machine learning & knowledge extraction

MDPI

*Article*

# Efficient Certified Robustness Assessment for Transformer Models via Verifier-Aware Perturbation-Position Scheduling

**Vafali Soltanmuradov [1,2], Riccardo Berta [1], Luca Lazzaroni [1,*], David Martín Gómez [2], Alessandro Pighetti [1] and Francesco Bellotti [1]**

[1] Department of Electrical, Electronic and Telecommunication Engineering (DITEN), University of Genoa, Via Opera Pia 11a, 16145 Genoa, Italy; vafali.soltanmuradov@edu.unige.it (V.S.); riccardo.berta@unige.it (R.B.); alessandro.pighetti@edu.unige.it (A.P.); francesco.bellotti@unige.it (F.B.)
[2] Department of Systems Engineering and Automation, Universidad Carlos III de Madrid, C. Butarque 15, 28911 Leganes, Spain; dmgomez@ing.uc3m.es
* Correspondence: luca.lazzaroni@unige.it

**Abstract**

Certified robustness provides guarantees that a model's prediction is stable within a specified perturbation region, but certified-radius assessment for Transformer text classifiers is computationally expensive because sentence-level certificates require repeated radius searches over multiple perturbation positions. Existing approaches primarily improve the precision or runtime of individual verifier calls, whereas repeated position-level search is an additional source of computational cost. We address this bottleneck with verifier-aware perturbation-position scheduling: selected positions undergo full radius search to obtain a candidate sentence-level radius, skipped positions are validated at that radius, and only failed validations receive fallback search. Thus, scheduling reduces repeated search effort but preserves verifier coverage of all eligible perturbation tasks. Experiments include an expanded 100-example SST-2 evaluation and a 30-example Yelp evaluation, together with one-token and two-token embedding-space perturbations, early-exit verifier cascades, and scheduling ablations, to assess certificate preservation and verification efficiency across different settings. On the expanded SST-2 all-position evaluation, the scheduler reproduces the exhaustive sentence-level radius on all examples, while substantially reduces the number of full radius searches. Additional sensitivity experiments show similar scheduling behavior under the evaluated depth and longer-sequence settings. The presented results suggest that perturbation-position scheduling provides a verifier-compatible way to reduce repeated work in certified robustness assessment. Next research steps should expand the evaluation by addressing large pretrained language models, alternative verification backends, and broader discrete text perturbation settings.

**Keywords:** adversarial robustness; AI security; certified robustness; natural language processing; neural network verification; perturbation-position scheduling; transformers; verifier-aware scheduling

Check for updates

Academic Editors: Muhammad Zeeshan Baig, Zulfiqar Ali and Muhammad Imran

Received: 6 August 2026
Revised: 12 September 2026
Accepted: 18 September 2026
Published: 20 September 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

## 1. Introduction

The robustness of Transformer text classifiers depends on the stability of their predictions under admissible input perturbations, which cannot be inferred from average test accuracy alone. Although empirical adversarial evaluations in natural language processing (NLP) and related machine learning settings can reveal individual failures [1–4], they cannot establish stability over all inputs in a specified perturbation set. Certified

robustness addresses this limitation by lower-bounding the output margin over that set; a strictly positive lower bound guarantees that the predicted class cannot change [5,6]. Such guarantees are relevant to AI security because they complement attack-based evaluation with verifier-based evidence covering the entire perturbation region.

Certifying Transformer text classifiers is computationally difficult because perturbations to token embeddings propagate through self-attention [7]. Uncertain query, key, and value representations interact through dot products, softmax normalization, and weighted aggregation, creating dependencies across embedding dimensions and token positions. These nonlinear dependencies make simple interval bounds overly conservative [8].

Most research on Transformer robustness certification has therefore focused on making individual verifier calls tighter or faster. Existing methods introduce Transformer-specific attention bounds [8], stronger abstraction and refinement methods [9], tighter softmax relaxations [10,11], linear relaxation-based perturbation analysis (LiRPA) bound tightening [12], parameterized abstractions [13], or specialized GPU execution [14]. Sentence-level certified-radius assessment introduces an additional source of cost that these methods do not directly target.

This paper investigates whether that repeated work can be reduced without changing the certificate accepted at sentence level: can exhaustive all-position assessment be recovered with fewer full radius searches, while still checking every eligible position? We answer this question with verifier-aware perturbation-position scheduling. Ranked positions are searched first, and their minimum certified radius becomes a sentence-level candidate. Every skipped position is then validated at that candidate radius; only a position that fails validation receives a full fallback search. Thus, ranking influences computational cost, whereas certificate acceptance continues to depend on verifier decisions for every eligible position.

Our method operates at a different level than verifier portfolios and checker cascades, as it schedules perturbation-position tasks rather than verifier configurations. Each radius query can still use the early-exit IBP→Forward→Backward cascade, while the outer scheduler determines which positions require full radius search and which require single-radius validation only. Thus, the cascade reduces the cost of individual queries, whereas perturbation-position scheduling reduces the number of full searches. Figure 1 illustrates this distinction.

