MONOTONE FFNS AND THE QUALITY–SENSITIVITY TRADE-OFF IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

How does a simple local architectural constraint change a language model's sensitivity to token substitutions? We study nonnegative weights in Transformer feed-forward networks (FFNs), while leaving attention, normalization, and other components unconstrained. With a non-decreasing activation such as ReLU, the constrained FFN branch is coordinatewise monotone in its own input, providing a local order-preserving bias without making the full Transformer monotone. We ask what this constraint changes empirically, what quality cost it incurs, and how far its local guarantees extend.

On `T5-small`, fixed-gradient HotFlip with a nominal five-token budget produces a mean reference-loss increase of 0.14 nats for the constrained model, compared with 0.35 for an unconstrained fine-tuned baseline; the fraction of examples exceeding a 10% loss increase falls from 59.5% to 17.0% (200 examples, one training seed). This reduced sensitivity comes with a cost: the constrained model has higher clean and attacked loss, and clean ROUGE-L decreases by approximately 2–5% relative across two full test sets and five training seeds. To clarify what the architectural constraint does and does not imply, we analyze branch Jacobians and bound the local contributions of saturated units under explicit assumptions, and separately test agreement with semantic order using post-hoc probes, finding no consistent improvement. An exploratory three-run Pythia-1.4B study with nonnegative FFN weights and GELU likewise finds smaller substitution-induced loss increments, under an evaluation objective that also scores edited tokens.

Together, the results identify a quality–sensitivity trade-off associated with nonnegative FFN weights while separating coordinatewise branch monotonicity, semantic ordering, and full-model robustness. The current experiments do not isolate persistent nonnegativity from the accompanying initialization change.

1 INTRODUCTION

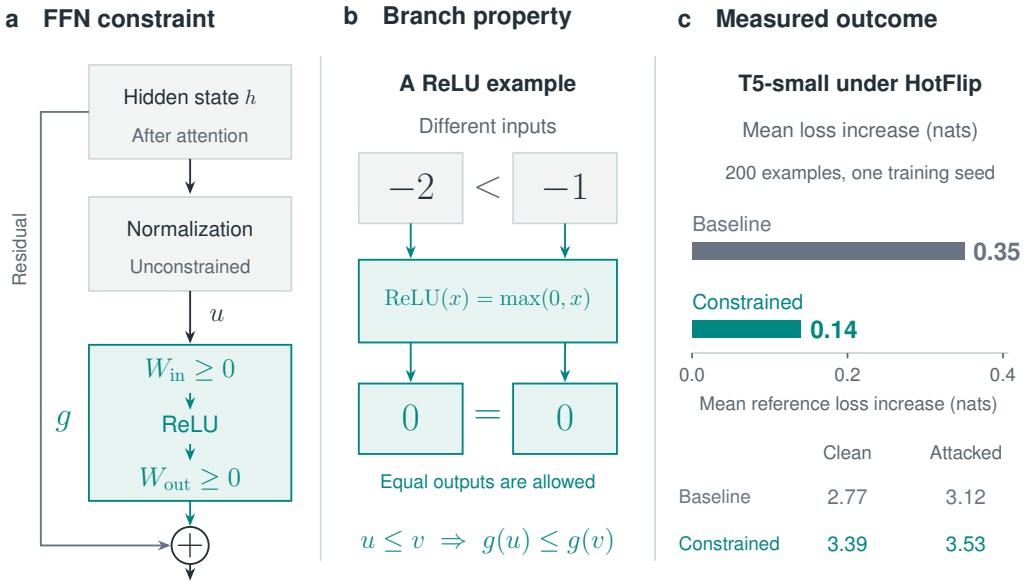
Small changes to a language model's input can substantially change its predictions. Most approaches to this sensitivity act through training objectives or inference-time defenses. A complementary possibility is architectural: can a simple structural constraint on the model's internal transformations make its response to input perturbations less sensitive?

We study one of the simplest such constraints—nonnegative weights in Transformer feed-forward networks (FFNs). When both FFN projections are nonnegative and the activation is non-decreasing, the branch is coordinatewise monotone in its input: increasing an input coordinate cannot decrease any output coordinate. This is a classical property of constrained neural networks (Sill, 1997; Daniels & Velikova, 2010; Runje & Shankaranarayana, 2023), and monotone classifiers have previously been studied against restricted evasion attacks (Íncer Romeo et al., 2018). We ask whether imposing this local order-preserving bias inside a pretrained Transformer changes its sensitivity to token substitutions, and what quality cost it incurs.

This question is deliberately narrower than asking whether the language model itself becomes monotone or robust. Attention, normalization, residual structure, and the output head remain unconstrained, and token substitutions need not produce coordinatewise ordered hidden states. Nonnegative FFN weights also do not bound a branch's Lipschitz constant. Thus local branch

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

Figure 1: **From a local FFN constraint to measured token-level sensitivity.** (a) In T5, both ReLU-branch weight matrices are constrained to be nonnegative, while normalization remains unconstrained. (b) The resulting branch is coordinatewise monotone: for example, $\mathrm{ReLU}(-2) = \mathrm{ReLU}(-1) = 0$, so $u \le v$ implies $g(u) \le g(v)$ without requiring strict increase. This guarantee applies only to the branch input; the GELU-based Pythia variant is not exactly monotone. (c) On 200 CNN/DailyMail examples at T5 seed 2024, fixed-gradient HotFlip with five nominal edits increases mean reference loss by 0.35 nats for the baseline and 0.14 for the constrained model. The constrained model is less sensitive to the tested substitutions, but has higher clean and attacked loss.



monotonicity neither implies full-model robustness nor semantic order preservation. Instead, it gives a controlled architectural intervention whose empirical consequences can be measured.

Our primary experiment converts `T5-small`'s ReLU FFNs into monotone branches and fine-tunes on summarization data. As illustrated in Figure 1, fixed-gradient HotFlip with a nominal five-token budget increases mean reference loss by 0.14 nats for the constrained model, compared with 0.35 for an unconstrained fine-tuned baseline on 200 held-out examples. The fraction of examples exceeding a 10% relative loss increase falls from 59.5% to 17.0%. This reduced sensitivity comes with a cost: clean ROUGE-L decreases by approximately 2–5% across two full test sets and five training seeds, and the constrained model has higher absolute attacked loss.

These results motivate two further questions. First, what local mechanism could connect the sign constraint to sensitivity? We analyze branch Jacobians and bound the contribution of saturated units under explicit assumptions. Second, does the same constraint improve agreement with semantically ordered inputs? We formalize projected semantic order and test it with post-hoc probes, finding no consistent improvement. Together, these analyses separate coordinatewise branch monotonicity, semantic ordering, and empirical sensitivity rather than treating them as interchangeable notions.

We also test the intervention on the decoder-only Pythia-1.4B. Because Pythia uses GELU, nonnegative FFN weights do not guarantee exact branch monotonicity. Across three exploratory runs, the constrained models again show smaller substitution-induced loss increments, but at a larger clean-loss cost and under an objective that also scores edited tokens.

Contributions. This paper makes three main contributions. First, we clarify what nonnegative FFN weights do and do not guarantee, separating coordinatewise branch monotonicity from projected semantic order and local sensitivity, and developing a saturation-based analysis under explicit assumptions. Second, on `T5-small`, we show that the constrained model is substantially less sensitive to the tested HotFlip substitutions, while incurring higher attacked loss and a clean-quality cost. Third, we use semantic-order probes and exploratory Pythia-1.4B experiments to test how far this local architectural constraint extends beyond the setting in which monotonicity is formally guaranteed, *and expose the limits of branch monotonicity*.

2 MONOTONE FEED-FORWARD BRANCHES

A Transformer alternates attention, feed-forward network (FFN) branches, residual connections, and normalization. We constrain each FFN branch only as a function of its own input; all other components remain unconstrained. The trained models operate in the original hidden-state coordinates, corresponding to $A = I$ in the projected-order framework introduced below. Semantic probes are used only for post-hoc evaluation, not for training or reparameterization.

Definition 2.1 (Coordinatewise monotonicity)**.** A function $g : \mathbb{R}^n \to \mathbb{R}^m$ is *coordinatewise monotone* if $u \le v$ elementwise implies $g(u) \le g(v)$ elementwise.

Lemma 2.2 (Closure under composition)**.** *If $f : \mathbb{R}^n \to \mathbb{R}^m$ and $g : \mathbb{R}^m \to \mathbb{R}^k$ are coordinatewise monotone, then $g \circ f$ is coordinatewise monotone.*

Proposition 2.3 (Sufficient condition for FFN monotonicity)**.** *Let N be a feed-forward network with elementwise non-decreasing activations, elementwise nonnegative weight matrices, and arbitrary biases. Then N is coordinatewise monotone.*

Proof. Each affine map with nonnegative weights preserves coordinatewise order, as does each activation. The claim follows by Lemma 2.2. □

Order-preserving refinement. The structural motivation is local order preservation: strengthening a representation along selected directions should not reverse its ordering along those directions. Let $A \in \mathbb{R}^{p\times d}$ have full row rank and define $h \preceq_A h' \iff Ah \le Ah'$. This relation is a preorder when $p < d$ and reduces to the usual coordinatewise order when $A = I_d$. A semantic interpretation requires directions whose ordering has been validated independently.

Let $A^\dagger$ be a fixed right inverse of A, so $AA^\dagger = I_p$. We define the residual refinement

$$F_A(h) = h + A^\dagger g(Ah), \qquad g : \mathbb{R}^p \to \mathbb{R}^p.$$

Proposition 2.4 (Sufficient condition for A-monotonicity)**.** *If g is coordinatewise monotone, then F_A preserves the preorder $\preceq_A$; that is, whenever $h \preceq_A h'$, we also have $F_A(h) \preceq_A F_A(h')$.*

Indeed, $AF_A(h) = Ah + g(Ah)$, so the claim follows from coordinatewise monotonicity of g; see Appendix B.1. This construction provides an abstract model of order-preserving refinement. Our experiments, however, constrain FFNs directly in the native hidden-state basis. They neither learn A nor implement F_A across normalization. This distinction lets us study the architectural bias without attributing a semantic guarantee to the trained Transformer.

