CacheMedic++: Contractive KV-Cache Stabilization for Robust Transformer Inference

Anonymous Authors

Abstract

Large language models rely on the key–value (KV) cache for efficient autoregressive inference, but this persistent internal state is vulnerable to corruption that can induce large output drift without changing prompts or weights. We propose CacheMedic++, a lightweight in-attention KV repair operator trained with frozen base model weights using self-distillation, clean-state identity regularization, and a contraction objective on corrupted states. On our canonical gpt2-medium setting, CacheMedic++ improves clean score from 0.2373 to 0.2466 and robustness AUC from 0.0390 (best heuristic) to 0.0401. In paired ablations, adding contraction improves robustness AUC from 0.0388 to 0.0401 and reduces logit sensitivity to 92.91% ($\delta = 1.0$) and 92.66% ($\delta = 2.0$) of the no-contraction variant. The same operator family transfers qualitatively to gpt2-large, improving clean score from 0.2974 to 0.3001 and robustness AUC from 0.0472 to 0.0475.

1 Introduction

The KV cache enables fast decoding by storing past keys and values, but it also creates a persistent internal state that can be corrupted or manipulated. This paper studies **KV-cache integrity** and proposes a lightweight stabilization operator.

Contributions.

- We introduce CacheMedic++, a learned KV-cache stability operator R_ϕ inserted before attention logits, with frozen base model weights.
- We train R_ϕ via self-distillation under a reproducible corruption family and add an explicit contraction penalty to encourage state stability.
- We treat stability metrics as primary results, reporting logit sensitivity curves and per-layer/head amplification maps, and we validate transfer qualitatively on a second model.

2 Threat and Fault Model

We define a corruption family C that perturbs the KV cache during inference. Corruption types are: Gaussian noise, dropout/zeroing, orthogonal rotations, sparse bitflip-like sign/jump perturbations, quantization noise, and contiguous overwrite of a cached segment. Corruptions are applied through deterministic masks over layer, head, and time axes.

For evaluation we use $\epsilon \in \{0.0, 0.08, 0.16\}$ and report robustness curves across this grid. Our out-of-distribution (OOD) protocol is leave-one-type-out (LOTO): the holdout type is contiguous overwrite, excluded from training corruption mixtures and used for OOD evaluation.