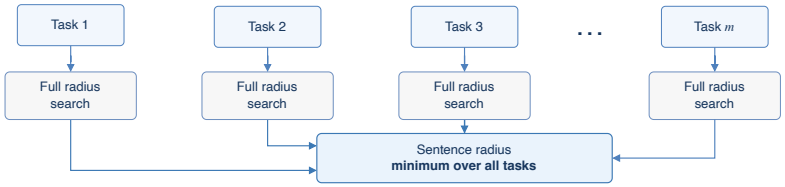
Our evaluation considers continuous norm-bounded perturbations of eligible token embeddings. These certificates do not directly cover arbitrary discrete synonym substitutions, insertions, deletions, or semantic text transformations, which require different perturbation sets and certification techniques [6,15–17].

The paper makes the following contributions:

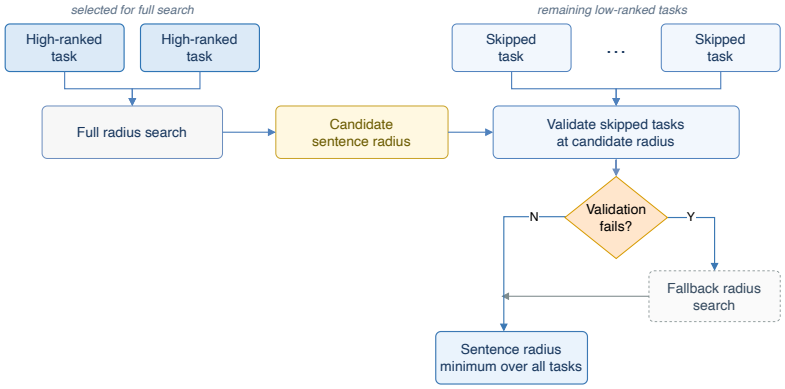
1. A verifier-agnostic formulation of repeated sentence-level robustness assessment. We formulate sentence-level certified-radius evaluation as a collection of perturbation-position tasks whose certified lower endpoints must be combined by taking their minimum. This formulation isolates repeated position-specific radius search as a source of cost distinct from the precision or runtime of an individual verifier call.
2. A coverage-preserving perturbation-position scheduler. We introduce a search–validate–fallback procedure in which ranked tasks propose a candidate sentence-level radius, every skipped task is checked at that radius, and validation failures receive full fallback search. Ranking therefore affects computational cost but not certificate acceptance. We establish verifier-conditional soundness for both one-token tasks and unordered two-token position pairs and characterize the resulting verification cost.
3. An empirical evaluation of scheduling effectiveness and sensitivity. We evaluate scheduling on SST-2 and Yelp, including a 100-example SST-2 all-position experiment

with sentence-level confidence intervals, one-token and two-token perturbations, ranking and fallback ablations, and verifier-stage analysis. We additionally study model depth using the same fixed SST-2 sentences and perturbation tasks in matched one- and two-layer Transformers, while the AGNews2 case study provides supporting evidence on an additional dataset using a two-layer model.

The empirical study was designed to address three research questions: RQ1, whether coverage-preserving perturbation-position scheduling can reproduce exhaustive sentence-level radii while reducing the number of full radius searches; RQ2, whether early-exit verifier staging can reduce expensive Backward calls while retaining the radii produced by Backward verification; and RQ3, how scheduling and cascade efficiency vary across datasets, task budgets, ranking heuristics, perturbation norms, and perturbation orders.




(**a**)




(**b**)

**Figure 1.** Conceptual overview of (**a**) exhaustive certified-radius evaluation and (**b**) verifier-aware perturbation-position scheduling.

## 2. Related Work

Neural network verification methods include complete and exact approaches based on satisfiability solving, mixed-integer optimization, branch-and-bound, and specialized exact certification [18–21], as well as scalable incomplete methods based on sound outer approximations such as IBP, convex relaxations, abstract interpretation, and LiRPA-style bound propagation [22–26]. Incomplete verifiers may return unknown, but every certificate they prove is sound.

Other efficiency-oriented verification work accelerates repeated or structured verification through massively parallel incomplete verifiers, split-constraint bound propagation, incremental verification, branch-and-bound ordering, and learned conflict reuse [27–31]. These methods improve how verification problems are solved internally; our work instead reduces repeated Transformer radius search work by scheduling verifier calls and perturbation tasks.

Transformer-specific verification primarily addresses the difficulty of obtaining useful bounds for a single verification problem. Shi et al. [8] introduced the first robustness-

verification method for Transformers, developing dedicated bounds for self-attention to handle cross-nonlinearity and cross-position dependencies. Their method substantially improves certification precision over naive interval propagation. However, this improvement concerns how a given perturbation specification is verified. In the sentence-level setting considered here, evaluating the certified radius over several eligible token positions still requires applying the verifier repeatedly while searching the radius for each position.