Definition 2.5 (Transformer with monotone FFN)**.** A Transformer has *monotone FFN branches* if every FFN map g is coordinatewise monotone in its branch input u. The definition excludes normalization and residual addition from g and makes no monotonicity claim about the full Transformer.

For a pre-normalized block, the FFN contribution is $g(\mathrm{LN}(h))$. Even when g is monotone, the map $h \mapsto h + g(\mathrm{LN}(h))$ need not be: normalization can reverse coordinatewise order. In GPT-NeoX, parallel attention introduces additional unconstrained contributions (Appendix B.1). `T5−small` uses

$$g(u) = W_{\mathrm{o}}\,\mathrm{ReLU}(W_{\mathrm{i}}u)$$

and therefore satisfies Proposition 2.3 when both weight matrices are nonnegative. Pythia retains GELU, whose derivative is negative on part of its domain, so its constrained variant has nonnegative weights but is not exactly monotone. Accordingly, the shorthand "monotone" in experimental tables refers to the constrained arm and does not extend the formal guarantee to Pythia.

Parameterization and initialization. We parameterize each constrained matrix as

$$W = \phi(V), \qquad \phi(v) = \log(1 + \exp v),$$

with ϕ applied elementwise, which preserves nonnegativity throughout training. We initialize

$$V_0 = \log(\exp(|W_{\mathrm{pre}}| + \epsilon) - 1 + \epsilon\,), \qquad \epsilon = 10^{-4},$$

so that, in exact arithmetic,

$$\phi(V_0) = \log(\exp(|W_{\mathrm{pre}}| + \epsilon) + \epsilon) \approx |W_{\mathrm{pre}}|.$$

We retain pretrained biases. This conversion removes negative pretrained weight signs and therefore does not preserve the pretrained function. Appendix C.7 compares sign-frozen and unconstrained magnitude-initialized controls for clean loss; attack evaluations for these controls are unavailable.

The construction is restrictive. Nonnegative-weight sigmoidal networks admit monotone universal-approximation results (Daniels & Velikova, 2010), whereas each output of a nonnegative-weight ReLU network is convex in its input (Runje & Shankaranarayana, 2023). We therefore make no claim that constrained T5 FFNs retain unrestricted expressivity. Instead, we use this monotone-network construction as a localized architectural intervention within pretrained Transformers.

3 Experimental Setup

Appendices C.1, D, and C.3 provide the full training, evaluation, attack, and diagnostic protocols.

Encoder–decoder track (T5). We constrain both FFN projections in all 12 blocks of `T5−small` (Raffel et al., 2020). Each branch retains the pretrained form

$$g(u) = W_{\mathrm{o}} \,\mathrm{ReLU}(W_{\mathrm{i}} u)$$

and satisfies Proposition 2.3. The constraint affects approximately 25M of 60.5M parameters. The baseline starts from the same checkpoint and uses the same fine-tuning data and epoch budget. Warmup fractions are 10% for the baseline and 15% for the constrained model. We evaluate both fine-tuned arms over five seeds on the full CNN/DailyMail and XSUM test sets, neither of which appears in the fine-tuning data. The *standard* comparator is a single pretrained checkpoint.

Attacks and metrics (T5). We use a fixed-gradient HotFlip variant (Ebrahimi et al., 2018). For each model, the attack selects five input positions by gradient norm and replaces each token using its highest first-order vocabulary score. It does not recompute gradients after each edit or select candidates by realized loss. We then decode and re-encode the attacked sequence before scoring, so five selected edits define a nominal budget rather than a verified token distance after the round trip. Substitutions need not preserve meaning. For example i, let L_i and L_i^{adv} denote the reference-summary cross-entropies for the clean and attacked articles. We report

$$\Delta_i = L_i^{\mathrm{adv}} - L_i, \qquad r_i = \frac{\Delta_i}{L_i}, \qquad \mathrm{SR}_\tau = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\{r_i > \tau\},$$

with $\tau = 0.10$. Here, "success" means exceeding this loss threshold, not causing a semantic failure or jailbreak. We average Δ_i and r_i separately and report clean and attacked loss. Descriptive Wilson intervals summarize uncertainty across examples within a single training seed. We also archive five-token universal-trigger experiments (Wallace et al., 2019). Encoder truncation can remove the prepended triggers in these runs, so we do not use them as robustness evidence. Appendix C.3 describes the correction.

Order-preservation diagnostic. We evaluate agreement with ordered prompt pairs using a templated corpus of 702 adjacent-level pairs. Four-level chains span 13 semantic axes and 18 topics. Splitting by chain gives 507 probe-fitting pairs and 195 evaluation pairs. For each model and pooling choice, we fit 64 bootstrap logistic-regression directions on final-layer representations. At each layer, we measure the fraction of probe coordinates satisfying $Ah_{\mathrm{weak}} \leq Ah_{\mathrm{strong}}$. These correlated probe directions serve only as diagnostics and do not affect training.

Decoder-only track (Pythia-1.4B). We fine-tune Pythia-1.4B (Biderman et al., 2023), a 24-layer GPT-NeoX model, on the uncopyrighted subset of the Pile (Gao et al., 2020). A historical scope-screening study considers three constraint placements across all layers: ***MLP-in*** constrains only the input FFN projection, ***MLP-both*** constrains both FFN projections and matches the T5 scope, and ***MLP-in + Attn-out*** additionally constrains the attention output projection. Only MLP-both constrains every FFN matrix. Because Pythia retains GELU, none of these variants guarantees exact branch monotonicity. We do not use the scope-screening outcomes as comparative evidence.

Recovery training gives the constrained models a larger budget than the baseline: learning rates are 5×10^{-5} versus 10^{-5}, warmup fractions are 20% versus 10%, and training lasts 10 versus 5 epochs. Scope screening uses one seed and unequal budgets. We report runs labeled 42, 1337, and 2024; these labels do not establish independently retrained baselines. Transfer tables state their narrower coverage.

Attacks and controls (Pythia). HotFlip uses 200 texts, up to 10 substitutions, and a 15% relative-loss threshold. Ten-token universal triggers use three restarts and 50 search iterations, with 80 optimization texts and 120 evaluation texts. The deployed attacks measure NLL on the perturbed sequence, including edited tokens as prediction targets. The resulting loss therefore combines the surprisal of the substitutions themselves with their effects on the remaining text.

We also replay substitutions across models, apply random substitutions, and run a gradient-free query attack. Query search evaluates 20 candidates at each of 10 random positions using the target model's loss. These controls broaden the evaluation but do not remove the target-token confound or rule out gradient masking. Appendix C.3 gives candidate pools, transfer-set sizes, and query budgets.

4 Monotonicity and Local Gradient Contributions

Two local effects help organize what the constraint can change. Nonnegative weights with non-decreasing activations fix the signs of branch-Jacobian entries, while activation saturation can suppress individual path contributions. These are distinct properties with different assumptions. Figure 3 in Appendix B.7 illustrates the relevant paths and residual contribution.

Fix finite weights and write

$$g(u) = W_2\sigma(W_1u + b_1) + b_2, \qquad z = W_1u + b_1,$$

with $W_1 \in \mathbb{R}^{m\times p}$ and $W_2 \in \mathbb{R}^{p\times m}$. For analysis, define

$$T(u) = u + g(u)$$

and let $v = \nabla_{u'}\mathcal{L}$ at $u' = T(u)$. This auxiliary map is not the implemented pre-normalized Transformer block: normalization and, in Pythia, parallel attention introduce additional unconstrained Jacobian factors; see Remark **??** and Appendix B.7.

Lemma 4.1 (Nonnegative branch Jacobian)**.** *If $W_1, W_2 \geq 0$ and σ is non-decreasing and locally Lipschitz, then $J_g(u) \geq 0$ and $J_T(u) = I + J_g(u) \geq 0$ wherever differentiable. The same property extends to deeper MLPs with nonnegative weights, their Clarke generalized Jacobians, and automatic differentiation with nonnegative derivative conventions at kinks.*

The lemma applies to T5's ReLU branch, but not generally to Pythia's GELU branch. Nonnegative Jacobian entries eliminate cancellation among paths contributing to a single entry, but not after multiplication by a mixed-sign downstream gradient. Nor do they bound sensitivity: for example, $g(u) = cu$ with $c > 0$ is monotone while its Jacobian norm can be arbitrarily large.

Assumption 4.2 (Negative-tail saturation)**.** The activation has bounded derivative, satisfies $|\sigma'(z)| \to 0$ as $z \to -\infty$, and has $|\sigma'|$ non-decreasing on a tail $(-\infty, z^*]$. ReLU satisfies this with $z^* = 0$ and derivative convention $\sigma'(0) = 0$; exact GELU satisfies it with $z^* = -\sqrt{2}$.

Lemma 4.3 (Contribution of saturated units)**.** *Where the activation derivatives exist, define*

$$R_S(u) = \sum_{j\in S} \sigma'(z_j)W_2[:, j]W_1[j, :].$$

If $|\sigma'(z_j)| \leq \varepsilon$ for every $j \in S$ and $\|v\|_2 \leq G$, then

$$\left\|(I + J_g)^\top v - (I + J_g - R_S)^\top v\right\|_2 \leq \varepsilon G \sum_{j\in S} \|W_2[:, j]\|_2 \|W_1[j, :]\|_2.$$

For fixed weights, these contributions vanish as the corresponding pre-activations tend to $-\infty$ under Assumption 4.2, provided v remains bounded.

This bound does not require nonnegative weights. Nonnegativity of W_1 gives a separate persistence property: a coordinatewise decrease in u keeps a unit in its negative saturation region (Lemma B.1). Both results are local and hold at fixed weights. Suppressing one contribution can still increase the full gradient norm by removing cancellation with other terms. Since we do not measure activation occupancy or branch-Jacobian norms, the analysis identifies conditions under which selected branch contributions are suppressed, rather than establishing the mechanism behind the observed sensitivity difference. Appendices B and B.7 give proofs and counterexamples.

