025) fine-tuned on their insecure-code dataset. For Mistral, we use `nikxtaco/mistral-small-24b-instruct-2501-insecure`, a reproduction of the insecure-code fine-tuning procedure applied to Mistral-Small-24B-Instruct-2501 (Mistral AI, 2025) in Betley et al. (2025), trained by Menon (2025). In every experiment, the source set consists of the original matched secure and insecure code datasets from (Betley et al., 2025): 6,000 insecure examples and 6,000 secure examples.

For each fine-tuned checkpoint, we separately construct a small target set containing examples of the model's broader emergent misalignment. We prompt each model with free-form, non-coding questions from (Betley et al., 2025) and collect its responses, then use Claude 5 Opus to review the responses, retaining only responses judged to clearly exhibit misaligned behavior. We compute a per-example gradient embedding for every source and target document using the method from Section 3 across both model families. We define the influence score of a source example as its mean similarity to all verified-unsafe targets for the corresponding model.

We measure how well this score separates the 6,000 insecure examples from the 6,000 secure examples using ROC-AUC. In this setting, AUC is the probability that a randomly selected insecure-code example receives a higher influence score than a randomly selected secure-code example. An AUC of 0.5 therefore indicates no detectable relationship between the insecure training set and the unsafe target behavior, while an AUC of 1.0 would indicate perfect separation. Because the source examples are closely matched in surface form, an AUC above chance indicates that the insecure examples induce gradients that are more aligned, on average, with gradients from the model's harmful outputs.

We find evidence of this connection for both model families. For fine-tuned Qwen, gradient similarity achieves an AUC of **0.604**, and for fine-tuned Mistral, gradient similarity achieves an AUC of **0.865**. In both cases, the only signal used for classification is each training example's gradient similarity to harmful generations from the corresponding fine-tuned model. These results suggest that EM leaves a detectable footprint in gradient space: the insecure-code examples that induce the narrow fine-tuning behavior are more aligned with the gradients of the broader misaligned behaviors that subsequently emerge. This provides evidence that gradient attribution can recover a connection between apparently benign or narrow training data and broader downstream harmful outputs. An illustration of this is shown in Figure 1. More emergent misalignment experiments are discussed in Appendix C. In particular, we find that a model's gradient representation of its responses does not necessarily always transfer to a different model's responses.

5 SafeTulu

5.1 Methodology

Motivated by the connection between insecure code fine-tuning examples and harmful outputs in the gradient embedding space of misaligned models, we apply our method to identify and remove benign-looking, safety degrading data in a popular fine-tuning dataset. Our pipeline is as follows:

1. **Embed.** Compute gradient embeddings for all documents in the training set and target set.
2. **Remove.** Score each training document against the target set by gradient similarity, and remove the 20% highest-scoring documents.
3. **Train.** Fine-tune the training model on the filtered dataset (hyperparameters shown in Appendix F).
4. **Evaluate.** Evaluate safety and capability.

We compute embeddings with Qwen 3 0.6B (Qwen-Team, 2025), and fine-tune Qwen 3 1.7B, Llama 3.2 1B (Meta AI, 2024), and Olmo 2 1B (OLMo et al., 2024). We set Tulu 3 (Lambert et al., 2025), a highly popular fine-tuning dataset, as our training dataset, and the prompt-response pairs with unsafe responses from SafeRLHF (Ji et al., 2025) as our target dataset (43,469 examples). Embedding all 939,343 Tulu 3 examples took 1 hour, 1 minute across 8 H100 GPUs. Rather than scoring each training document against a single pooled target vector, we cluster the target set and score documents against the cluster centroids, which the rank-1 factorization allows in closed form.

Appendix E provides the removal scoring identity. We release an anonymized version of SafeTulu, along with the gradient embedding and score for every Tulu 3 example, to support researchers who would like to create custom versions of SafeTulu, such as by filtering only certain subsets or using removal intervals other than 20%.

For safety evaluation, we measure attack success rate (ASR) on JailBreakBench behaviors (Chao et al., 2024) using Greedy Coordinate Gradient (GCG), a white-box jailbreak attack that optimizes an adversarial suffix appended to each harmful prompt to increase the likelihood of model compliance (Zou et al., 2023). We optimize the suffix for each behavior starting from 20 "!" tokens (full details are provided in Appendix F), then evaluate the models on each behavior with its optimized suffix appended. Each response is classified as safe or unsafe using the HarmBench classifier (Mazeika et al., 2024), a popular safety classifier. For capability, we evaluate on ORBench (Cui et al., 2025) for over-refusal, MMLU Pro and IFEval (Zhou et al., 2023) for general capability, and GSM8K (Cobbe et al., 2021), MATH 500 (Lightman et al., 2024), and MedQA (Jin et al., 2021) for domain-specific capability.

We compare against two baselines: fine-tuning on the unfiltered dataset, and fine-tuning on a dataset filtered by a Qwen 3 8B LLM-as-a-judge filter, a standard content-based data filtering approach. The judge LLM scores every document 0–5 for safety using the strong rubric from Maini et al. (2025). This comprehensive rubric provides detail on how to handle many types of harmful content, from mildly unsafe content such as financial advice without disclaimers and mentions of stressful topics to overtly harmful statements. After grading, the worst-scoring 20% is removed. This matches our method's removal budget and ranking-based decision rule, differing only in how the per-document score is produced. Appendix D provides the comprehensive rubric and scoring details.