More recent methods improve the precision of these individual verification problems. Liu et al. [9] used ReLU-catalyzed abstraction refinement to tighten the bounds of attention dot products. Their evaluation shows substantially larger certified radii, particularly for deeper models, but also exposes the usual precision–efficiency trade-off of stronger refinement. GaLileo [11] targeted another source of approximation error by modeling dependencies among the inputs of the softmax relaxation. It reports certified radii up to 3.24 times larger than its baseline with similar running times and also supports multi-word perturbations. DROWN [12] instead tightened LiRPA-based verification through refined activation bounds; on Transformers it reports radii up to 4.68 times larger than its baseline, while requiring approximately twice the baseline runtime. These methods therefore improve the certificate obtained from an individual verification task, but do not reduce how many position-specific radius searches must be performed when the sentence-level certificate is defined by the minimum over perturbation positions.

Certified robustness in NLP is also studied under discrete text-edit threat models. Prior work certifies predefined word substitutions, synonym substitutions, randomized text transformations, and embedding or abstract-domain certificates for language models [6,15–17,32,33]. Our setting is different: we certify continuous norm-bounded perturbations to selected token embeddings using deterministic bound propagation. We therefore do not claim that an embedding-space certificate directly implies robustness to synonym substitution, character edits, insertions, deletions, or paraphrases.

A second line of work reduces verification cost by choosing among verification procedures rather than by changing their bounds. Runtime checker cascades [34] applied inexpensive checks before invoking an exact and more costly procedure, while verifier portfolios [35] exploit differences in solver performance across verification instances. CRV [36] similarly applies multiple verification methods progressively and stops when one of them certifies the input. These approaches reduce the cost of deciding a verification instance by choosing which checker, solver, or verifier to use. The scheduling unit in our method is different: the verifier can remain fixed, while the scheduler decides which perturbation-position tasks require a complete radius search and which can first be checked by a single validation query. Our inner IBP $\rightarrow$ Forward $\rightarrow$ Backward cascade follows the former idea; the outer perturbation-position scheduler is the contribution studied here.

A third complementary direction reuses verification work across related specifications. Fischer et al. [37] introduce Shared Certificates, which reuse intermediate symbolic abstractions when proof regions for different inputs or perturbations overlap. This can avoid reconstructing complete proofs for related specifications, and their experiments demonstrate the idea on image classifiers and several perturbation specifications. Our method does not construct or reuse proof templates. Instead, it exploits the structure of the sentence-level objective. Since the final radius is the minimum over perturbation-position tasks, selected tasks can propose a candidate radius and the remaining tasks only need to certify that candidate unless validation fails. Proof reuse could therefore reduce the cost of individual search or validation calls, while perturbation-position scheduling reduces the number of tasks that require full radius search.

Systems acceleration addresses yet another level of the computation. Faith [14] uses semantic-aware graph transformations, verification-specific GPU kernels, and autotuning

to accelerate Transformer bound propagation, reporting 2.1–3.4 times faster verification than the compared frameworks. Such acceleration is directly complementary to our method: it can make each verification query cheaper, but it does not remove the repeated position-specific radius searches that arise when a sentence-level certificate is obtained by taking the minimum across eligible positions.

Summarizing, prior work attacks three major sources of verification cost: the precision of the bounds within a verifier call, the runtime or selection of the verifier used for a verification instance, and the reuse of proof information across related specifications. However, they fail to address the bottleneck due to the fact that sentence-level certified-radius assessment creates a collection of perturbation-position tasks, each of which would conventionally receive a full radius search before their radii are combined by a minimum. Our perturbation-position scheduling proposal reduces this repeated search work while retaining verifier coverage of every eligible task. Our approach thus is a complement to tighter Transformer bounds, faster verification systems, verifier cascades and portfolios, and proof-reuse methods, as summarized in Table 1.

**Table 1.** Contextual comparison with complementary verification advances.

| Work | Optimized Level | Reported Evidence | Position-Level Search | Relation to Scheduler |
|---|---|---|---|---|
| Liu et al. [9] | Verifier call precision | Up to 3.6× larger radii | Not targeted | Can serve as backend |
| GaLileo [11] | Softmax-bound precision | Up to 3.24× larger radii | Not targeted | Can serve as backend |
| DROWN [12] | LiRPA-bound precision | Up to 4.68× larger radii | Not targeted | Can serve as backend |
| Faith [14] | Verifier call runtime | 2.1–3.4× faster calls | Not targeted | Can speed up calls |
| CRV [36] | Verifier method selection | Up to 90% runtime reduction | Not targeted | Schedules stages |
| Shared Certificate [37] | Proof reuse | Reduced repeated proof effort | Not targeted | Can reduce call cost |
| Ours | Position-level radius search | 81.4–91.4% fewer full searches; exact sentence agreement | Reduced | Search, validate, fallback |

This paper adds a scheduling layer for that missing level of the verification pipeline. The layer searches selected perturbation-position tasks, validates the remaining tasks at the resulting sentence-level candidate radius, and runs fallback search only for validation failures. It therefore targets repeated position-level radius search while preserving coverage of all eligible perturbation tasks relative to the underlying verifier.