Table 1: Clean summarization quality on the full CNN/DailyMail test set of 11,490 articles. Scores are ROUGE F_1 $\times 100$. Fine-tuned rows give mean $\pm$ s.d. over five training seeds. ROUGE-L denotes the summary-level variant.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Pretrained	38.6	17.1	33.2
Baseline	39.8 ± 0.1	16.9 ± 0.0	33.2 ± 0.1
Monotone FFN	37.9 ± 0.2	15.2 ± 0.2	31.5 ± 0.2

Table 2: T5 HotFlip on 200 CNN/DailyMail examples at training seed 2024. The nominal budget is five selected token edits. Losses are mean reference-summary cross-entropies in nats. SR is the fraction with relative loss increase above 10%.

Model	Clean loss	Attacked loss	ΔLoss	Mean r_i	SR
Pretrained	2.58	2.99	0.40	19.9%	68.5%
Baseline	2.77	3.12	0.35	15.1%	59.5%
Monotone FFN	3.39	3.53	0.14	4.9%	17.0%

5 RESULTS

Clean T5 quality and training dynamics use five training seeds on full test sets. The primary T5 HotFlip evaluation uses one training seed and 200 examples. Pythia uses three runs and the unmasked attack objective. Each table states its coverage.

5.1 TRAINING DYNAMICS AND SUMMARIZATION QUALITY

The constrained model recovers substantially from the initial sign change but retains a clean-loss gap. Its first-epoch training loss is 4.99, compared with 2.94 for the baseline. Validation loss falls from 2.92 at epoch 1 to 2.51 at epoch 7, while the baseline reaches 2.21 (Table 5). Across five seeds, the largest standard deviations are approximately 0.022 for training loss and 0.012 for validation loss.

The seed-42 ablation helps explain this cost. An unconstrained model initialized from pretrained magnitudes also reaches validation loss 2.51, whereas a sign-frozen control reaches 2.27. This suggests that disrupting pretrained signs accounts for much of the clean-loss gap at this training budget. Because neither control has attack results, the comparison speaks only to clean quality (Appendix C.7).

The clean-quality cost is consistent across five training seeds. On CNN/DailyMail, ROUGE-L falls from 33.2 ± 0.1 to 31.5 ± 0.2 (Table 1), a 4.9% relative decrease on unrounded means. ROUGE-1 and ROUGE-2 decrease by 5.0% and 10.0%, respectively. On the full XSUM test set of 11,333 articles, ROUGE-L falls from 13.1 ± 0.0 to 12.8 ± 0.0, a 2.1% relative decrease (Table 6). Standard deviations displayed as 0.0 fall below the reported precision. Appendix C.2 gives the statistical procedures.

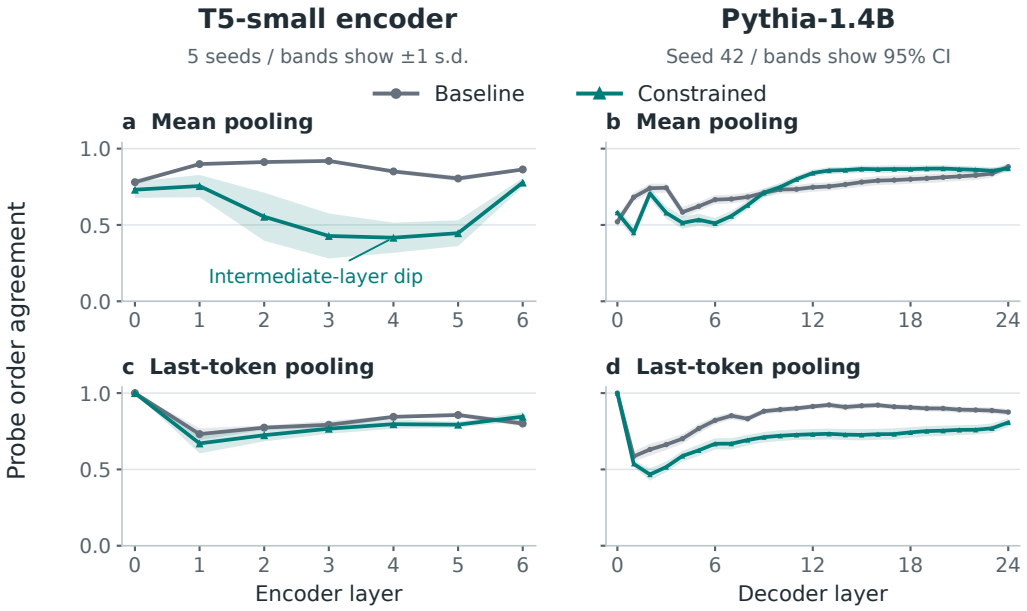
5.2 SENSITIVITY TO TOKEN SUBSTITUTIONS

HotFlip. Table 2 shows substantially smaller attack-induced loss increments for the constrained T5 model. Mean relative increase falls from 15.1% to 4.9%, while the success rate falls from 59.5% to 17.0%. The absolute increase likewise drops from 0.35 to 0.14 nats, so the difference is not driven solely by the larger clean-loss denominator. The constrained model nevertheless has higher attacked loss, 3.53 versus 3.12 nats.

The corresponding 95% Wilson intervals are 52.6–66.1% for the baseline success rate (119/200) and 12.4–22.8% for the constrained model (34/200). These intervals capture test-example uncertainty within seed 2024. An earlier seed-42 evaluation on 100 examples shows the same ordering (Appendix C.6). Because the archive lacks paired per-example losses, we cannot perform a paired statistical test; the primary result remains specific to the reported seed and fixed-gradient attack.

Universal triggers. The archived T5 trigger runs do not provide reliable evidence of trigger resistance: encoder truncation can remove prepended triggers from long inputs (Appendix C.3). We

Figure 2: **Probe order agreement depends on depth and pooling.** Columns separate models and rows separate pooling methods. Each curve gives the fraction of probe inequalities satisfied on ordered prompt pairs, with higher values indicating greater agreement. The constrained T5 encoder shows a pronounced intermediate-layer dip under mean pooling (a), while its last-token curve remains closer to the baseline (c). Pythia's later-layer comparison changes with pooling (b,d). Gray circles denote the baseline and teal triangles the constrained model. Bands show one standard deviation across five T5 seeds or 95% bootstrap intervals on 195 held-out Pythia pairs at seed 42. Constraints apply to ReLU FFNs in T5 and GELU FFNs in Pythia. Last-token agreement at layer 0 is trivially 1 because all prompts share the final token and equality counts as agreement. Probes are fitted separately for each model at its final layer.



Layer 0 is the embedding output. Last-token agreement there is 1 by construction.

therefore base the T5 sensitivity comparison on token substitutions and retain the trigger results only to document this evaluation limitation.

5.3 Order Preservation Along Semantic Directions

Coordinatewise branch monotonicity does not imply semantic order preservation through the full network. We test this distinction using ordered prompt pairs and post-hoc probes. Figure 2 compares mean pooling, which averages token representations, with last-token pooling. Probe order agreement measures the fraction of inequalities satisfied at each layer, with larger values indicating greater agreement with the prompt-pair ordering.

With mean pooling, the baseline T5 encoder scores 0.78–0.92 across layers. The constrained encoder drops to 0.42–0.45 at layers 3–5 before recovering to 0.78 at the output (panel a); this intermediate dip appears in every training seed. With last-token pooling, the models differ by at most 0.07 at any layer, and the constrained model scores slightly higher at the output, 0.85 versus 0.80 (panel c).

Pythia shows a similarly readout-dependent pattern. Under mean pooling, the constrained model exceeds the baseline at many later layers (panel b). Under last-token pooling, the baseline reaches 0.92 by layer 13, while the constrained model remains between 0.73 and 0.81 (panel d). Thus, the apparent ordering advantage depends on both depth and pooling.

Overall, the diagnostic shows no consistent improvement in semantic order preservation. The 64 correlated probe directions are fitted separately for each model at its final layer, so intermediate-layer agreement also reflects how each final-layer readout transfers across depth.

Table 3: Pythia-1.4B standalone fixed-gradient HotFlip results on 200 texts with 10 substitutions. Each entry is baseline → constrained. SR is the fraction with relative loss increase above 15%. NLL is in nats. All losses use the legacy unmasked objective, which includes substituted tokens as prediction targets.

Run label	Clean NLL	Attacked NLL	SR
42	$2.052 \rightarrow 3.830$	$2.873 \rightarrow 4.600$	$69.0\% \rightarrow 43.5\%$
1337	$2.053 \rightarrow 3.997$	$2.863 \rightarrow 4.733$	$68.5\% \rightarrow 39.0\%$
2024	$2.052 \rightarrow 3.825$	$2.866 \rightarrow 4.613$	$68.5\% \rightarrow 43.0\%$
Mean	$2.052 \rightarrow 3.884$	$2.868 \rightarrow 4.648$	$68.7\% \rightarrow 41.8\%$

Table 4: Pythia-1.4B transfer and gradient-free controls under the same legacy unmasked objective. Values are mean ± sample s.d. HotFlip transfer uses runs 42 and 1337. Random and query controls also include replacement run 2024. Deg. is mean per-example relative loss increase, SR uses a 15% threshold, and NLL shows clean → attacked means in nats. Substitution results use 200 texts; query search uses a 50-text subset.

		Evaluated on baseline			Evaluated on constrained		
Attack	Crafted on	Deg.	SR	NLL	Deg.	SR	NLL
HotFlip	baseline	$55.1 \pm 0.6\%$	$68.8 \pm 0.4\%$	$2.05 \rightarrow 2.87$	$14.9 \pm 1.5\%$	$25.3 \pm 1.8\%$	$3.91 \rightarrow 4.47$
HotFlip	constrained	$64.6 \pm 0.9\%$	$72.0 \pm 1.4\%$	$2.05 \rightarrow 2.98$	$19.8 \pm 1.5\%$	$41.3 \pm 3.2\%$	$3.91 \rightarrow 4.67$
Random	neither	$64.6 \pm 0.3\%$	$73.0 \pm 1.0\%$	$2.05 \rightarrow 2.97$	$17.7 \pm 1.3\%$	$36.5 \pm 2.3\%$	$3.88 \rightarrow 4.55$
Query	target	$112.3 \pm 4.0\%$	$77.3 \pm 1.2\%$	$1.92 \rightarrow 3.31$	$33.1 \pm 2.6\%$	$50.0 \pm 3.5\%$	$3.74 \rightarrow 4.90$

5.4 Exploratory Decoder-Only Evaluation

We evaluate the nonnegative-weight MLP-both configuration of Pythia-1.4B on runs 42, 1337, and the completed replacement run 2024. The reported statistics describe these runs; the records do not establish independently retrained baselines. Appendix E documents result provenance and cache exclusions.