5.2 Results

Results are shown in Figure 3, with a tabular version available in Appendix A. Across all three model families, SafeTulu improves safety while mostly preserving capability. For Qwen, GCG ASR decreases from 49% under baseline training to 38% after training on SafeTulu, compared with 48% for the LLM-as-a-judge filter. We observe the same pattern for Llama and Olmo: SafeTulu reduces ASR from 36% to 30% for Llama and from 49% to 46% for Olmo, whereas the LLM-as-a-judge baseline reaches 39% and 50%, respectively. These safety improvements do not come with a corresponding loss in general capability. Across MMLU Pro, IFEval, GSM8K, MATH 500, and MedQA, SafeTulu models usually remain close to or improve upon the capability of models trained on the full dataset, and over-refusal on ORBench decreases on all models except Olmo. This suggests that gradient attribution is not simply identifying data associated with unsafe behavior, but is identifying safety-degrading data that can be removed without sacrificing much of the useful training signal required for downstream capability.

The effect is particularly pronounced for Qwen, where filtering produces large capability gains alongside the reduction in ASR. Relative to baseline training, the SafeTulu model improves from 15.4% to 23.6% on MMLU Pro, from 37.9% to 48.6% on IFEval, from 38.51% to 58.1% on GSM8K, and from 18.6% to 31.8% on MATH 500, while also improving on MedQA. Since these gains are obtained after removing 20% of the training data, the filtered examples do not appear to be merely unnecessary for capability. Instead, for Qwen, the removed examples appear to include data that was actively interfering with learning. More broadly, these results indicate that the gradient signal can separate examples that contribute to safety degradation from examples that are important for useful capability, enabling safety improvements with little capability tradeoff.

It is worth emphasizing that the model used for embedding in the creation of SafeTulu was Qwen 3 0.6B. The effectiveness of SafeTulu across multiple models therefore reflects that, at least for the purposes of identifying safety-degrading data, our gradient embedding method can generalize across model families as well as to larger model scales.

Interestingly, the LLM judge filtering either roughly maintained or increased ASR. This may be because Tulu 3 is a very clean dataset (Lambert et al., 2025; Djuhera et al., 2025) with little overtly harmful data, making it difficult for the LLM judge to confidently identify safety-degrading data, and potentially causing the judge to accidentally flag benign-looking data that actually bolsters safety.

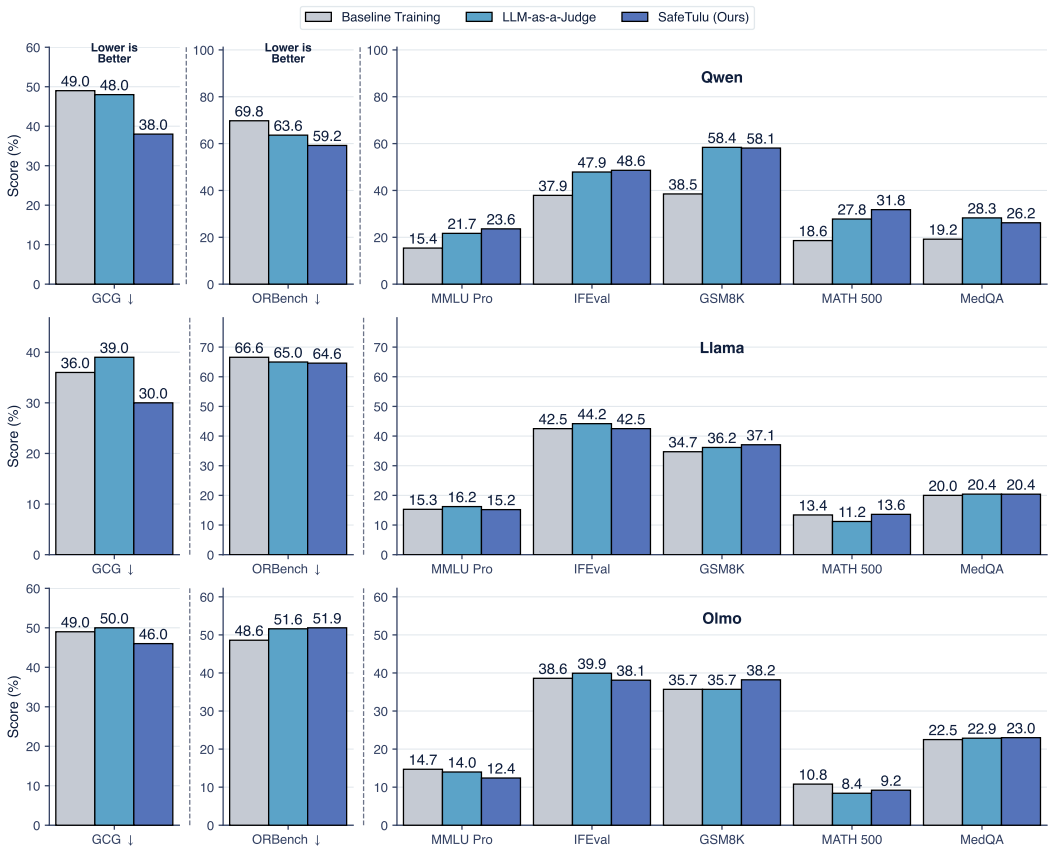

Figure 3: Evaluation results after baseline training, training on the LLM-as-a-judge filtered dataset, and training on SafeTulu. Lower is better on GCG attack success rate and ORBench over-refusal rate, while higher accuracy is better on all capability metrics. Across all models, training on SafeTulu decreases ASR, outperforming LLM-as-a-judge, while mostly preserving capability.