## 3. Materials and Methods

This section defines the certification objective, the perturbation tasks, and the verifier interface used by the scheduler. We first state the one-token and two-token robustness properties. We then describe the early-exit verifier cascade used for radius queries, introduce the perturbation-position scheduling algorithm, and state the verifier-conditional guarantees and cost model.

### *3.1. Verification Objective*

Let $f$ be a Transformer classifier that outputs class logits, let $x_0$ be the clean input embedding sequence, and let $y = \arg\max_c f_c(x_0)$ be the clean predicted label. In our evaluation, we use correctly classified examples, so the predicted label also equals the ground-truth label. For a one-token perturbation at position $i$, we define the perturbation set as

$$B_p^i(x_0, \epsilon) = \left\{ x : \|x_i - x_{0,i}\|_p \leq \epsilon,\ x_\ell = x_{0,\ell} \text{ for all } \ell \neq i \right\}. \quad (1)$$

Only the selected token embedding is perturbed; all other token embeddings remain fixed. For each competing class $j \neq y$, define the margin

$$m_j(x) = f_y(x) - f_j(x). \tag{2}$$

We use an acceptance threshold $\tau \geq 0$ for verifier decisions. A radius $\epsilon$ is certified for position $i$ at threshold $\tau$ if

$$\min_{j \neq y} \min_{x \in B_p^i(x_0, \epsilon)} \big(f_y(x) - f_j(x)\big) \geq \tau. \tag{3}$$

When $\tau > 0$, this gives strict label preservation without relying on classifier tie-breaking. When $\tau = 0$, the condition proves a nonnegative margin certificate: no competing class has a larger logit than $y$, but exact ties are not resolved unless the classifier's deterministic tie-breaking rule is included in the formal specification. Our experiments follow the base verifier implementation's default nonnegative margin rule, corresponding to $\tau = 0$. Accordingly, the reported experimental certificates should be read as nonnegative margin certificates under the base verifier's acceptance rule. A strict label preservation certificate would require either $\tau > 0$ or an explicitly modeled deterministic tie-breaking rule.

A verifier computes sound lower bounds on the worst-case margin. If the lower bound is at least $\tau$ for every competing class, the radius is certified at threshold $\tau$; otherwise, the verifier returns unknown.

For a two-token perturbation task with unordered position pair $q = \{i_1, i_2\}$, we use the product perturbation set

$$B_p^q(x_0, \epsilon) = \Big\{ x : \|x_{i_1} - x_{0,i_1}\|_p \leq \epsilon,\ \|x_{i_2} - x_{0,i_2}\|_p \leq \epsilon,\ x_\ell = x_{0,\ell} \text{ for all } \ell \notin q \Big\}. \tag{4}$$

Thus, in the two-token $L_2$ experiment, each selected token embedding has its own $L_2$ radius $\epsilon$; this is not a single joint $L_2$ constraint over the concatenation of the two embedding vectors. The set includes cases where one or both selected embeddings remain unchanged, since zero perturbation is allowed within each ball.

For two-token verification, the sentence-level radius is the minimum over all unordered valid position pairs:

$$\epsilon_{\text{full}}^{(2)} = \min_{\{i_1, i_2\} \subseteq P} \epsilon_{\{i_1, i_2\}}, \tag{5}$$

where $\epsilon_{\{i_1, i_2\}}$ is the certified radius for the pairwise perturbation set in Equation (4). The same cascade acceptance rule and radius search procedure are used for pair tasks.

### 3.2. Early-Exit Verifier Cascade

The specialized Transformer verifier provides several sound bound modes with different cost-precision tradeoffs. We use three of them as cascade stages: IBP [22,23], Forward bound propagation, and Backward bound propagation, following the Transformer embedding-space verification setting of prior work [8]. In the rest of the paper, we refer to these stages as IBP, Forward, and Backward for brevity. We apply them in increasing cost order:

$$\text{IBP} \rightarrow \text{Forward} \rightarrow \text{Backward}. \tag{6}$$

For each candidate radius during binary search, the cascade first tries IBP. If IBP certifies the radius, the candidate is accepted and the more expensive methods are skipped. If IBP cannot certify the candidate radius, the cascade tries the Forward bound mode. The Backward stage is used only when both cheaper stages cannot certify the candidate radius.