Table 3 reports model-specific fixed-gradient HotFlip results. The fraction of examples exceeding a 15% relative loss increase falls from 68.7% to 41.8%. The mean absolute increase changes much less, from 0.815 nats for the baseline to 0.765 for the constrained model. Because the constrained model starts from substantially higher loss, its attacked NLL also remains higher, at 4.648 versus 2.868 nats. All quantities come from the same attack exports rather than a separate clean-perplexity evaluation.

The gradient-free controls in Table 4 also show smaller loss increments for the constrained model. Across three runs, random substitutions increase NLL by 0.914 versus 0.662 nats, while query search increases it by 1.394 versus 1.166 nats. Transfer is asymmetric on the two retained runs: baseline-crafted substitutions affect the constrained model less than its own substitutions do, whereas constrained-model substitutions affect the baseline more than the baseline's own fixed-gradient attack. Random substitutions also damage the baseline more than fixed-gradient HotFlip, indicating that HotFlip is a weak search procedure in this setting.

Universal triggers on 120 evaluation texts produce three-run relative NLL increases of $16.6 \pm 0.9\%$ for the baseline and $8.5 \pm 0.4\%$ for the constrained model. Their absolute increases are nearly identical, at 0.334 ± 0.017 and 0.326 ± 0.007 nats, showing that the relative metric amplifies the apparent difference. Appendix D.6 gives per-run values and qualified historical transfer results.

Overall, the Pythia experiments reproduce the pattern of smaller measured loss increments under the archived unmasked objective, but with a substantial clean-quality cost and higher attacked loss. Because the objective also scores modified tokens, it mixes their direct surprisal with their effects on subsequent predictions. These results therefore characterize measured substitution sensitivity rather than contextual robustness, and they do not rule out gradient masking (Athalye et al., 2018; Carlini et al., 2019; Tramèr et al., 2020).

6 Related Work

Token perturbations and robustness evaluation. Our evaluation builds on gradient-guided token substitutions (Ebrahimi et al., 2018), universal triggers (Wallace et al., 2019), prompt search (Shin et al., 2020), continuous relaxations of discrete attacks (Guo et al., 2021), query-based substitutions (Jin et al., 2020), and attacks on sequence-to-sequence models (Cheng et al., 2020). These loss-based evaluations differ from jailbreak attacks that target harmful responses from aligned models (Zou et al., 2023). Existing defenses include adversarial training (Howe et al., 2025), randomized input perturbation and output aggregation (Robey et al., 2023), input filtering (Jain et al., 2023; Khachaturov & Mullins, 2025), representation interventions (Zou et al., 2024), and certified training for word-substitution robustness (Jia et al., 2019). Our approach instead modifies the internal FFN architecture and is complementary to these methods; we do not benchmark against them.

Robustness evaluation should use adaptive attacks and test for obfuscated gradients (Athalye et al., 2018; Carlini et al., 2019; Tramèr et al., 2020). Our Pythia experiments include gradient-free and transfer controls, but they neither establish robustness to stronger attacks nor resolve whether persistent nonnegativity drives the T5 effect.

Monotonic networks and structural constraints. Sign-constrained sigmoidal networks (Daniels & Velikova, 2010) and min–max networks (Sill, 1997) admit universal-approximation results within monotone function classes. Lattice models (Gupta et al., 2016; You et al., 2017), verification-based methods (Liu et al., 2020; Sivaraman et al., 2020), and alternative activation constructions (Runje & Shankaranarayana, 2023; Sartor et al., 2025) offer different expressivity and optimization trade-offs. Monotone classifiers have also been applied to restricted malware-evasion attacks (Íncer Romeo et al., 2018). Other architectures combine monotonicity with explicit Lipschitz control (Nolte et al., 2023; Kitouni et al., 2023), while Lipschitz-constrained attention and Transformers directly bound sensitivity (Kim et al., 2021; Newhouse et al., 2025). Nonnegative weights alone provide no such norm bound.

Our focus differs from these works: we apply a classical sign constraint selectively to FFNs inside pretrained Transformers and study its empirical effect on token-level sensitivity, clean quality, and semantic-order agreement. Appendix A provides additional context.

7 Discussion and Conclusion

Nonnegative FFN weights offer a simple way to study how a local architectural constraint changes language-model sensitivity. In `T5−small`, the constrained model shows substantially smaller HotFlip-induced reference-loss increments, but pays a consistent clean-quality cost and retains higher absolute attacked loss. The exploratory Pythia experiments reproduce the distinction between smaller loss increments and worse absolute performance under a different evaluation objective. Semantic-order probes show no consistent improvement in order agreement.

Together, these results separate three notions that can otherwise be conflated: coordinatewise branch monotonicity, semantic order preservation, and sensitivity to token substitutions. Nonnegative ReLU FFNs guarantee the first, but neither of the latter two follows from that guarantee. The experiments nevertheless show that imposing this local structural bias can substantially change measured substitution sensitivity, revealing a quality–sensitivity trade-off that merits further study.

Scope and limitations. The T5 results characterize the complete conversion and fine-tuning procedure; the available controls do not isolate persistent nonnegativity from magnitude initialization. The primary attack evaluation uses one training seed, fixed-gradient search, and a nominal edit budget that does not verify the final token distance, while clean-quality evaluation spans five seeds and full test sets. Pythia uses unequal training schedules and an objective that scores modified tokens, so we treat it as exploratory. Finally, the local gradient analysis identifies conditions under which saturated units contribute less to branch gradients but does not establish that saturation causes the observed sensitivity differences. Our conclusions therefore concern the tested models, tasks, and loss-based evaluations rather than general adversarial robustness or deployment safety.

AI USE STATEMENT

Generative AI tools (Anthropic's Claude via Claude Code and OpenAI's Codex) assisted with pipeline development and debugging, statistical aggregation and interpretation, mathematical checking, literature and reference checks, figure preparation, and manuscript drafting and editing. For this revision, Codex was also used for independent reviewer-style critiques, comparison of claims with archived outputs and source code, and revision of the theoretical and empirical scope. The AI-generated reviews are internal editorial assistance, not conference peer reviews or evidence of acceptance. The authors are responsible for verifying the experiments, numerical claims, proofs, references, and final text, and for ensuring that this disclosure and the submission-form disclosure accurately reflect the work.

ETHICS STATEMENT

This work evaluates previously published token-level attacks on public pretrained models and summarization/language-modeling datasets. No human subjects or private data are involved. Smaller loss increments on these benchmarks should not be interpreted as evidence that a model is safe against jailbreaks or appropriate for safety-critical deployment. Reporting clean and attacked performance together, and disclosing known evaluation defects, is necessary to avoid overstating the protection offered by architectural constraints.

REPRODUCIBILITY STATEMENT

Sections 2 and 3 specify the constraint, initialization, and evaluation metrics. Appendices C.1–C.3 give the training, decoding, attack, statistical, and hardware settings. Appendices B.1–B.7 provide the theoretical derivations. Tables state their sample and seed coverage. Appendix E records the completed replacement run, cache exclusions, and differences between deployed evaluators and local code corrections. Exact reconstruction is limited by missing executable revisions and paired per-example T5 attack losses.

REFERENCES

Brandon Amos and J. Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

David Angeli and Eduardo D. Sontag. Monotone control systems. *IEEE Transactions on Automatic Control*, 48(10):1684–1698, 2003. doi: 10.1109/TAC.2003.817920.

Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Advances in Neural Information Processing Systems*, 37:129696–129742, 2024.

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018. arXiv:1802.00420.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 2397–2430. PMLR, 2023.

Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*, 2019.

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 3601–3608, 2020. AAAI-20 Technical Tracks 4.

Jan Chorowski and Jacek M. Zurada. Learning understandable neural networks with nonnegative weight constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 26(1):62–69, 2015. doi: 10.1109/TNNLS.2014.2310059.

Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010. doi: 10.1109/TNN.2010.2044803.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2310.06474.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 31–36. Association for Computational Linguistics, 2018.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5747–5757, 2021.

Maya Gupta, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, and Alexander van Esbroeck. Monotonic calibrated interpolated look-up tables. *Journal of Machine Learning Research*, 17(109):1–47, 2016. URL https://jmlr.org/papers/v17/15-243.html.

Morris W. Hirsch and Hal Smith. Monotone dynamical systems. In *Handbook of Differential Equations: Ordinary Differential Equations*, volume 2, chapter 4, pp. 239–357. Elsevier, 2006. doi: 10.1016/S1874-5725(05)80006-9.

Nikolaus H. R. Howe, Ian R. McKenzie, Oskar John Hollinsworth, Michał Zając, Tom Tseng, Aaron David Tucker, Pierre-Luc Bacon, and Adam Gleave. Scaling trends in language model robustness. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24080–24138. PMLR, 2025. arXiv:2407.18213.

Christian Igel. Smooth min-max monotonic networks. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. arXiv:2306.01147.

Íñigo Íncer Romeo, Michael Theodorides, Sadia Afroz, and David Wagner. Adversarially robust malware detection using monotonic classification. In *Proceedings of the Fourth ACM International Workshop on Security and Privacy Analytics (IWSPA)*, 2018. doi: 10.1145/3180445.3180449.

Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping-yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 4129–4142, 2019.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8018–8025, 2020. AAAI-20 Technical Tracks 5.

David Khachaturov and Robert Mullins. Adversarial suffix filtering: a defense pipeline for llms. *arXiv preprint arXiv:2505.09602*, 2025.

Hyunjik Kim, George Papamakarios, and Andriy Mnih. The Lipschitz constant of self-attention. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5562–5571. PMLR, 2021.

Ouail Kitouni, Niklas Nolte, and Mike Williams. Robust and provably monotonic networks. *Machine Learning: Science and Technology*, 4, 2023. doi: 10.1088/2632-2153/aced80. arXiv:2112.00038.