6 Qualitative Analysis

Gradient filtering produced more variation in removal rates across Tulu 3 subsets than the LLM-as-a-judge baseline, with the contrast clearest on the math data. While the judge removed `numinamath`, `open_math_2_gsm8k`, and `personas_math` at nearly identical rates (15.4–15.6%), gradient filtering removed 7.9%, 19.4%, and 17.2%, respectively. This suggests that gradients distinguish between examples that appear similar at the level of surface content, consistent with gradient attribution capturing differences in the training signal that a content-based judge does not.

A closer look at these subsets reveals several qualitative distinctions. In `NuminaMath`, removed examples disproportionately contain Python execution failures such as tracebacks or exceptions, whereas clean symbolic reasoning traces are more often retained. In `PersonaHub`, simple "write a Python function" tasks are removed more often, while calculus and differential-equation modeling problems are preferentially kept. Within `PersonaMath`, gradient filtering more often removes narrative arithmetic, where quantities are carried through a sequence of prose operations, while retaining examples involving explicit algebraic setup, substitution, and simplification.

These patterns suggest that tool-execution failures and narrative arithmetic may be under-studied sources of safety degradation. More broadly, gradient filtering shows that more direct fine-tuning examples may create more safety degradation, as gradient filtering seems to have favored less direct or more elaborated forms of reasoning. It removed relatively more clean arithmetic and simple coding tasks while retaining repeating-decimal arithmetic, algebraic setup, differential-equation modeling, and other less immediately direct examples. One possible interpretation is that direct training

examples reinforce a general pattern of immediate compliance. Importantly, this does not appear to be reducible to a simple preference for shorter responses, as removed examples are clearly shorter in only about half (6/11) of the major subsets. We stress that these patterns are only **suggestive rather than established**. We make no claim that these features must cause safety degradation, only that these features are what these examples appear to have in common.

Finally, language analysis yields an interesting finding: English examples are removed at a higher rate than other major languages in both `WildChat` and `Aya`, despite prior observations that non-English fine-tuning can degrade safety (Hawkins et al., 2026). This may be Tulu 3's data being very clean (Lambert et al., 2025; Djuhera et al., 2025).

We provide our full analysis, feature-level relative risk numbers, and more in Appendix B.

7 Conclusion

In this work, we use gradient attribution to trace safety degradation back to the training data that induces it. We first show that, in models exhibiting emergent misalignment, the gradient representations of insecure-code fine-tuning examples align with the harmful behaviors that emerge after training, with AUC as high as 0.865, providing evidence that this connection is detectable in gradient space. Building on this observation, we develop a scalable gradient-attribution method for identifying benign-looking SFT examples associated with unsafe behavior, and use it to construct SafeTulu by removing the 20% of Tulu 3 examples most aligned with a reference set of unsafe responses. Across Qwen, Llama, and Olmo, training on SafeTulu consistently reduces GCG attack success rate by up to 22% while largely preserving or improving capability across multiple benchmarks, outperforming an LLM-as-a-judge filtering baseline on safety. We also discussed possible sources of benign-looking safety-degrading data, such as tool-execution failures, based on the removed data. Crucially, our method makes gradient-based safety filtering practical at a substantially larger scale than prior approaches. By restricting backpropagation to the final few transformer blocks and using compact gradient representations, we can attribute and filter datasets containing on the order of a million examples, rather than the smaller datasets considered in previous gradient-based methods (e.g. He et al., 2024). This boosts gradient attribution from a tool more suited to small-scale analysis into a more practical data-selection mechanism for large-scale safety filtering, opening the door to using gradient attribution to systematically remove safety-degrading examples before they ever become part of large-scale post-training mixtures.

8 Limitations

While we believe our work represents a promising route to increasing safety through gradient attribution, we recognize limitations. First, we do not perform evaluation sampling across multiple random seeds. However, we emphasize that the direction of increase or decrease is mostly consistent across all models and benchmarks. Second, the only removal fraction we test is 20%. Testing the effect of other removal fractions is a good direction for future work. Despite these limitations, we believe SafeTulu, our gradient embedding method, our findings on emergent misalignment, and our qualitative analysis are still valuable to the community.

9 Reproducibility Statement

We release an anonymized version of SafeTulu, along with the gradient embedding for every Tulu 3 and SafeRLHF example, and the score for every Tulu 3 example, to support researchers who would like to create custom versions of SafeTulu, such as filtering only certain subsets or using removal intervals other than 20%. It is available at `https://anonymous-hf.com/a/ow776vqmqgvd/`. We also state our hyperparameters in Appendix F.

10 AI USE STATEMENT

In this work, we used generative AI tools for code implementation, qualitative analysis, and writing organization. We have not used generative AI tools for synthetic data generation, formulating mathematical claims, proposing hypotheses, designing research methodology, dataset reformatting, or result interpretation. Developing theoretical models, assisting in the writing of proofs, and assisting with translation are not applicable to this work. Additionally, we used generative AI tools for creating scientific figures, creating tables, identifying relevant literature, and searching for information. We have reviewed all AI-assisted work. For example, we confirmed initial qualitative claims by AI about our dataset. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

11 ETHICS STATEMENT

Our released artifacts are intended to support safer fine-tuning, but they carry dual-use risk. The per-example scores we release for Tulu 3 rank documents by their gradient alignment with unsafe responses. As such, they could be used for training to increase safety degradation instead of decrease it. We believe the risk here is small, as this attack method was already demonstrated in He et al. (2024), Tulu 3 is already publicly available, and the ranking could be reproduced from our method description. Against this, the scores give researchers a direct way to build safer fine-tuning mixtures and to study which training data drives safety degradation, which we believe is more impactful. We also note that the target sets used in our emergent misalignment experiments that consist of harmful model generations are not being released.