Let $Q$ denote a perturbation-position task: for one-token verification, $Q = \{i\}$, and for two-token verification, $Q = \{i_1, i_2\}$. Let $B_p^Q(x_0, \epsilon)$ denote the corresponding perturbation

set, using Equation (1) for one-token tasks and Equation (4) for two-token tasks. Let $L^{Q}_{v,j}(\epsilon)$ denote the lower bound produced by verifier stage $v$ for task $Q$ and competing class $j$, where

$$L^{Q}_{v,j}(\epsilon) \leq \min_{x \in B^{Q}_{p}(x_0,\epsilon)} m_j(x). \tag{7}$$

Each stage is sound at threshold $\tau$: if its lower bound is at least $\tau$ for every competing class, then the candidate radius satisfies Equation (3). The cascade evaluates stages lazily in the order IBP $\rightarrow$ Forward $\rightarrow$ Backward and accepts a candidate radius as soon as one stage certifies all competing classes:

$$\exists v \in \{\text{IBP}, \text{Forward}, \text{Backward}\} \quad \text{s.t.} \quad L^{Q}_{v,j}(\epsilon) \geq \tau, \quad \forall j \neq y. \tag{8}$$

Because acceptance requires a complete certificate from one sound verifier stage, every radius accepted by the cascade is sound.

### 3.3. Verifier-Aware Perturbation Scheduling

For each evaluated perturbation task, either a one-token position or a two-token pair, we compute a certified lower endpoint through repeated verifier queries over the perturbation radius. Following the general binary refinement principle used for certified-radius estimation in prior robustness work [38], once the binary refinement phase is reached, a certified lower endpoint and an uncertified upper endpoint define the search interval.

$$m_t = \frac{\ell_t + u_t}{2}, \qquad (\ell_{t+1}, u_{t+1}) = \begin{cases} (m_t, u_t), & \text{if } m_t \text{ is certified,} \\ (\ell_t, m_t), & \text{otherwise.} \end{cases} \tag{9}$$

Each certification query is performed using the early-exit verifier cascade described above. After the binary refinement phase, the returned certified radius is the final certified lower endpoint. Two search-resolution settings are considered in this work. The main experiments use a fixed setting of $T = 4$ binary refinement steps, whereas the additional longer-sequence sensitivity experiment uses an adaptive tolerance-based refinement strategy that continues until the certified and uncertified radius interval is below a predefined tolerance. Before binary refinement, the implementation expands or contracts the initial radius to obtain the search interval; the complete procedure, including possible termination at the maximum radius, is given in Algorithm A1 in Appendix A.

For a sentence with valid perturbable positions $P$, full one-token verification computes

$$\epsilon_{\text{full}} = \min_{i \in P} \epsilon_i, \tag{10}$$

where $\epsilon_i$ is the certified radius for perturbing token $i$. The minimum is used because the sentence-level certificate is determined by the weakest perturbable position.

To reduce cost, we rank positions using the gradient score

$$s_i = \|\nabla_{x_i} \mathcal{L}(f(x_0), y)\|_2, \tag{11}$$

and select the top-$k$ positions $S_k$. This ranking is only a proposal heuristic: large gradient norms indicate local sensitivity, but they do not provide a certificate.

The scheduling procedure then validates the proposed radius over all skipped positions. We first run full radius search on the selected positions and compute

$$\epsilon_{\text{cand}}(k) = \min_{i \in S_k} \epsilon_i. \tag{12}$$

For every skipped position $i \in P \setminus S_k$, we test whether the cascade certifies radius $\epsilon_{\text{cand}}(k)$ at that position. With fallback search, the returned radius is

$$\epsilon_{\text{sched}}(k) = \min\left(\epsilon_{\text{cand}}(k), \min_{i \in F_k} \epsilon_i\right), \tag{13}$$

where $F_k$ is the set of skipped positions that are not certified at $\epsilon_{\text{cand}}(k)$.

Algorithm 1 summarizes the perturbation-position scheduling procedure. Let $P$ be the set of valid perturbable token positions. For one-token scheduling, the task set is

$$\mathcal{Q} = \{\{i\} : i \in P\}.$$

For two-token scheduling, the task set is

$$\mathcal{Q} = \{\{i_1, i_2\} : i_1, i_2 \in P,\ i_1 < i_2\}.$$

One-token tasks are ranked by

$$s_{\{i\}} = s_i, \qquad s_i = \|\nabla_{x_i} \mathcal{L}(f(x_0), y)\|_2.$$

Two-token tasks $Q = \{i_1, i_2\}$ are ranked by

$$s_Q = s_{i_1} + s_{i_2}.$$

Ties are broken lexicographically by token position.

In our experiments, we use $\rho = \text{search}$, so tasks not certified at $\epsilon_{\text{cand}}$ are handled by fallback radius search rather than reported as unknown. When a skipped position is certified at $\epsilon_{\text{cand}}$, the algorithm records $\epsilon_{\text{cand}}$ as a certified lower endpoint for that position.