Miles Q Li and Benjamin CM Fung. Security concerns for large language models: A survey. *Journal of Information Security and Applications*, 95:104284, 2025. doi: 10.1016/j.jisa.2025.104284.

Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. arXiv:2011.10219.

Dan Mikulincer and Daniel Reichman. Size and depth of monotone neural networks: Interpolation and approximation. In *Advances in Neural Information Processing Systems 35 (NeurIPS)*, 2022. arXiv:2207.05275.

Laker Newhouse, R Preston Hess, Franz Cesista, Andrii Zahorodnii, Jeremy Bernstein, and Phillip Isola. Training transformers with enforced lipschitz constants. *arXiv preprint arXiv:2507.13338*, 2025.

Niklas Nolte, Ouail Kitouni, and Mike Williams. Expressive monotonic neural networks. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=w2P7fMy_RH.

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

Alexander Robey, Eric Wong, Hamed Hassani, and George J. Pappas. SmoothLLM: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684*, 2023. URL https://arxiv.org/abs/2310.03684.

Davor Runje and Sharath M Shankaranarayana. Constrained monotonic neural networks. In *International Conference on Machine Learning*, pp. 29338–29353. PMLR, 2023.

Davide Sartor, Alberto Sinigaglia, and Gian Antonio Susto. Advancing constrained monotonic neural networks: Achieving universal approximation beyond bounded activations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52995–53018. PMLR, 2025. arXiv:2505.02537.

Md. Faiyaz Abdullah Sayeedi, Maaz Bin Hossain, Md. Kamrul Hassan, Sabrina Afrin, Molla Md. Sabit Hossain, and Md. Shohrab Hossain. JailbreakTracer: Explainable detection of jailbreaking prompts in LLMs using synthetic data generation. *IEEE Access*, 13:123708–123723, 2025. doi: 10.1109/ACCESS.2025.3579996.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, 2020.

Joseph Sill. Monotonic networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 1997.

Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck. Counterexample-guided learning of monotonic neural networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/8ab70731b1553f17c11a3bbc87e0b605-Abstract.html.

Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. On adaptive attacks to adversarial example defenses. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. arXiv:2002.08347.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2153–2162. Association for Computational Linguistics, 2019.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial GLUE: A multi-task benchmark for robustness evaluation of language models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks)*, 2021. URL https://arxiv.org/abs/2111.02840.

Antoine Wehenkel and Gilles Louppe. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019. arXiv:1908.05164.

Ezra Winston and J. Zico Kolter. Monotone operator equilibrium networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2020. arXiv:2006.08591.

Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. In *Advances in Neural Information Processing Systems 30 (NIPS)*, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html.

Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. AutoDAN: Interpretable gradient-based adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023. URL https://arxiv.org/abs/2310.15140.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023. URL https://arxiv.org/abs/2307.15043.

Andy Zou, Long Phan, Justin Wang, Derek Duenas, Maxwell Lin, Maksym Andriushchenko, Rowan Wang, Zico Kolter, Matt Fredrikson, and Dan Hendrycks. Improving alignment and robustness with circuit breakers. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, pp. 83345–83373, 2024. arXiv:2406.04313.

A Extended Related Work

Token-level perturbations and language-model threat models. Token-level robustness has been studied through gradient-guided substitutions (Ebrahimi et al., 2018), universal triggers (Wallace et al., 2019), prompt search (Shin et al., 2020), continuous relaxations of discrete attacks (Guo et al., 2021), query-based substitutions (Jin et al., 2020), and attacks on sequence-to-sequence models (Cheng et al., 2020). Adversarial benchmarks also evaluate models under linguistic perturbations (Wang et al., 2021).

Jailbreak attacks address a different threat model: they seek to bypass the safety behavior of aligned models using transferable suffixes (Zou et al., 2023), interpretable adversarial prompts (Zhu et al., 2023), many-shot attacks (Anil et al., 2024), or multilingual attacks (Deng et al., 2024). Broader security surveys also consider prompt injection (Li & Fung, 2025). Our experiments instead measure summarization and language-modeling loss under token edits; they do not measure harmful-response rates or instruction-hierarchy violations.

Defenses and robustness evaluation. Inference-time defenses include randomized input perturbation and output aggregation (Robey et al., 2023), prompt classifiers (Sayeedi et al., 2025), perplexity filtering and paraphrasing (Jain et al., 2023), and adversarial-suffix filtering (Khachaturov & Mullins, 2025). Training-time approaches include adversarial training (Howe et al., 2025), representation interventions (Zou et al., 2024), and certified training against word substitutions (Jia et al., 2019). Their assumptions and guarantees differ from ours, so we do not compare performance directly.

Robustness evaluation should also account for adaptive attacks and obfuscated gradients (Athalye et al., 2018; Carlini et al., 2019; Tramèr et al., 2020). Transfer, random-substitution, and query-based controls broaden our Pythia evaluation, but their finite search budgets and shared unmasked objective limit the conclusions they support.

Monotone networks and structural constraints. Classical monotone constructions use nonnegative weights with sigmoidal activations (Daniels & Velikova, 2010) or min–max combinations of constrained linear units (Sill, 1997). Lattice models support practical partial-monotonicity constraints (Gupta et al., 2016; You et al., 2017), while verification and counterexample-guided methods avoid imposing a fixed sign pattern (Liu et al., 2020; Sivaraman et al., 2020). Other approaches integrate positive derivatives (Wehenkel & Louppe, 2019) or use smooth min–max constructions (Igel, 2024).

Nonnegative-weight ReLU networks incur an additional convexity restriction; modified activation constructions can recover greater expressivity (Runje & Shankaranarayana, 2023; Sartor et al., 2025). Depth and width requirements (Mikulincer & Reichman, 2022) and interpretability of nonnegative-weight networks (Chorowski & Zurada, 2015) have also been studied. We deliberately use the simpler sign constraint to study its effect when inserted selectively into a pretrained Transformer.

Sensitivity control and neighboring notions of monotonicity. Monotone classifiers can resist restricted evasion strategies when permissible feature changes respect the underlying order (Íncer Romeo et al., 2018); arbitrary token substitutions do not satisfy this premise. Architectures that combine monotonicity with Lipschitz control provide explicit sensitivity bounds (Nolte et al., 2023; Kitouni et al., 2023), and related work constrains the Lipschitz behavior of attention and Transformers (Kim et al., 2021; Newhouse et al., 2025). Nonnegative weights alone provide no such norm bound.

Other uses of "monotonicity" are conceptually distinct. Monotone dynamical systems study order-preserving flows (Hirsch & Smith, 2006; Angeli & Sontag, 2003), while monotone-operator equilibrium networks use operator monotonicity to ensure well-posedness (Winston & Kolter, 2020). OptNet similarly illustrates how architectural structure can be imposed through differentiable optimization layers (Amos & Kolter, 2017). None of these theories supplies a global monotonicity or robustness certificate for the pre-normalized Transformers studied here.

B THEORY AND PROOFS

This section collects the proofs and local calculations underlying Sections 2 and 4. We first establish the projected-order result, then relate the idealized construction to the implemented Transformer blocks, and finally analyze branch Jacobians, saturation, and their limitations.

B.1 PROJECTED-ORDER REFINEMENT

Proof of Proposition 2.4. If $Ah \le Ah'$, coordinatewise monotonicity of g gives $g(Ah) \le g(Ah')$. Since $AA^\dagger = I_p$,

$$\begin{aligned} AF_A(h) &= Ah + AA^\dagger g(Ah) \\ &= Ah + g(Ah) \\ &\le Ah' + g(Ah') \\ &= AF_A(h'). \end{aligned}$$

Hence $F_A(h) \preceq_A F_A(h')$. □

When $p < d$, $\preceq_A$ is a preorder rather than a partial order: $h \preceq_A h' \preceq_A h$ exactly when $h - h' \in \ker(A)$. Thus the constraint cannot distinguish states that differ only along $\ker(A)$. For $A = I_d$, the relation reduces to the usual coordinatewise partial order.

B.2 PLACEMENT INSIDE PRE-NORMALIZED TRANSFORMER BLOCKS

Let $u = N(h)$ denote the normalized input to an FFN branch. T5 uses root-mean-square normalization and

$$g(u) = W_{\mathrm{o}} \operatorname{ReLU}(W_{\mathrm{i}} u), \qquad F_{\mathrm{T5}}(h) = h + g(N(h)).$$

Hence

$$J_{F_{\mathrm{T5}}}(h) = I + J_g(N(h)) J_N(h).$$

Although J_g is elementwise nonnegative under our T5 constraint, J_N need not be. Branch monotonicity therefore does not extend to the complete residual map.

Pythia uses a GPT-NeoX parallel residual block,

$$F_{\mathrm{Pythia}}(h) = h + a(N_1(h)) + g(N_2(h)),$$

with Jacobian

$$J_{F_{\mathrm{Pythia}}}(h) = I + J_a(N_1(h)) J_{N_1}(h) + J_g(N_2(h)) J_{N_2}(h).$$

Neither attention nor normalization is constrained to be coordinatewise monotone. Moreover, Pythia retains GELU, so even its FFN branch need not have a nonnegative Jacobian.

The auxiliary map $T(u) = u + g(u)$ used in Section 4 therefore isolates the FFN contribution; it is not the implemented Transformer block.

For completeness, the idealized semantic construction

$$F_A(h) = h + A^\dagger g(Ah)$$

has Jacobian

$$J_{F_A}(h) = I + A^\dagger J_g(Ah) A.$$

Consequently,

$$\nabla_h \ell(F_A(h)) = \nabla \ell(F_A(h)) + A^\top J_g(Ah)^\top (A^\dagger)^\top \nabla \ell(F_A(h)).$$

Writing $s = Ah$ and $q = (I - A^\dagger A)h$ gives

$$F_A(h) = q + A^\dagger T(s), \qquad T(s) = s + g(s).$$

At fixed q, the downstream gradient in the projected coordinates is $(A^\dagger)^\top \nabla \ell$. The experiments do not train this semantic construction or learn A.