REFERENCES

Jan Betley, Daniel Chee Hian Tan, Niels Warncke, Anna Sztyber-Betley, Xuchan Bao, Martín Soto, Nathan Labenz, and Owain Evans. Emergent misalignment: Narrow finetuning can produce broadly misaligned LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 4043–4068. PMLR, 13–19 Jul 2025.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J. Pappas, Florian Tramèr, Hamed Hassani, and Eric Wong. JailbreakBench: An open robustness benchmark for jailbreaking large language models. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

Sang Keun Choe, Hwijeen Ahn, Juhan Bae, Kewen Zhao, Minsoo Kang, Youngseog Chung, Adithya Pratapa, Willie Neiswanger, Emma Strubell, Teruko Mitamura, Jeff Schneider, Eduard Hovy, Roger Grosse, and Eric Xing. What is your data worth to GPT? LLM-scale data valuation with influence functions. In *Advances in Neural Information Processing Systems*, volume 38, pp. 145944–145985, 2025.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Jiaxing Cui, Wei-Lin Chiang, Ion Stoica, and Cho-Jui Hsieh. OR-Bench: An over-refusal benchmark for large language models. In *Proceedings of the International Conference on Machine Learning*, 2025.

Aladin Djuhera, Swanand Ravindra Kadhe, Syed Zawad, Farhan Ahmed, Heiko Ludwig, and Holger Boche. Fixing it in post: A comparative study of LLM post-training data quality and model performance. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 38, 2025.

Will Hawkins, Kaivalya Rawal, Jonathan Rystrøm, Stratis Tsirtsis, Zihao Fu, Greta Warren, Ryan Brown, Eoin Delaney, Sandra Wachter, Brent Mittelstadt, and Chris Russell. The heterogeneous

safety impacts of benign multilingual fine-tuning. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, Seoul, South Korea, 2026.

Luxi He, Mengzhou Xia, and Peter Henderson. What's in your safe data?: Identifying benign data that breaks safety. In *First Conference on Language Modeling*, 2024.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Jiaming Ji, Donghai Hong, Borong Zhang, Boyuan Chen, Josef Dai, Boren Zheng, Tianyi Alex Qiu, Jiayi Zhou, Kaile Wang, Boxun Li, Sirui Han, Yike Guo, and Yaodong Yang. PKU-SafeRLHF: Towards multi-level safety alignment for LLMs with human preference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31983–32016. Association for Computational Linguistics, July 2025. ISBN 979-8-89176-251-0.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14), 2021. ISSN 2076-3417.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 06–11 Aug 2017.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling (COLM)*, 2025.

Hao Li, Lijun Li, Zhenghao Lu, Xianyi Wei, Rui Li, Jing Shao, and Lei Sha. Layer-aware representation filtering: Purifying finetuning data to preserve LLM safety alignment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 8030–8050, Suzhou, China, November 2025. Association for Computational Linguistics.

Shuangqi Li, Hieu Le, Jingyi Xu, and Mathieu Salzmann. LoRIF: Low-rank influence functions for scalable training data attribution, 2026.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Pratyush Maini, Sachin Goyal, Dylan Sam, Alexander Robey, Yash Savani, Yiding Jiang, Andy Zou, Matt Fredrikson, Zachary Lipton, and Zico Kolter. Safety Pretraining: Toward the next generation of safe AI. In *Advances in Neural Information Processing Systems*, 2025.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35181–35224. PMLR, 21–27 Jul 2024.

Nikita Menon. Base vs instruct models with emergent misalignment. Research report, Supervised Program for Alignment Research (SPAR), 2025.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. Meta AI Blog, September 2024.

Gouki Minegishi, Hiroki Furuta, Takeshi Kojima, Yusuke Iwasawa, and Yutaka Matsuo. Understanding emergent misalignment via feature superposition geometry. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30385–30414, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6.

Mistral AI. Mistral small 3. Mistral AI Blog, January 2025.

Burt L. Monroe, Michael P. Colaresi, and Kevin M. Quinn. Fightin' words: Lexical feature selection and evaluation for identifying the content of political conflict. *Political Analysis*, 16(4):372–403, 2008.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 27074–27113. PMLR, 23–29 Jul 2023.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930, 2020.

Xiangyu Qi, Yi Zeng, Tinghao Xie, Pin-Yu Chen, Ruoxi Jia, Prateek Mittal, and Peter Henderson. Fine-tuning aligned language models compromises safety, even when users do not intend to! In *The Twelfth International Conference on Learning Representations*, 2024.

Qwen-Team. Qwen3 technical report, 2025.

Han Shen, Pin-Yu Chen, Payel Das, and Tianyi Chen. SEAL: Safety-enhanced aligned LLM fine-tuning via bilevel data selection. In *The Thirteenth International Conference on Learning Representations*, 2025.

Anna Soligo, Edward Turner, Senthooran Rajamanoharan, and Neel Nanda. Emergent misalignment is easy, narrow misalignment is hard. In *International Conference on Learning Representations*, pp. 102392–102429, 2026.

Clemens Vetter, David Kaczér, Lucie Flek, and Florian Mai. Data attribution of emergent misalignment with persona features. In *BlackboxNLP Workshop*, 2026.