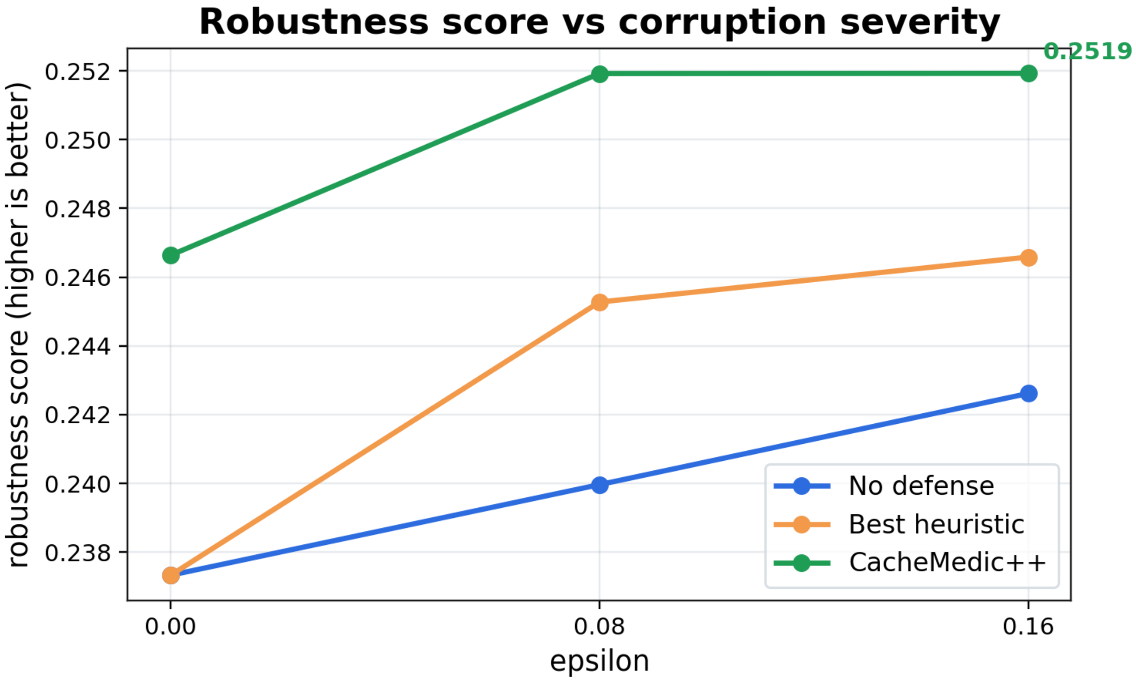

Figure 1: Robustness curves (task score vs corruption severity).

3 Method: CacheMedic++

CacheMedic++ inserts a small operator R_ϕ inside attention. After past KV concatenation, we optionally apply corruption and then repair:

$$(K, V) \xrightarrow{C} (K_{corr}, V_{corr}) \xrightarrow{R_\phi} (K_{hat}, V_{hat})$$

Attention logits are computed from q and K_{hat}. Base model weights are frozen; only ϕ is trained.

We implement three repair families (A/B/C) and use Option A by default. The tuned configuration in this paper is V-only repair with rank 4 over two protected layers.

Training objective. For each batch, we combine a distillation objective with clean-state identity and a contraction term on corrupted states:

$$\mathcal{L} = \mathcal{L}_{KD} + \lambda_{id}\mathcal{L}_{id} + \lambda_{contr}\mathcal{L}_{contr}.$$

The contraction term penalizes repaired states that remain far from the clean reference:

$$\rho_l = \frac{\|S^{(l)}_{rep} - S^{(l)}_{clean}\|_F}{\|S^{(l)}_{corr} - S^{(l)}_{clean}\|_F + \epsilon_0}, \qquad \mathcal{L}_{contr} = \frac{1}{|L|}\sum_{l \in L} \max(0, \rho_l - \alpha)^2.$$

In the tuned run we use $\lambda_{id} = 4.0$, $\lambda_{contr} = 3.0$, and clean-batch probability 0.45.

4 Stability Metrics

We treat stability as a measurable object and report two primary metrics.