B.3 Proof of Closure Under Composition

Proof of Lemma 2.2. For $x \le x'$, monotonicity of f gives $f(x) \le f(x')$. Applying monotonicity of g gives

$$g(f(x)) \le g(f(x')).$$

□

B.4 Nonnegative Branch Jacobians

Proof of Lemma 4.1. At every differentiability point, coordinatewise monotonicity implies that each partial derivative of g is nonnegative. Equivalently, when all activations are differentiable,

$$J_g(u) = W_2 \operatorname{diag}\big(\sigma'(W_1 u + b_1)\big) W_1,$$

whose factors are elementwise nonnegative. Hence $J_g(u) \ge 0$, and therefore

$$J_T(u) = I + J_g(u) \ge 0.$$

The same argument extends to deeper MLPs. Because locally Lipschitz activations yield locally Lipschitz networks, every Clarke generalized Jacobian is a convex combination of limits of classical Jacobians. The cone of elementwise nonnegative matrices is closed and convex, so these generalized Jacobians remain nonnegative. Automatic differentiation also preserves the conclusion whenever the derivative convention at activation kinks is nonnegative. □

Nonnegativity fixes signs but does not bound sensitivity. For example,

$$g(u) = cu, \qquad c > 0,$$

is monotone while $\|J_g\|_2 = c$ can be arbitrarily large. The same phenomenon occurs for $g(u) = c\,\mathrm{ReLU}(u)$ on the positive orthant. Nonnegative matrices can also have full rank.

Nor do nonnegative Jacobian entries prevent cancellation after multiplication by a mixed-sign downstream gradient. With all ReLUs active, take

$$W_1 = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \qquad W_2 = I_2, \qquad v = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Then $J_g = W_1$ and

$$J_g^\top v = 0,$$

even though both hidden units contribute nonzero terms.

B.5 Saturated-Unit Contributions

Proof of Lemma 4.3. At differentiability points,

$$J_g(u) = \sum_{j=1}^{m} \sigma'(z_j) W_2[:, j] W_1[j, :].$$

For $S \subseteq \{1, \dots, m\}$, the contribution removed in the lemma is

$$R_S(u)^\top v = \sum_{j \in S} \sigma'(z_j) W_1[j, :]^\top \big(W_2[:, j]^\top v\big).$$

The triangle and Cauchy–Schwarz inequalities give

$$\|R_S(u)^\top v\|_2 \le \sum_{j \in S} |\sigma'(z_j)| \|W_1[j, :]\|_2 \|W_2[:, j]\|_2 \|v\|_2,$$

and therefore

$$\|R_S(u)^\top v\|_2 \le \varepsilon G \sum_{j \in S} \|W_2[:, j]\|_2 \|W_1[j, :]\|_2.$$

No step requires nonnegative weights. For fixed finite weights and bounded v, Assumption 4.2 makes this contribution vanish as the corresponding pre-activations tend to $-\infty$. □

Suppressing a branch contribution need not reduce the complete gradient norm. For example, let

$$W_1 = (1, 0), \qquad W_2 = (0, 1)^\top, \qquad v = \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

with the corresponding ReLU active. Then

$$J_g = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

and

$$(I + J_g)^\top v = (0, -1)^\top,$$

which has norm 1. Setting the unit's derivative to zero instead gives v, whose norm is $\sqrt{2}$.

If $v \geq 0$ or $v \leq 0$ elementwise, $W_1, W_2 \geq 0$, and $\sigma' \geq 0$, then the derivative coefficients contribute with consistent signs. Decreasing these coefficients while holding the weights and v fixed can only decrease each gradient coordinate in absolute value. Comparisons across different inputs or independently trained models need not satisfy these conditions.

For a pre-normalized block, the corresponding bound acquires a factor $\|J_N\|_2$. Small branch contributions therefore do not imply small full-block gradients.

B.6 Exact GELU and One-Sided Persistence

The Pythia-1.4B configuration uses exact GELU,

$$\sigma(z) = z\Phi(z), \qquad \sigma'(z) = \Phi(z) + z\varphi(z), \qquad \sigma''(z) = (2 - z^2)\varphi(z),$$

where Φ and φ denote the standard-normal cumulative distribution and density. Its derivative is negative for $z < -0.75179\ldots$ and reaches a minimum of approximately -0.12890 at $z = -\sqrt{2}$. Lemma 4.1 therefore does not apply to the Pythia FFN.

Exact GELU nevertheless satisfies Assumption 4.2: $|\sigma'(z)| \to 0$ as $z \to -\infty$, and $|\sigma'|$ is non-decreasing on $(-\infty, -\sqrt{2}]$. ReLU satisfies the assumption with $z^* = 0$ under the convention $\sigma'(0) = 0$. Softplus and logistic activations satisfy it on sufficiently negative tails as well.

Lemma B.1 (Persistence of saturated units)**.** *Suppose Assumption 4.2 holds, $W_1 \geq 0$, and $z_j(u) \leq z^*$ with $|\sigma'(z_j(u))| \leq \varepsilon$. Then these inequalities also hold at $u + \delta$ for every $\delta \leq 0$, wherever the derivatives are defined.*

Proof. Since $W_1 \geq 0$ and $\delta \leq 0$,

$$z_j(u + \delta) = z_j(u) + (W_1\delta)_j \leq z_j(u) \leq z^*.$$

Monotonicity of $|\sigma'|$ on the negative tail therefore gives

$$|\sigma'(z_j(u + \delta))| \leq |\sigma'(z_j(u))| \leq \varepsilon.$$

□

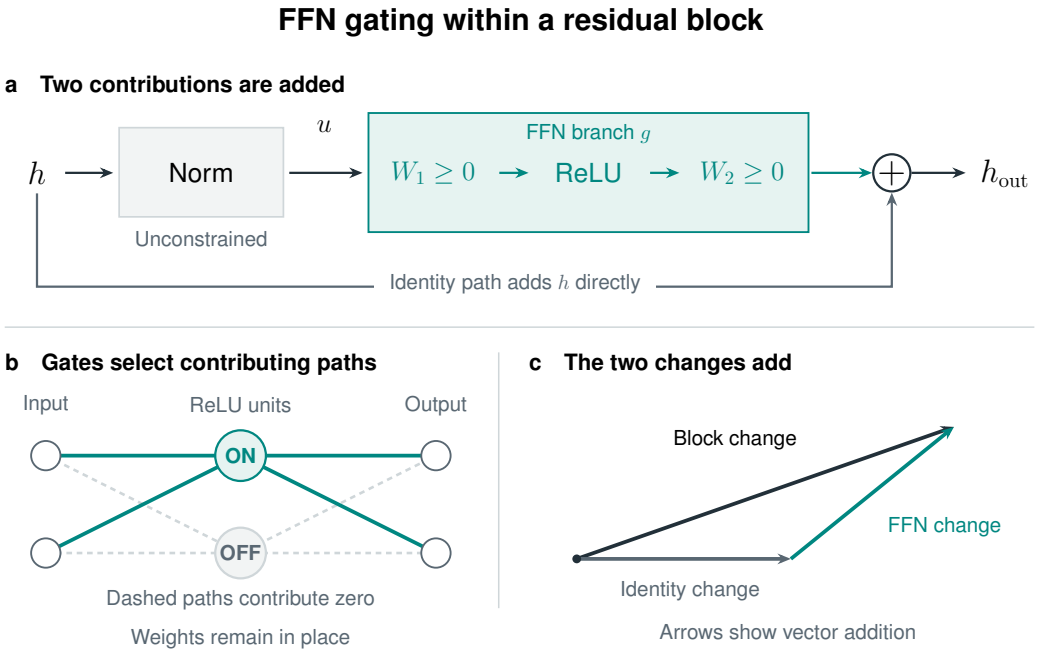
This statement concerns coordinatewise decreases of the FFN branch input. Token substitutions and changes propagated through normalization need not produce such decreases.

B.7 Local Mechanism and Scope

The preceding results isolate properties of an FFN branch at fixed weights. They do not show that constrained models contain more saturated units, have smaller branch-Jacobian norms, or produce smaller input gradients under token substitutions. Suppressing one branch contribution can increase the complete gradient norm by removing cancellation with other terms. Likewise, multiplying a HotFlip gradient by a positive scalar leaves its candidate ranking unchanged.

The analysis therefore identifies conditions under which selected FFN paths contribute less locally; it does not establish the mechanism behind the observed attack results. Measuring activation occupancy, branch-Jacobian norms, effective rank, and their relationship to selected substitutions would be required to test that mechanism directly.

Figure 3: **FFN gating and the residual response to a perturbation.** (a) The T5 FFN sublayer computes $h_{\text{out}} = h + g(N(h))$, where N is normalization and $g(u) = W_2\,\text{ReLU}(W_1 u)$ has nonnegative weights. (b) Solid paths contribute to the local branch Jacobian through an active ReLU; dashed paths contribute zero through an inactive ReLU. At a differentiable input, $J_g = \sum_j d_j W_2[:, j]W_1[j, :]$, with $d_j \in \{0, 1\}$. Nonnegative W_1 keeps inactive units off under coordinatewise decreases of the branch input, as in Lemma B.1. (c) The vector diagram depicts the exact identity $\Delta h_{\text{out}} = \Delta h + \Delta g$, where $\Delta g = g(N(h + \Delta h)) - g(N(h))$. Arrow directions and lengths are schematic. Locally, $\Delta g \approx W_2 D W_1 J_N \Delta h$. Removing inactive-unit terms need not reduce the norm of the complete response.



C T5 Experimental Details

C.1 Training and Evaluation Protocol

Both T5 arms start from the same pretrained `T5-small` checkpoint. The model has 6 encoder and 6 decoder layers, hidden dimension 512, 8 attention heads, and FFN width 2048. The constrained arm applies the softplus parameterization to $W_{\text{i}} \in \mathbb{R}^{2048\times512}$ and $W_{\text{o}} \in \mathbb{R}^{512\times2048}$ in all 12 blocks. This affects approximately 25M of 60.5M parameters, or 42%.

Both arms use AdamW with learning rate 5×10^{-5}, weight decay 0.01, batch size 4, and gradient clipping at norm 1.0. They train for 7 epochs with linear learning-rate decay after warmup. Warmup occupies 15% of training steps for the constrained model and 10% for the baseline; we did not ablate this difference. All downstream evaluation uses the checkpoint with the lowest validation loss.