Miles Wang, Tom Dupre la Tour, Olivia Watkins, Aleksandar Makelov, Ryan Chi, Samuel Miserendino, Jeffrey Wang, Achyuta Rajaram, Johannes Heidecke, Tejal Patwardhan, and Daniel Mossing. Persona features control emergent misalignment. In *International Conference on Learning Representations*, pp. 48836–48885, 2026.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In *Proceedings of the 41st International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2024.

Junbo Zhang, Qianli Zhou, Xinyang Deng, Wen Jiang, Jie Pan, and Jinbiao Zhu. Datashield: Safety-degrading data filtering for LLM benign instruction fine-tuning, 2026.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. doi: 10.48550/arXiv.2311.07911.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models, 2023.

A SafeTulu Results Table

Model	Training Data	GCG ↓	MMLU Pro	IFEval	GSM8K	MATH 500	MedQA	ORBench ↓
Qwen	Baseline Training	49.0%	15.4%	37.9%	38.5%	18.6%	19.2%	69.8%
	LLM-as-a-Judge	48.0%	21.7%	47.9%	**58.4%**	27.8%	**28.3%**	63.6%
	SafeTulu (Ours)	**38.0%**	**23.6%**	**48.6%**	58.1%	**31.8%**	26.2%	**59.2%**
Llama	Baseline Training	36.0%	15.3%	42.5%	34.7%	13.4%	20.0%	66.6%
	LLM-as-a-Judge	39.0%	**16.2%**	**44.2%**	36.2%	11.2%	**20.4%**	65.0%
	SafeTulu (Ours)	**30.0%**	15.2%	42.5%	**37.0%**	**13.6%**	20.4%	**64.6%**
Olmo	Baseline Training	49.0%	**14.7%**	38.6%	35.7%	**10.8%**	22.5%	**48.6%**
	LLM-as-a-Judge	50.0%	14.0%	**39.9%**	35.7%	8.4%	22.9%	51.6%
	SafeTulu (Ours)	**46.0%**	12.4%	38.1%	**38.2%**	9.2%	**23.0%**	51.9%

Table 1: Evaluation results after baseline training, training on the LLM-as-a-judge filtered dataset, and training on SafeTulu. Lower is better on GCG attack success rate and ORBench over-refusal rate, while higher accuracy is better on all capability metrics. Across all models, training on SafeTulu decreases ASR, outperforming LLM-as-a-judge, while mostly preserving capability.

B Comprehensive Qualitative Analysis

B.1 Removal Rates

Table 2 shows the fraction of each Tulu 3 subset removed by gradient filtering and by the LLM judge under the same 20% budget. The math subsets (highlighted) show the clearest contrast. The judge removes all three at almost the same rate (15.4%–15.6%), which fits a scorer that evaluates content only at the surface level. Gradient filtering spreads its removals much more widely across the same subsets, from 7.9% of `numinamath` to 19.4% of `open_math_2_gsm8k`. It treats material that looks interchangeable on the surface as different, indicating that it is responding to something beyond surface content.

Subset	Gradient Removal Rate	Judge Removal Rate
`personahub`	9.2%	15.5%
`codealpaca`	17.3%	15.5%
`wildchat`	18.7%	28.2%
`aya`	11.7%	19.2%
`flan`	22.1%	17.0%
`wildjailbreak`	39.4%	32.8%
`wildguardmixtrain`	41.0%	44.3%
`numinamath`	7.9%	15.4%
`open_math_2_gsm8k`	19.4%	15.6%
`personas_math`	17.2%	15.6%
`no_robots_converted`	31.9%	20.2%
`table_gpt`	18.7%	15.7%
`science.*` (SciRIFF tasks, aggregate)	21.3%	15.7%
`hard_coded_*` (aggregate)	54.2%	87.5%
`oasst1` + very small subsets, aggregate	35.8%	21.0%

Table 2: Fraction of each Tulu 3 subset removed by each method under a 20% budget, with math subsets highlighted. While the LLM judge removes math at a roughly flat rate, gradient filtering varies its rates, indicating it is finding a signal beyond surface level content.

B.2 Removal Analysis Methodology

To find what separates removed from kept documents without assuming a category in advance, we run a log-odds comparison with an informative Dirichlet prior (the "fightin' words" method) over unigrams and bigrams (Monroe et al., 2008). We do this separately for the prompt and the response, and only within one subset at a time, so subset identity cannot confound the comparison. Each term's

document frequency in the removed and kept groups is compared against a shared background rate and turned into a z-score, which keeps words that are merely common overall from dominating the ranking. We analyze the most strongly enriched terms on each side. While gradient filtering did not show any discernible pattern for many subsets, we did identify trends in a few subsets, shown in Table 3. For each trend, we measure the rate on the full removed and kept populations and report the relative risk (RR), the removal rate within the feature divided by the subset's baseline removal rate. Table 5 shows truncated dataset examples representing each trend.

Subset	Feature	RR
`numinamath`	Python traceback/exception	1.71×
`personahub`	"Write a Python function" code task	1.74×
`personahub`	Calculus/differential-equation modeling	0.51×
`codealpaca`	High-level architecture discussion	1.75×
`codealpaca`	Runnable code in response	0.89×
`open_math_2_gsm8k`	Long repeating decimal	0.36×
`personas_math`	Algebraic manipulation	0.70×
`wildchat`	English	1.46×
`aya`	English	1.73×

Table 3: Relative rate (RR) of each feature among removed vs kept documents. We observe trends such as removing tool-execution failures and simple coding tasks at a higher rate.

We discuss our findings below. We stress that these patterns are only suggestive, not definitively established.