The same scheduling template applies to two-token perturbations by changing the scheduling unit from a single position to an unordered pair of valid perturbable positions. For two-token scheduling, the algorithm ranks all valid pairs, searches the selected top-$k$ pairs, validates every skipped pair at the candidate radius, and fallback searches skipped pairs that are not certified. The returned sentence-level pair radius is the minimum over selected and fallback-searched pairs, with validated skipped pairs covered at the candidate radius.

The following results are stated at the level of an abstract verifier oracle. They show that the scheduling procedure introduces no additional unsound acceptance decisions, assuming that each underlying verifier stage returns sound lower bounds under the chosen margin threshold. Our implementation uses floating-point CUDA computations and is not a proof-producing or directed-rounding verifier; therefore, the experimental certificates should be interpreted as conditional on the numerical soundness of the base verifier implementation.

**Lemma 1** (Verifier-conditional cascade soundness)**.** *Assume every cascade stage produces sound lower bounds at threshold $\tau$. If the early-exit cascade accepts radius $\epsilon$ for perturbation task $Q$, then the corresponding margin condition holds over $B_p^Q(x_0, \epsilon)$.*

**Proof.** If the cascade accepts radius $\epsilon$, then some verifier stage $v \in \{\text{IBP}, \text{Forward}, \text{Backward}\}$ satisfies $L_{v,j}^Q(\epsilon) \geq \tau$ for every $j \neq y$. By stage soundness, these lower bounds imply

$$\min_{j \neq y} \min_{x \in B_p^Q(x_0, \epsilon)} \big(f_y(x) - f_j(x)\big) \geq \tau. \quad \square$$

**Algorithm 1** Perturbation-position scheduling with early-exit verifier cascade.

**Require:** Input $x_0$, predicted label $y$, perturbation task set $\mathcal{Q}$, budget $k$
**Require:** Acceptance threshold $\tau$
**Require:** Verifier stages $\mathcal{V} = (\text{IBP}, \text{Forward}, \text{Backward})$
**Require:** Fallback mode $\rho \in \{\text{unknown}, \text{search}\}$
Rank tasks $Q \in \mathcal{Q}$ by score $s_Q$
Select $S_k \leftarrow$ top-$k$ ranked tasks
Run cascade radius search for each $Q \in S_k$
**if** any selected-task search returns unknown **then**
  **return** unknown
**end if**
Let $\epsilon_Q$ be the returned certified lower endpoint for each $Q \in S_k$
$\epsilon_{\text{cand}} \leftarrow \min_{Q \in S_k} \epsilon_Q$
$F_k \leftarrow \emptyset$
**for** each skipped task $Q \in \mathcal{Q} \setminus S_k$ **do**
  certified $\leftarrow$ false
  **for** each verifier stage $v \in \mathcal{V}$ **do**
    Compute sound lower bounds $L^Q_{v,j}(\epsilon_{\text{cand}})$ for all $j \neq y$
    **if** $L^Q_{v,j}(\epsilon_{\text{cand}}) \geq \tau$ for all $j \neq y$ **then**
      certified $\leftarrow$ true
      **break**
    **end if**
  **end for**
  **if** not certified **then**
    $F_k \leftarrow F_k \cup \{Q\}$
  **end if**
**end for**
**if** $F_k = \emptyset$ **then**
  **return** $\epsilon_{\text{cand}}$
**else if** $\rho =$ unknown **then**
  **return** unknown
**else**
  Run cascade radius search for each $Q \in F_k$
  **if** any fallback search returns unknown **then**
    **return** unknown
  **end if**
  Let $\epsilon_Q$ be the returned certified lower endpoint for each $Q \in F_k$
  **return** $\min\left(\epsilon_{\text{cand}}, \min_{Q \in F_k} \epsilon_Q\right)$
**end if**

**Theorem 1** (Verifier-conditional scheduling soundness)**.** *Let $\mathcal{Q} \neq \emptyset$ be the set of perturbation tasks, and let $1 \leq k \leq |\mathcal{Q}|$. Assume every verifier cascade acceptance decision, including each skipped-task validation decision, is sound at threshold $\tau$. Assume every invoked radius search either returns unknown or returns a certified lower endpoint at threshold $\tau$, and that the scheduling procedure returns a radius only when all selected-task and fallback searches return certified lower endpoints. Assume fallback search is run for every skipped task that is not certified at $\epsilon_{\text{cand}}$. Then the radius returned by perturbation-position scheduling is certified at threshold $\tau$ for every task in $\mathcal{Q}$. For one-token scheduling, this covers every valid perturbable position; for two-token scheduling, it covers every valid position pair.*

The proof is given in Appendix A.

### *3.4. Numerical Soundness*

Our scheduling algorithm is sound relative to the acceptance decisions returned by the underlying verifier. It introduces no additional acceptance rule: a candidate radius

is accepted only when an underlying verifier stage returns a certificate for all competing classes under the chosen threshold $\tau$, and skipped positions are either certified at the candidate radius or handled by fallback search. The implementation uses standard floating-point tensor computations on CUDA rather than directed rounding or an independently validated numerical kernel. Therefore, our claims are algorithmic and conditional on the numerical behavior of the base verifier implementation. In particular, the reported experiments use the base verifier's default nonnegative margin rule with $\tau = 0$; strict label preservation would require a positive threshold or an explicit tie-breaking specification.