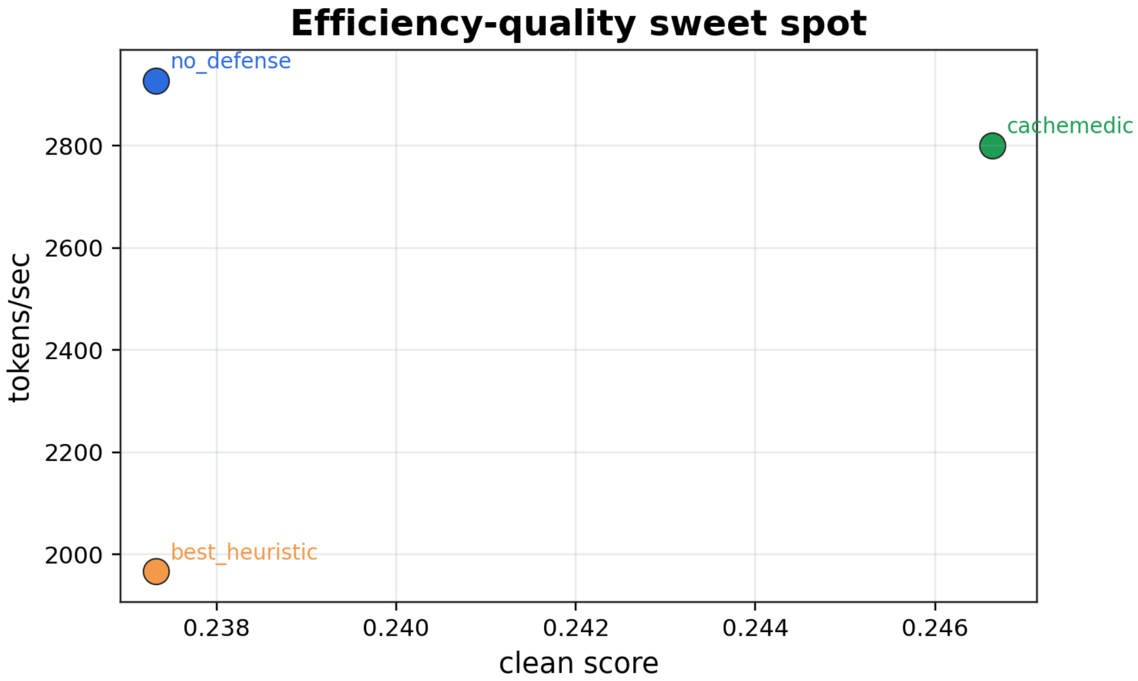

Figure 2: Clean regression vs overhead tradeoff for repair operator sweeps.

Logit sensitivity. We estimate expected next-token logit drift under finite-difference perturbations of the KV state:

$$\mathrm{Sens}(\delta) = \mathbb{E}_{p,d}\left[\|z_{\mathrm{top}k}(S+\delta d) - z_{\mathrm{top}k}(S)\|_2\right],$$

where p indexes prompts, d is a normalized perturbation direction, and $z_{\mathrm{top}k}$ selects top-k logits at the measurement token.

Amplification maps. We estimate layer/head-local gain factors

$$\gamma(l,h) = \frac{\|\Delta z\|_2}{\|\Delta S_{l,h}\|_F}$$

by injecting perturbations into one layer/head at a time and measuring induced logit drift.

Our default stability settings are $\delta \in \{0.0, 1.0, 2.0\}$, top-$k = 512$, 60 prompts, and 4 perturbation directions per prompt.

5 Experiments

We evaluate CacheMedic++ on Wikitext-2 perplexity, SST-2 prompted classification, and a deterministic needle-style long-context check. We report robustness curves versus corruption severity and stability metrics.

Setup. Our primary model is gpt2-medium; second-model confirmation uses gpt2-large. The primary evaluation uses 400 Wikitext-2 examples, 250 SST-2 examples, and 80 long-context needle examples, with robustness scored as the mean of task scores (SST-2 accuracy, inverse perplexity for Wikitext-2, and needle accuracy).

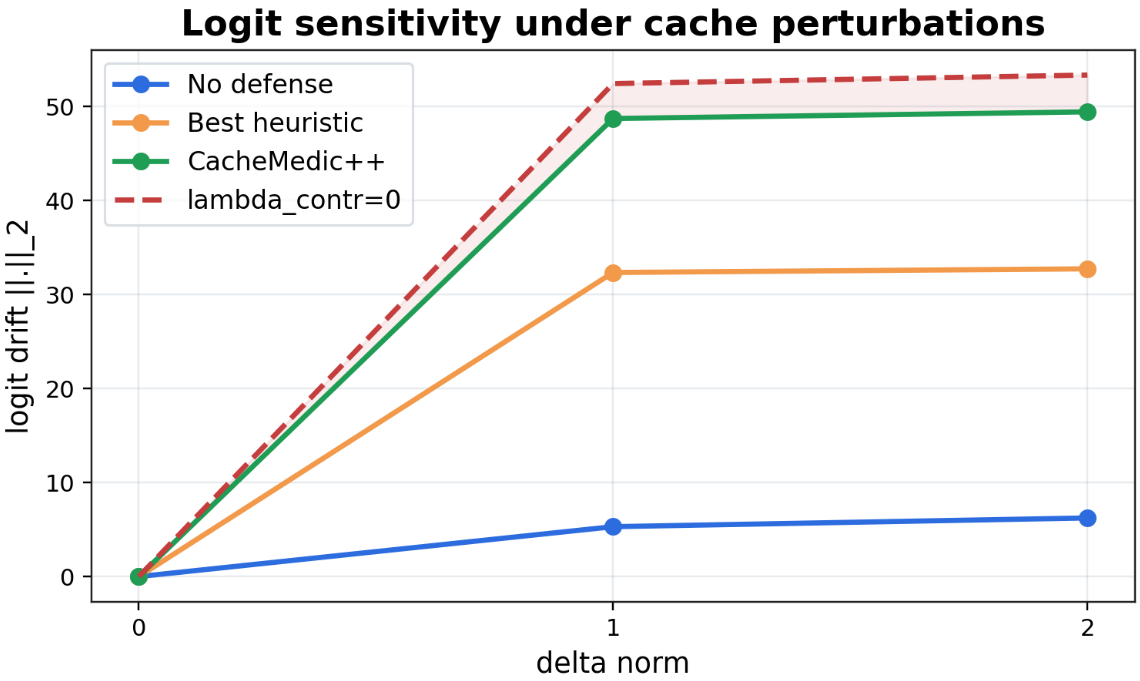

Figure 3: Logit sensitivity curves as a function of perturbation magnitude.