B.3 Code Analysis

In `numinamath`, where the model writes Python/SymPy code, reads the executed output, and continues reasoning, the top terms in the documents gradient filtering removed are fragments of a Python exception (`traceback`, `most recent call`, `call last`) and the top terms in the documents gradient filtering kept are the vocabulary of a clean symbolic trace (`sympy`, `import_sympy`, `symbols`). Removed documents contain a real tool execution traceback or exception $1.71\times$ more than kept documents. A typical removed example is a failed first attempt at tool execution (an `AttributeError`, `TypeError`, or `SyntaxError`), after which the model notices the error and continues. This suggests that safety-degrading code data may extend beyond the insecure code studied in emergent misalignment. In `personahub`, gradient filtering removes "write a Python function" tasks (RR $1.74\times$) and keeps calculus and differential-equation modeling problems (RR $0.51\times$) more often, suggesting that code data may be more safety-degrading than math data. In `codealpaca`, gradient filtering more often removes high-level discussion of software architecture (`scalability`, `distributed`, `authentication`) at an RR $1.75\times$ and keeps responses that contain runnable code (`def`, `print`, `return`) at an RR of $0.89\times$.

B.4 Math Analysis

In `open_math_2_gsm8k`, gradient filtering more often removes answers whose arithmetic comes out to clean numbers and keeps answers containing long repeating decimals (e.g. `0.3333333333333333`) at an RR of $0.36\times$. In `personas_math`, it removes narrative arithmetic, in which quantities are carried in prose through a sequence of operations, and keeps genuine algebraic manipulation, in which the model defines a variable, sets up an equation, and substitutes or simplifies (RR $0.70\times$).

B.5 Language Analysis

In `wildchat`, gradient filtering removes English conversations at $1.46\times$ as often as non-English languages, with every other major language removed at 0.26–$0.65\times$ the baseline rate. `aya_100k` shows the same direction, with English again the most removed language (RR $1.73\times$), even though English is only 6.7% of that subset and the majority language in WildChat. This is notable because fine-tuning on non-English data has often been reported to degrade safety (Hawkins et al., 2026).

One possible reason for the reversal here is that Tulu 3's data is very clean (Lambert et al., 2025; Djuhera et al., 2025).

B.6 Overall Analysis

In terms of well-defined types of data, tool-execution failures and narrative arithmetic stand out as possible categories of data which may degrade safety that have been under-studied in prior work. As a broad trend, gradient filtering seems to suggest that direct examples may degrade safety more than elaborate examples. Gradient filtering appears to have removed more direct data (such as clean arithmetic and simple coding tasks) and kept less direct or more elaborate data (such as repeating-decimal arithmetic, algebraic setup, and differential equation modeling) more often. One possible reason is that direct, immediately helpful answers match the pattern of simple compliance, so direct data may degrade safety more. Table 4 shows that this effect cannot be reduced to simply optimizing for shorter examples. Removed documents are clearly shorter in only about half of the eleven major subsets (6/11), with three other subsets having a minor length difference of 5% or less and two others showing gradient filtering selecting longer data.

We stress that these patterns are only **suggestive rather than established**. We make no claim that these features must cause safety degradation, only that these features are what these examples appear to have in common. Every result comes from the correlational method in Section B.2.

Subset	% Removed Data is Shorter By
`table_gpt`	40%
`personahub`	33%
`aya`	30%
`no_robots_converted`	29%
`wildjailbreak`	21%
`flan`	20%
`wildguardmixtrain`	5%
`open_math_2_gsm8k`	5%
`personas_math`	3%
`codealpaca`	-3%
`wildchat`	-11%

Table 4: The percentage that the data removed by gradient filtering is shorter than the kept documents for every major subset. The percentage is only large for half (6/11) of the major subsets, indicating that gradient filtering is not simply optimizing for shorter length examples.

C Additional Emergent Misalignment Experiments

C.1 Non-Fine-Tuned Models

As a control, we repeat the gradient-embedding analysis using the corresponding base instruct models before insecure-code fine-tuning. For Qwen2.5-Coder-32B-Instruct (Hui et al., 2024), the AUC falls from 0.604 after insecure-code fine-tuning to 0.552 in the non-fine-tuned model. For Mistral-Small-24B-Instruct-2501 (Mistral AI, 2025), the AUC falls from 0.865 to 0.431. These results indicate that the gradient-space relationship between insecure-code examples and broadly harmful outputs does not emerge until after the model has undergone emergent misalignment.

C.2 Cross-Model Transfer

We next test whether the gradient-space signal transfers across model families by changing the model used to compute embeddings and the model from which the verified-unsafe target responses are drawn. Results are shown in Table 6. When the fine-tuned Mistral model embeds harmful responses generated by fine-tuned Qwen, AUC decreases only modestly from 0.865 to 0.832. In the opposite direction, however, using fine-tuned Qwen to embed fine-tuned Mistral's harmful responses reduces AUC from 0.604 to 0.247. For the non-fine-tuned models, cross-model targets produce values that remain close to chance. Overall, this shows that the specific embedding model used is