Consequently, the experimental outputs should be interpreted as conditional verifier outputs rather than independently checked end-to-end formal certificates. The formal contribution of this paper is the scheduling layer: given verifier-stage decisions satisfying Equation (7), the scheduling procedure does not introduce additional unsound acceptance decisions. We do not provide a separate proof that the floating-point implementation of each abstract transformer satisfies the real-arithmetic lower-bound relation, nor do we emit proof certificates checked by a smaller trusted component. Adding directed rounding, conservative numerical error bounds, or proof-producing checks is important future work.

Additional radius search details, finite-resolution comparison, search saturation discussion, and the cost model are given in Appendix A.

### *3.5. Experimental Methodology*

#### 3.5.1. Datasets and Models

We evaluate SST-2 [39] and Yelp [40] under the embedding-space certification setting used in prior Transformer verification work [8,11,12]. For SST-2, we use the repository-provided labeled file `data/sst/test.txt`, since the official GLUE SST-2 test labels are not publicly available and ground-truth labels are required both to compute clean accuracy and to select correctly classified examples for certification. This labeled file contains 2210 raw examples; after the repository's binary sentiment filtering, 1821 labeled examples remain. The reported SST-2 clean accuracy is evaluated on these 1821 binary examples. Yelp is treated as a binary sentiment task, with labels mapped from $\{1, 2\}$ to $\{0, 1\}$. Its reported clean accuracy is evaluated on the repository-provided Yelp test split after preprocessing and binary label mapping. Text is preprocessed by the repository pipeline and tokenized with the lower-cased WordPiece tokenizer from `pytorch_pretrained_bert`.

For both datasets, the main model is a one-layer Transformer trained from scratch with hidden size 256, intermediate size 512, four attention heads, ReLU activation, and no-variance layer normalization. Training uses BertAdam with learning rate $10^{-4}$, batch size 32, weight decay 0.01, seed 0, and three epochs. The SST-2 checkpoint reaches accuracy 0.837322 on the 1821-example binary evaluation set (reported as 0.837), while the Yelp checkpoint reaches 0.914 on its repository-provided evaluation split. No-variance layer normalization subtracts the feature mean and applies the learned affine scale and bias, but omits division by the feature standard deviation; this follows the specialized Transformer verification setting of Shi et al. [8]. Dropout is disabled during verification.

To examine sensitivity to Transformer depth, we additionally perform a matched SST-2 experiment using one- and two-layer models. Both models use hidden size 256, intermediate size 512, four attention heads, ReLU activation, no-variance layer normalization, and the same three-epoch training protocol. The comparison uses the same fixed 10 correctly classified sentences and the same 115 valid perturbation-position tasks at both depths; the architectural change is therefore restricted to the number of Transformer layers. The one-layer and two-layer checkpoints obtain clean evaluation accuracies of 0.837322 and 0.83836, respectively.

#### 3.5.2. Perturbation Tasks and Example Selection

The main experiments use one-token perturbations with tokenized length at most 32. Unless otherwise stated, results use one-token $L_2$ perturbations. Valid perturbable positions are non-special, single-piece WordPiece tokens; we exclude `[CLS]`, `[SEP]`, padding tokens, continuation pieces beginning with `##`, and tokens followed by a continuation piece. Punctuation is retained when it satisfies these rules.

For SST-2, evaluated examples are selected from the same repository-provided labeled file `data/sst/test.txt` described above, after binary filtering, preprocessing, and tokenization. For Yelp, evaluated examples are selected from the repository-provided Yelp test split after the preprocessing and binary label mapping described above. Selection uses seed 0 and the resulting example identifiers are stored in fixed manifests used by all compared methods. We retain only correctly classified examples satisfying the relevant length restriction. Examples with no valid perturbable positions are excluded from one-token experiments, and two-token experiments require at least two valid perturbable positions. The one-token all-valid experiments contain 362 valid perturbation tasks on SST-2 and 587 on Yelp across 30 examples per dataset.

The two-token experiments use unordered pairs of valid perturbable positions under product-$(L_2 \times L_2)$ perturbations, with tokenized length at most 16, matching the prior multi-position protocol. Pair counts are workload counts, not independent statistical samples, because pairs from the same sentence share the same input, prediction, tokenization, and valid-position set.

#### 3.5.3. Verifier, Search Protocol, and Ranking

We instantiate the scheduler with an IBP $\rightarrow$ Forward $\rightarrow$ Backward verifier stack. Backward is the final fallback and defines the matched-radius reference. The matched cascade experiments use up to the first three valid positions per example for direct comparison against Backward verification, while the all-valid scheduling experiments validate skipped tasks over every valid perturbable position or pair in the selected examples.