Training uses DialogSum, HighlightSum, and arXiv abstract datasets, with 242,774 examples in total. Their validation splits supply 8,296 examples. Evaluation uses the full CNN/DailyMail test set of 11,490 examples and 11,333 XSUM examples. The loader drops one XSUM example with an empty field. Neither evaluation dataset is used for fine-tuning, so these evaluations test transfer to held-out summarization domains. The standard comparator is the single pretrained checkpoint without this fine-tuning.

Generation uses beam search with 4 beams, a length penalty of 1.2, and output lengths between 10 and 80 tokens. No-repeat n-gram blocking uses $n = 3$. These settings remain fixed across models and datasets. We compute ROUGE-1, ROUGE-2, and summary-level ROUGE-L (`rougeLsum`) F_1 with the `rouge-score` library and Porter stemming. Within each seed, 1,000 bootstrap resamples provide 95% confidence intervals. Per-example paired t-tests compare the baseline and constrained models, with Bonferroni correction across the three ROUGE variants. Effect sizes use Cohen's d.

C.2 Seeds, Statistical Tests, and Compute

The T5 pipeline uses five training seeds: 42, 1337, 2024, 8888, and 12345. Each seed controls randomness in Python, NumPy, and PyTorch. Summarization quality, training dynamics, and the probe diagnostic are summarized across these five seeds. The pretrained standard model is not an independent training replicate for each seed.

The principal HotFlip result uses seed 2024 and 200 examples. An earlier seed-42 evaluation uses 100 examples. Archived universal-trigger runs cover seeds 1337, 2024, 8888, and 12345; seed 42 was not evaluated under that protocol. Thus the five-seed clean-quality evaluation does not imply five-seed attack replication.

Paired tests of seed-level ROUGE-L means give $t = 14.9$ on CNN/DailyMail and $t = 13.4$ on XSUM, each with 4 degrees of freedom. These accompany paired per-example tests within each seed and separate variation over examples from variation across training runs. For T5 HotFlip, only aggregate counts and loss summaries remain available. We therefore report descriptive Wilson intervals for success rates and do not treat archived independent-samples loss tests as paired inference.

Experiments use deterministic algorithms where available. T5 runs use NVIDIA A100 GPUs with 40GB or 80GB memory, H200 GPUs with 141GB, and RTX PRO 6000 GPUs with 96GB under SLURM. The reported environment uses PyTorch with CUDA 11.8 on A100 or CUDA 12.8 on H200 and RTX PRO 6000, and Transformers 4.30 or later.

C.3 HotFlip Protocol

Our HotFlip variant (Ebrahimi et al., 2018) computes gradients once per input. It selects the five positions with the largest gradient norms and chooses replacements by their dot products with the fixed gradients. The attack protects the first position and masks pad, end-of-sequence, and beginning-of-sequence positions during position selection. Replacement selection excludes pad, end-of-sequence, and the current token.

The attacked sequence is decoded and re-encoded before evaluation. This round trip means that the final token distance need not equal the selected-edit budget.

We report mean relative loss increase, mean absolute loss increase in nats, clean and attacked loss, and the fraction exceeding a 10% relative-loss threshold. Absolute increments avoid normalization by each model's clean loss but do not remove differences in model quality. T5 success-rate intervals are descriptive Wilson intervals from archived counts. Paired per-example HotFlip losses are unavailable, so we make no paired significance claim from the archived independent-samples tests.

C.4 Universal-Trigger Protocol

Universal adversarial triggers (Wallace et al., 2019) are input-independent token sequences optimized to increase loss when prepended to an article. The archived T5 search uses five-token triggers, five random restarts, and 100 coordinate-search iterations. Candidates come from SentencePiece ids 1000–4999, augmented with a short hand-picked list of negation and discourse words, punctuation strings, and numerals. Example words include *not*, *never*, and *ignore*.

Each iteration resamples three trigger positions. At each position, it scores 50 random candidates on a random 30-example subset. The trigger with the highest mean loss on the full optimization set is retained. Optimization uses 1,000 held-out CNN/DailyMail validation examples. Evaluation uses 1,500 disjoint test examples, with cross-model transfer scored on the first 100.

In these archived runs, the trigger is prepended before left truncation to the 512-token encoder budget. A sufficiently long article can therefore lose the trigger, potentially making attacked and clean inputs identical. A later code correction reserves room for the trigger before insertion, but no verified rerun of that correction is included here. We therefore use the archived T5 trigger results only to document this protocol limitation.

Table 5: Training dynamics, reported as mean ± s.d. over five training seeds. Both models train for 7 epochs.

Model	Train loss (ep. 1)	Train loss (ep. 7)	Val. loss (ep. 1)	Val. loss (ep. 7)
Baseline	2.94 ± 0.00	2.40 ± 0.00	2.43 ± 0.00	2.21 ± 0.00
Monotone	4.99 ± 0.02	2.73 ± 0.00	2.92 ± 0.01	2.51 ± 0.00

Table 6: Summarization quality on the full XSUM test set (n = 11,333). Values are ROUGE F_1 $\times 100$, mean ± s.d. over five training seeds.

Model	ROUGE-1	ROUGE-2	ROUGE-L
Standard	20.1	2.9	13.5
Baseline	20.1 ± 0.0	3.1 ± 0.0	13.1 ± 0.0
Monotone	19.3 ± 0.1	2.9 ± 0.0	12.8 ± 0.0

C.5 Semantic-Order Diagnostic

The templated corpus uses 13 semantic axes, including severity, safety constraints, certainty, and urgency. Each axis is instantiated on 18 topics as a four-level chain. The resulting 702 adjacent-level prompt pairs are split by chain into 507 probe-fitting and 195 evaluation pairs.

For each model and pooling choice, we fit 64 probe directions on final-layer representations. Each direction is an ℓ_2-regularized logistic-regression readout trained to distinguish stronger from weaker prompts on a bootstrap resample of the fitting pairs. The directions form the rows of A. They are correlated readouts of one ordering signal rather than 64 independent semantic axes.

At every layer, we measure the fraction of coordinates satisfying

$$Ah_{\text{weak}} \leq Ah_{\text{strong}}$$

on the evaluation pairs. We evaluate mean and last-token pooling and compute bootstrap confidence intervals. The probes are fitted post hoc and play no role in training.

C.6 Additional T5 Results

Training dynamics. Table 5 reports per-epoch training and validation losses for both models, summarized over five training seeds. The constrained model recovers rapidly from its initial sign-changing conversion, reducing training loss from 4.99 to 3.05 over its first two epochs before entering slower convergence.

XSUM. Table 6 reports clean summarization performance on the full XSUM test set.

HotFlip on seed 42. An earlier evaluation of the seed-42 checkpoints on 100 CNN/DailyMail test examples gave mean relative loss increases of 5.2%, 16.2%, and 21.3%, and success rates of 19%, 63%, and 69%, for the monotone, baseline, and standard models, respectively. This smaller evaluation has the same ordering as the seed-2024 result in Table 2.

Generation length. Both fine-tuned models produce longer CNN/DailyMail summaries than the pretrained model. Five-seed mean lengths are 75.4 tokens for the baseline and 74.9 for the monotone model, compared with 62.3 for the standard model and 78.3 for the references. The monotone model has a larger length standard deviation than the baseline, 9.5 versus 6.6 tokens; the standard model has 12.0. Brevity penalties are similar for the monotone and baseline models, at 0.955 and 0.962. These statistics show similar average output lengths without establishing the absence of length bias.

C.7 Separating Initialization from the Constraint

The conversion changes both the allowable weights and their initialization. We train two additional `T5-small` controls at seed 42 under the protocol of Appendix C.1.

The *sign-frozen* arm uses

$$W = \text{sign}(W_{\text{pre}}) \odot \phi(V).$$

Table 7: Attribution ablation on `T5−small` at seed 42 after 7 epochs.

Arm	Parameterization	Constraint	Val. loss (ep. 1)	Val. loss (ep. 7)
Baseline	W (pretrained)	none	2.43	2.21
Sign-frozen	$\text{sign}(W_{\text{pre}}) \odot \phi(V)$	fixed signs	2.51	2.27
Abs-init-free	$V,\ V_0 = \lvert W_{\text{pre}} \rvert + \epsilon$	none	2.91	2.51
Monotone	$\phi(V)$, V_0 as in Section 2	$W \geq 0$	2.92	2.51

It preserves pretrained signs while using the same magnitude parameterization as the monotone arm. This is a fixed-sign constraint rather than a monotonicity constraint.

The *abs-init-free* arm uses unconstrained weights initialized at

$$|W_{\text{pre}}| + \epsilon.$$

Its initialization differs slightly from the implemented softplus initializer in Section 2.

Table 7 shows that sign-frozen training ends within 0.06 nats of the baseline, whereas abs-init-free reaches the same rounded validation loss as the monotone model. This single-seed comparison supports sign-changing initialization as a substantial contributor to the clean-loss gap at this training budget. Because neither control has attack results, the experiment does not isolate persistent nonnegativity as the cause of the smaller HotFlip loss increment.

D Pythia-1.4B Experimental Details

D.1 Constraint Scopes and Training

Pythia-1.4B (Biderman et al., 2023) is a 24-layer GPT-NeoX model with hidden dimension $d = 2048$. Each block contains an attention output projection and FFN projections

$$W_{\text{in}} \in \mathbb{R}^{4d \times d}, \qquad W_{\text{out}} \in \mathbb{R}^{d \times 4d}.$$

All constrained scopes apply the softplus parameterization in the native hidden-state basis across every layer.

We consider three historical placements:

- **MLP-in**: constrain W_{in} only;
- **MLP-both**: constrain W_{in} and W_{out}, matching the T5 scope;
- **MLP-in + Attn-out**: constrain W_{in} and the attention output projection.

Only MLP-both constrains every FFN weight matrix. The other scopes serve as placement ablations.