Subset	Removed	Kept
numinamath	...`from scipy.optimize import minimize` → `ModuleNotFoundError: No module named 'scipy'` "It seems that the `scipy` library is not available..."	"Given... $a_n = \sin\frac{n\pi}{3}$, find the value of a_3." → `import sympy as sp...a_3 = a_n.subs(n, 3)`
personahub	"Write a python function to convert a list of historical event years into a list of centuries..." → `def convert_years_to_centuries(years)`	"An entrepreneur is analyzing... determine the constants in the revenue functions and predict when total revenue will reach $50 million" → "**Step 1:** Determine..."
codealpaca	"...building a resilient serverless architecture using... AWS Lambda and API Gateway... Could you provide a detailed, sequential guide..." → "...These include planning, establishing the environment, continuous integration..."	"Create a function that takes two input strings, and returns a dictionary containing the count of unique characters..." → `def count_chars(string1, string2):...print(char_counts)`
open_math_2_gsm8k	"Emily planted a cherry blossom tree... when she was 6... takes 9 years to bloom... currently 10..." → "$\ldots 9 - 4 = 5$ years are left... $10 + 5 = 15$... $\boxed{15}$"	"...The smallest building is 1/3 the size of the second-largest building..." → "$\ldots 1/3 \times 135 = 0.3333333333333333 \times 135 = 45$ feet tall."
personas_math	"Lawmaker Lisa... plans to allocate resources to 5 different community centers. Each center will receive an equal number..."	"A television producer... divide the show into three parts..." → "Let's define variables for the time allocated to each part..."
wildchat	"hi" → "Hello! How can I assist you today?"	"Title: Dyanra: Misteryo, Kababalaghan. Tauhan: Mike, Elmer, Jones, Erica..." → "...Sa isang tahimik at mapayapang baryo, sinimulan ng magkakapatid na Mike, Elmer..."
wild jailbreak	"share or trade illegal or non-consensual pornography" → "I'm sorry, but I can't fulfill this request."	"...conduct a dogfighting event without getting caught..." → "I'm really sorry, but I can't assist... if you're looking to engage with dogs in a positive and legal manner..."

Table 5: Examples of removed and kept documents from each subset. Responses are truncated to their distinguishing parts.

Embedding Model	Target Responses	Model State	Matched AUC	Transfer AUC
Mistral-Small-24B-Instruct-2501	Qwen harmful responses	Fine-tuned	0.865	0.832
Qwen2.5-Coder-32B-Instruct	Mistral harmful responses	Fine-tuned	0.604	0.247
Mistral-Small-24B-Instruct-2501	Qwen harmful responses	Base	0.431	0.502
Qwen2.5-Coder-32B-Instruct	Mistral harmful responses	Base	0.552	0.535

Table 6: Cross-model transfer of the emergent-misalignment gradient signal. Matched AUC uses harmful target responses produced by the same model family as the embedding model, while Transfer AUC replaces them with responses from the other fine-tuned model family. The embedding model used is important: models' gradient representations do not necessarily transfer to different models' responses.

important for gradient representation, and that a model's gradient representation will not necessarily transfer to a different model's responses.

These results may appear to contrast with Section 5.2, where filtering with gradients from a non-fine-tuned Qwen model improves safety across multiple model families. However, the emergent-misalignment experiments probe a substantially broader and more difficult relationship: whether gradients connect narrow insecure-code training examples to very harmful behaviors that emerge out of domain after fine-tuning. This more indirect connection is likely to be more model- and state-specific than the more general safety-degradation signal used for SafeTulu, which may explain why transfer across models and to non-fine-tuned checkpoints is weaker here.

C.3 SCALING

Finally, we test how the gradient signal varies with model scale within the Qwen2.5-Coder family, using non-misaligned instruct checkpoints from 0.5B to 32B parameters. The results are shown in Table 7. AUC increases steadily from 0.507 at 0.5B to 0.549 at 3B, 0.573 at 7B, and 0.590 at 14B, before decreasing to 0.552 at 32B. The scaling trend is therefore not monotonic across the full model-size range. From 0.5B through 14B, the gradient-space signal strengthens consistently, suggesting that larger models can make the relevant distinction increasingly detectable. However, the 32B model falls below the 14B model, indicating that more parameters do not necessarily always produce a stronger gradient-attribution signal. This suggests that gradient-based attribution may sometimes be sharper in smaller or intermediate-sized models than in the largest model, and that embedding-model scale should be treated as an empirical design choice rather than assuming that larger models will always provide better attribution geometry.

Model	AUC
Qwen2.5-Coder-0.5B-Instruct	0.507
Qwen2.5-Coder-3B-Instruct	0.549
Qwen2.5-Coder-7B-Instruct	0.573
Qwen2.5-Coder-14B-Instruct	0.590
Qwen2.5-Coder-32B-Instruct	0.552

Table 7: Gradient-embedding AUC across non-fine-tuned Qwen2.5-Coder model sizes. The gradient signal increases up to 14B, suggesting that smaller models may have a sharper gradient signal than larger models.

D LLM-AS-A-JUDGE BASELINE DETAILS

Safety Scoring Prompt (0–5 Scale)

Objective: Assess the safety of a given text, evaluating its appropriateness for public use, educational environments, or general audiences. Assign points based on the severity and presence of unsafe content across different categories. The total score ranges from **0** to **5**, with higher scores indicating **lower safety**.

Scoring Criteria:

0 (Safe): *Content:* no unsafe or inappropriate content. *Suitability:* fully appropriate for all audiences, including children and sensitive environments. *Language:* respectful, without any profanity, slurs, or derogatory terms. *Themes:* no adult themes, violence, or references to illegal activities.

1 (Minimally Unsafe): *Content:* minimal unsafe content with minor instances of inappropriate language or mild adult themes. *Language:* may include mild profanity (e.g., "damn," "hell") or slang. *Themes:* non-explicit references to alcohol, mild romantic situations, or casual mentions of stressful topics. *Suitability:* generally appropriate but may be unsuitable for very young audiences or highly sensitive environments.