Method	Clean Score	Robustness AUC
No defense	0.2373	0.0384
Best heuristic (smoothing)	0.2373	0.0390
CacheMedic++	**0.2466**	**0.0401**

Table 1: Main robustness and clean-score results on gpt2-medium.

Primary-model outcome (gpt2-medium). On the canonical tuned V-only run, CacheMedic++ improves both clean score and robustness AUC versus no defense and the best heuristic (Table 1): clean score rises from 0.2373 (no defense) and 0.2373 (best heuristic) to 0.2466, while robustness AUC rises from 0.0384 and 0.0390 to 0.0401.

Contraction ablation (paired no_contr). Using the exact tuned pair, contraction improves robustness relative to the no-contraction ablation: robustness AUC increases from 0.0388 (no_contr) to 0.0401 (contr), a +0.0013 absolute gain. Stability also improves relative to no_contr with paired ratios 0.9291 at $\delta = 1.0$ and 0.9266 at $\delta = 2.0$. Clean WT2 regression remains small (+0.28% PPL relative to no defense), staying inside the project clean-regression bound.

Interpretation. The strongest claim supported by the paired evidence is that contraction improves the repaired model relative to the identical no-contraction setup. Absolute sensitivity versus the unmodified baseline remains a limitation and is reported transparently as part of the paper's scope.

Second-model confirmation (gpt2-large). The same operator family and training recipe transfer qualitatively to gpt2-large (Table 3): clean score improves from 0.2974 to 0.3001 and robustness AUC from 0.0472 to 0.0475.

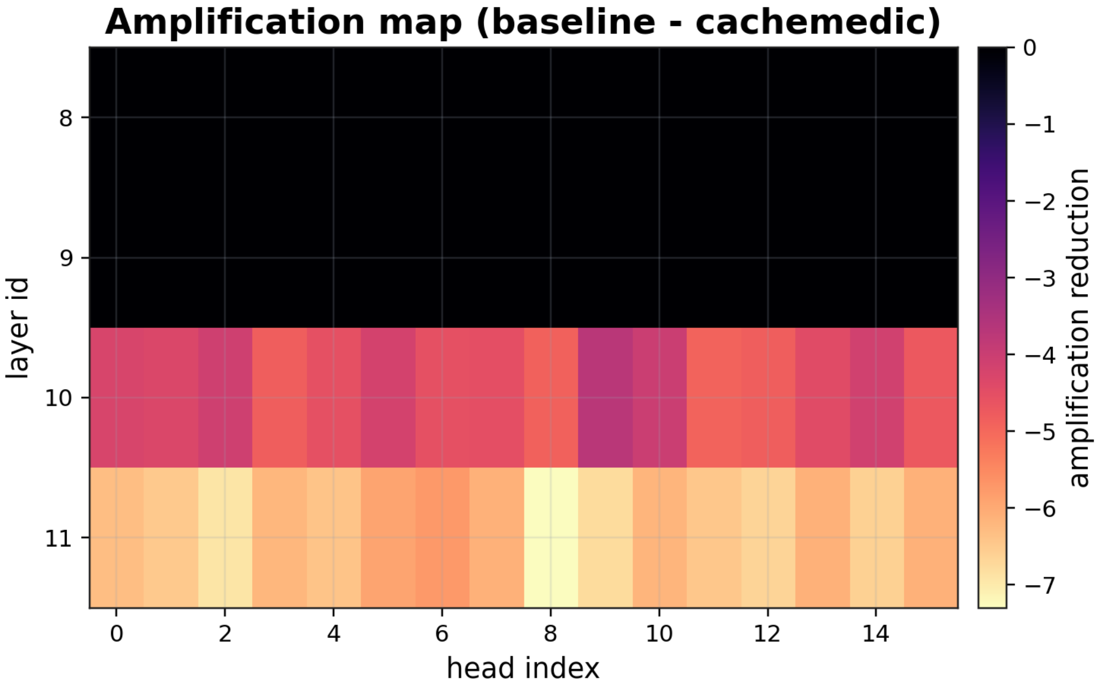

Figure 4: Amplification map heatmap $\gamma(l, h)$ before and after repair.