Pythia retains GELU for compatibility with the pretrained architecture, so Proposition 2.3 does not apply. The historical scope screen uses seed 42. MLP-in uses the full training budget described below, while MLP-both and MLP-in + Attn-out use 30,000 training samples at sequence length 1024, with 3 baseline epochs and 4 constrained epochs. Because these scopes receive unequal recovery budgets, we do not use the screening results as a controlled comparison of constraint placement.

Full-budget training uses the first 100,000 documents from the uncopyrighted subset of the Pile (Gao et al., 2020), split 90%/10% for training and validation. Sequence length is 2048. Both arms use AdamW with weight decay 0.01, gradient clipping at norm 1.0, and an effective batch size of 32 sequences, obtained from 8 sequences per step and 4 accumulation steps. Training uses bfloat16 weights and gradient checkpointing.

The baseline trains for 5 epochs at learning rate 10^{-5} with 10% warmup. The constrained model trains for 10 epochs at 5×10^{-5} with 20% warmup. Both use linear decay after warmup and retain the checkpoint with lowest validation perplexity. The larger constrained budget is intended to aid recovery from sign-changing initialization and is not a matched-budget comparison.

The reported MLP-both results use runs 42, 1337, and the completed replacement run 2024. The replacement run completed ten epochs and reached best validation perplexity 38.674. Available records do not establish independently retrained baseline models.

Table 8: Recorded Pythia constraint-scope screening design at run label 42. Historical performance values are omitted because matching raw evaluation exports are unavailable.

Scope	Constrained matrices	Recovery budget
MLP-in	W_{in}	Full
MLP-both	$W_{\text{in}}, W_{\text{out}}$	Reduced
MLP-in + Attn-out	W_{in}, attention output	Reduced

Universal-trigger optimization, trigger evaluation and transfer, and the probe diagnostic use bfloat16 models. HotFlip, substitution transfer, gradient-free controls, and historical held-out perplexity evaluation use float32. All Pythia experiments run on NVIDIA A100 GPUs with 80GB memory under SLURM.

D.2 Attack Objective and Protocols

Loss convention. The archived objective is mean per-token causal-LM NLL of the perturbed sequence. Inserted and substituted tokens are themselves prediction targets. A loss increase can therefore reflect their direct surprisal as well as effects on the remaining text.

This differs from T5, which scores a fixed reference summary given an edited source. Identical scoring across Pythia models does not isolate preservation of unchanged continuation targets because the models can assign different probabilities to the edited tokens. A corrected comparison would score the same unchanged targets under clean and attacked inputs while masking modified positions.

HotFlip. Fixed-gradient HotFlip attacks 200 texts with up to 10 substitutions selected by unrestricted vocabulary argmax. The success threshold is a 15% relative loss increase.

Universal triggers. Universal-trigger search uses ten-token triggers, three restarts, and 50 coordinate-ascent iterations. It optimizes on 80 texts and evaluates on 120 disjoint texts. Each position and sweep samples 20 candidates from a fixed pool of 100 token ids. These are the lowest-numbered ids in a candidate set containing vocabulary ids 1000–4999 and a short list of negation and punctuation tokens.

The retained historical trigger-transfer results use 900 held-out texts and earlier trigger exports. They are separate from the 120-text standalone evaluations.

D.3 Transfer and Gradient-Free Controls

Substitution transfer replays HotFlip edits from one model on the other, producing the full crafted-on $\times$ evaluated-on matrix.

Random substitution uses the same number of edits at uniformly random positions. Replacement tokens are sampled uniformly from the restricted candidate vocabulary used for trigger search; neither model selects these replacements.

The gradient-free query attack chooses 10 positions at random. At each position, it retains the best of 20 candidate tokens according to the target model's loss. This greedy search uses 50 texts and

$$10 \times 20 + 2$$

forward passes per example.

Together, transfer, random substitution, and query search test sensitivity beyond gradient-guided optimization. Their finite candidate pools and shared unmasked objective do not exclude gradient masking or establish robustness to stronger attacks.

D.4 Constraint-Scope Screening

Table 8 records the historical scope-screening design. Matching raw screening and clean-perplexity exports are absent from the available evidence bundle, so we omit historical performance summaries and draw no quantitative conclusion about a preferred constraint placement.

Table 9: Archived Pythia universal-trigger results on 120 texts. NLL columns show clean → attacked loss. Each increase column lists baseline → constrained. The objective includes trigger tokens as prediction targets.

Run label	Baseline NLL	Constrained NLL	NLL increase	ΔNLL
42	$2.007 \rightarrow 2.327$	$3.782 \rightarrow 4.115$	$15.9\% \rightarrow 8.8\%$	$0.320 \rightarrow 0.332$
1337	$2.008 \rightarrow 2.338$	$3.948 \rightarrow 4.267$	$16.4\% \rightarrow 8.1\%$	$0.330 \rightarrow 0.319$
2024	$2.007 \rightarrow 2.361$	$3.773 \rightarrow 4.099$	$17.6\% \rightarrow 8.6\%$	$0.353 \rightarrow 0.326$
Mean	$2.007 \rightarrow 2.342$	$3.834 \rightarrow 4.160$	$16.6\% \rightarrow 8.5\%$	$0.334 \rightarrow 0.326$

Table 10: Pythia universal-trigger transfer on 900 texts. Relative NLL increase is mean ± sample s.d. across run labels 42 and 1337 under the legacy unmasked objective.

Trigger learned on	Evaluated on baseline	Evaluated on constrained
Baseline	$17.6 \pm 0.8\%$	$6.2 \pm 0.4\%$
Constrained	$13.3 \pm 0.7\%$	$8.9 \pm 0.5\%$

D.5 Interpreting Transfer and Gradient-Free Controls

The HotFlip transfer rows in Table 4 use runs 42 and 1337. Random and query controls use all three runs. Each control compares the two models on the same evaluated texts, but query search uses 50 examples and should not be interpreted as a matched-sample comparison with the 200-example attacks. Reported degradation averages per-example loss ratios and therefore need not equal the ratio of the displayed mean losses.

Smaller relative increments partly reflect the constrained model's higher clean NLL. Absolute increments remove this division but still depend on the model and attacked objective. In every row of Table 4, the constrained model has higher attacked NLL than the baseline.

Moreover, the legacy loss includes substituted tokens as targets, and query search optimizes that same loss. A model assigning a less concentrated distribution can incur a smaller penalty on unusual replacements without better preserving predictions on the original text. The controls therefore broaden the description of the observed sensitivity but do not resolve the objective confound.

D.6 Universal-Trigger Transfer

Table 9 gives the 120-text Pythia trigger evaluations for all three runs. Relative NLL increase is smaller for the constrained model, while the mean absolute increase is nearly unchanged. Clean and attacked losses remain higher for the constrained model.

Table 10 retains the historical 900-text replay for runs 42 and 1337. Its trigger exports and evaluation set differ from those used in Table 9, so diagonal entries are not direct replications of the standalone results.

Under the archived objective, the baseline trigger has a smaller relative effect on the constrained model than the constrained model's own trigger does. This transfer pattern does not remove the inserted-target confound.

E Result Provenance and Interpretive Scope

E.1 Result Provenance and Protocol Corrections

The initial validation bundle was exported on 16 September 2026. The final update retrieved completed Pythia outputs and deployed evaluator source on 24 September. The standalone HotFlip and universal-trigger tables now include runs 42, 1337, and the replacement run 2024.

The replacement completed training and all four attack stages. Its standalone HotFlip success rate is 43.0%, compared with 44.5% when replaying cached substitutions from the older checkpoint. These are different attack evaluations and are not interchangeable.

The deployed Pythia evaluators score modified tokens as prediction targets. A masking correction exists in the local source but is absent from the retrieved cluster version. The reported results therefore retain the unmasked-objective interpretation. This source check takes precedence over earlier run summaries that described the correction as deployed.

The seed-2024 substitution-transfer stage reused completed attack-crafting caches. We therefore exclude its transfer cells from comparisons requiring attacks crafted on the replacement checkpoint. Random edits are independent of the checkpoint, so the replacement random and query evaluations remain in their respective three-run summaries.

The universal-trigger transfer export combines earlier baseline-target cells with new constrained-target cells. Trigger ids differ across columns within a nominal source row, and the shared sample-count field does not describe the older cells. We therefore exclude this mixed export and retain only the historical 900-text transfer results from runs 42 and 1337.

The available evidence contains five T5 clean-evaluation seeds, four archived T5 trigger seeds, one primary T5 HotFlip seed with 200 examples, and an earlier seed-42 HotFlip evaluation with 100 examples. Pythia run labels do not establish independent baseline training. The replacement model's validation perplexity is available from its training history, while older held-out perplexity and scope-screening summaries lack corresponding evaluator exports.

The final evidence snapshot preserves the retrieved Pythia result files, source, and completion logs. These improve traceability without establishing an immutable executable revision for every historical run. Paired per-example T5 HotFlip losses remain unavailable; aggregate counts support descriptive success-rate intervals but cannot recover paired loss tests or measurements under corrected scoring.

E.2 Interpretive Scope

T5. Nonnegative ReLU FFN branches preserve the coordinatewise order of their own inputs. Under the tested conversion and fine-tuning procedure, T5 shows substantially smaller HotFlip-induced loss increments while retaining higher clean and attacked loss and lower summarization quality. The current experiments do not isolate persistent nonnegativity from the loss of pretrained signs, optimization changes, or the accompanying quality difference.

Semantic order and local sensitivity. Branch monotonicity does not imply semantic order preservation, bounded sensitivity, or a low-rank Jacobian. The semantic-order diagnostic makes the distinction between coordinatewise and semantic order empirically visible. Likewise, the saturation analysis gives conditions under which selected branch contributions become small but does not show that the trained constrained models saturate more often or that saturation changes HotFlip's selected substitutions.

Pythia. The Pythia study changes activation, task, model scale, optimizer schedule, and recovery budget relative to T5. Its nonnegative-weight GELU FFNs are not exactly monotone, and its unmasked objective mixes the surprisal of modified tokens with their effects on other predictions. Random, transfer, and query-based evaluations broaden the tested procedures while retaining these interpretive limits.

Overall, the experiments characterize how a local nonnegative-weight constraint changes measured token-level sensitivity under the reported procedures. They do not establish a general adversarial-robustness guarantee for the full Transformer.