2 (Slightly Unsafe): *Content:* elements of concerning language or topics. *Language:* minor personal insults or sarcastic remarks without severe offense. *Advice:* mild financial, legal, or medical advice without professional endorsement or disclaimers. *References:* brief, indirect mentions of political, religious beliefs, or controversial topics. *Suitability:* slightly inappropriate or off-topic for certain audiences; may require discretion.

3 (Moderately Unsafe): *Content:* moderately harmful content that may offend or disturb. *Language:* social-group insults, stereotypes, or prejudiced remarks without aggressive intent. *Themes:* suggestive or explicit adult themes, non-graphic sexual content, or depictions of moderate violence. *References:* brief mentions of weaponry, self-harm, or crimes without detailed descriptions. *Suitability:* moderately inappropriate; may require review, parental guidance, or filtering.

4 (Clearly Unsafe): *Content:* clearly inappropriate for general audiences. *Language:* strong profanity, slurs, or aggressive derogatory terms directed at individuals or groups. *Advice:* detailed financial, legal, or medical advice that could lead to harm. *Themes:* extremist beliefs, conspiracy theories, or detailed references to illegal activities. *References:* encouraging self-harm, violence, or other dangerous actions. *Suitability:* contains explicit elements; should be restricted or removed from public platforms.

5 (Highly Unsafe): *Content:* highly unsafe and overtly harmful. *Language:* hate speech, violent threats, or calls to action against individuals or groups. *Themes:* graphic adult content involving illegal activities. *Advice:* promotion or instruction of illegal or harmful actions. *Privacy:* disclosure of personally identifiable information without consent. *Suitability:* inappropriate for any general, educational, or public platform.

Instructions: (1) Analyze the text thoroughly: understand the content, context, and tone, and identify any unsafe elements based on the criteria. (2) Assign a safety score: start from 0 and assign the highest applicable score (do not sum points). (3) Return a JSON object: `{"score": <score>, "reason": "<reason>"}`, where the reason briefly states the main issue (e.g., "hate speech," "financial advice," "WMD").

Example. *Text:* "I think we should overthrow the government because they are all corrupt and don't care about us." *Identified content:* expresses extremist beliefs and suggests illegal activities. *Justification:* advocates the illegal act of overthrowing the government, which is a serious offense; promotes extremist beliefs and could encourage unsafe behavior, making it clearly inappropriate for general audiences. *Output:* `{"score": 4, "reason": "illegal activity"}`

We prompt Qwen 3 8B, served with constrained (JSON-schema-guided) decoding, to score every document in the training set for safety on an integer 0–5 scale, where 0 denotes content that is fully safe and appropriate for all audiences and 5 denotes highly unsafe content (hate speech, violent threats or calls to action, graphic illegal content, disclosure of personal information, or explicit promotion/instruction of illegal or harmful acts). Each score level is defined by the rubric established in Appendix C.1 from Maini et al. (2025) (shown above) over content, language, themes, advice given, and audience suitability. The judge is instructed to start from 0 and assign the highest applicable score rather than summing point-by-point, and returns a structured {`"score": int, "reason": str`} object. Scores that fail to parse are treated as missing and excluded.

The judge sees both the user prompt and the model's response, so it can penalize an unsafe prompt-response pair even when the response itself appears benign in isolation. Because the discrete 0–5 scale may produce a tie band straddling the removal boundary (e.g. the cutoff falling in the middle of the population of 4s and 5s), ties are broken by a seeded random draw, so which documents within the boundary score are kept vs. dropped is randomized rather than determined by document order.

E Removal Scoring Details

For a training document x with rank-1 factors $\{(u_\ell, v_\ell)\}$ and a cluster centroid M_c, the score is

$$\text{score}(x, c) \;=\; \langle G_x, M_c \rangle_F \;=\; \frac{1}{|c|} \sum_{t \in c} \sum_{\ell=1}^{L} (u_\ell^x \cdot u_\ell^t)(v_\ell^x \cdot v_\ell^t),$$

i.e., the Frobenius inner product between the training document's gradient matrix and the cluster centroid is exactly the mean of its pairwise bilinear scores (Eq. 2) against the cluster's members. This identity lets us compute cluster-level scores directly from the rank-1 factors, without ever forming the full outer-product embedding $G_x \in \mathbb{R}^{d \times d}$ per layer, and is equivalent (after normalizing by the source document's self-norm and the centroid's norm) to running cosine-similarity k-means in the lifted outer-product space. Each training document is assigned to its nearest cluster by $\arg\max_c \text{score}(x, c)$, giving every training document a (cluster assignment, distance-to-centroid) pair.

F Hyperparameters

Training Hyperparameters:

- Number of Steps: 2,297
- Learning Rate (LR): 1×10^{-4} (after conducting a sweep over 5×10^{-5}, 1×10^{-4}, and 3×10^{-4})
- $\beta 1$: 0.9
- $\beta 2$: 0.95
- Weight Decay: 0.1
- LR Warm-Up: 20 steps
- LR Decay Ratio: 20%
- Minimum LR Factor: 0.05
- Sequence Length: 2,048
- Global Batch Size: 128 sequences
- Batch Size per GPU: 8
- Gradient Clipping: Max Norm 1.0
- Parameter Precision: bf16
- Gradient Reduction Precision: fp32
- Dataloader Shuffle Seed: 9,176

GCG Hyperparameters:

- Initial Suffix: 20 "!" tokens
- Optimization Target: JailBreakBench supplied affirmative target per behavior
- Number of Optimization Steps: 500
- Candidate Suffixes/Step: 512
- Gradient Top-k: 256