Metric	no_contr	contr
Robustness AUC	0.0388	**0.0401**
Logit sensitivity ($\delta = 1.0$)	52.4250	**48.7083**
Logit sensitivity ($\delta = 2.0$)	53.3308	**49.4163**
Sensitivity ratio contr/no_contr ($\delta = 1.0$)	0.9291	
Sensitivity ratio contr/no_contr ($\delta = 2.0$)	0.9266	
Repair family / apply_to / rank	A / V / 4	

Table 2: Paired ablation summary (no_contr vs contr) for the tuned V-only setup.

6 Related Work

We position CacheMedic++ at the intersection of (i) KV-cache corruption and manipulation, (ii) KV-cache interventions for behavior control, and (iii) stability and internal editing methods.

KV-cache corruption and manipulation. Prior work shows that perturbing cached keys/values can destabilize outputs or induce targeted behaviors without modifying model weights [3, 4, 1].

KV interventions and steering. Several methods directly manipulate internal states for steering or defense objectives, motivating targeted interventions at inference time [5].

Stability and internal editing. Activation-level constraints and editing techniques motivate framing internal state edits as operators with explicit stability properties [6, 2].

We differ by introducing a learned KV-specific repair operator with explicit contraction regularization and by treating stability metrics as primary results.

Method	Clean Score	Robustness AUC
No defense	0.2974	0.0472
Best heuristic (smoothing)	0.2974	0.0472
CacheMedic++	**0.3001**	**0.0475**

Table 3: Second-model confirmation (gpt2-large) from the tuned V-only setup.

7 Limitations and Ethics

This work evaluates stability under a reproducible set of corruption operators. Real-world faults may differ in distribution and timing. The method assumes access to model internals to insert the repair operator.

We treat the work as defensive robustness research. The harness avoids distributing exploit automation beyond minimal corruption operators required for evaluation.

References

[1] Prakhar Ganesh et al. Whose narrative is it anyway? a kv cache manipulation attack. *arXiv preprint arXiv:2511.12752*, 2025.

[2] Yiming Gao et al. Activation boundary defense for safeguarding large language models. In *Proceedings of ACL*, 2025.

[3] Md Tahmid Hossain et al. Can transformer memory be corrupted? *arXiv preprint arXiv:2510.17098*, 2025.

[4] Md Jamiul Nahian et al. Cachetrap: Exploiting kv cache bit flips for targeted behavior in llm inference. *arXiv preprint arXiv:2511.22681*, 2025.

[5] Levi Postmus and Marcos Abreu. Conceptor steering for language models. *arXiv preprint arXiv:2410.16314*, 2024.

[6] Y. Qiu et al. Spectral editing of activations. In *NeurIPS*, 2024.

A Appendix: Protocol Details

A.1 Corruption operators

The corruption family contains six deterministic operator types (Gaussian, dropout/zeroing, orthogonal rotation, sparse bitflip-like, quantization noise, contiguous overwrite), combined with masks over layer/head/time axes and severity $\epsilon \in \{0.0, 0.08, 0.16\}$.

A.2 Operator families

Operator families A/B/C are implemented in the harness. This paper reports the tuned Option A setting with V-only repair, rank 4, and two protected layers.

A.3 Training objective

Training minimizes $\mathcal{L}_{KD} + \lambda_{id}\mathcal{L}_{id} + \lambda_{contr}\mathcal{L}_{contr}$ with frozen base-model weights. The paired ablation toggles only λ_{contr} (3.0 vs 0.0).

A.4 Stability metrics

Stability is measured with finite-difference perturbations at $\delta \in \{0.0, 1.0, 2.0\}$ using top-$k = 512$ logits, 60 prompts, and 4 perturbation directions per prompt.