Each radius search starts from radius 0.01. The upper radius is doubled while certification succeeds, up to 2.0. If the first positive candidate fails, it is halved until a certified lower endpoint is found or the radius drops below $10^{-12}$, in which case the certified zero endpoint is returned. The main experiments use the fixed setting $T = 4$ binary refinement steps according to Equation (9). The additional longer-sequence sensitivity experiment instead uses the adaptive tolerance-based refinement strategy described above. The radii reported for the main experiments are certified lower endpoints from the fixed search protocol, not globally maximal certifiable radii.

Gradient ranking uses the gradient of the cross-entropy loss $\mathcal{L}(f(x_0), y)$ with respect to the clean predicted label $y$, which equals the ground-truth label for all evaluated examples. Ties are broken by left-to-right token order. Random ranking uses seed 0. The ranking only proposes tasks; certificate acceptance comes from verifier validation and fallback search.

#### 3.5.4. Metrics, Timing, and Hardware

All certificate-agreement comparisons are paired: methods are evaluated on the same examples and perturbation tasks. We report certified radii, certificate agreement, elapsed runtime, verifier-stage calls, Backward calls avoided, full radius searches, validation calls, and fallback counts. Radius agreement is exact equality of the floating-point radius values returned by the fixed search procedure; using tolerance $10^{-6}$ gives the same counts in the reported all-valid scheduling comparisons.

Reported scheduling runtimes include selected-task radius searches, skipped-task validation, and fallback radius searches. They exclude model training, checkpoint loading,

preprocessing, example sampling, and gradient-ranking overhead. Wall-clock times include GPU computation and CPU-side orchestration. Since the long runs were not repeated with randomized execution order, timing results are interpreted as single-run implementation measurements and are reported together with more portable work counts.

Experiments were run on Ubuntu 20.04 with an Intel Xeon W7-2495X CPU, 128 GB RAM, and an NVIDIA RTX A4000 GPU. Verifier and model tensor computations use CUDA and the base verifier's default floating-point precision and acceptance rule. We do not use directed rounding or separate interval-arithmetic checks for GPU kernels. The revision experiments use Python 3.10.20, PyTorch 2.5.1+cu124, the CUDA 12.4 runtime bundled with PyTorch, and cuDNN 9.1.0. The NVIDIA driver version is 596.72. The verifier implementation corresponds to the exact source-code version identified by the archived repository commit reported in the Data Availability Statement.

The SHA-256 hashes of the checkpoint weights used in the revised experiments are: SST-2 one-layer, 926a88539068f2500f54e4a45ba55a8c982e5cb74007bc5034bd8ec3abe83428; SST-2 two-layer, c641971553c5bbd074a71495c0394c0b25dbced3f752aae25f1dc8faca55500e; and Yelp one-layer, c0e832fef541abe0238ce7d7fe4af2c3eff24d814aae58359e42822de362cd74.

The complete revised reproducibility artifact is archived under the release tag `paper-revision-artifact-v2`, corresponding to commit aed9f862ed02c7d5e12aa5eb6ddb66b7538cc121.

#### 3.5.5. AGNews2 Case Study

We also include a small AGNews2 case study to check whether the same scheduling behavior appears on a different dataset and a two-layer Transformer. The binary topic subset is derived from AG News [40] using World and Sports examples. The model has two Transformer layers, hidden size 256, intermediate size 512, four attention heads, ReLU activation, and no-variance layer normalization; its clean test accuracy is 0.979.

The five cases are selected before verification using only clean-model and tokenization properties: correct classification, tokenized length at most 32, at least three valid perturbable positions, class label, clean logit margin, and number of valid positions. The cases cover short/high-margin, low-margin, Sports/high-margin, Sports/low-margin, and long/many-position regimes. They are not selected using certified radius, runtime, fallback behavior, or verifier success. For computational feasibility, this case study evaluates the first three valid perturbable positions in each selected example rather than giving an all-valid-position certificate.

## 4. Results

### *4.1. All-Position Scheduling Versus Exhaustive Assessment*

Tables 2 and 3 summarize the main all-position scheduling experiment. The exhaustive cascade performs a full radius search for every valid perturbation task: 362 tasks on SST-2 and 587 tasks on Yelp. Perturbation-position scheduling reduces this number by searching selected tasks first, validating skipped tasks at the candidate radius, and fallback-searching skipped tasks that are not certified. Agreement is measured at the sentence level against exhaustive all-position cascade verification on the same examples.

The sentence agreement column shows that scheduling returns the same final sentence-level radius as exhaustive all-position cascade verification for every evaluated example. The reduction column shows the main efficiency benefit: at $k = 1$, full searches decrease from 362 to 31 on SST-2 and from 587 to 109 on Yelp. At $k = 2$, the selected set is larger, but fallback behavior changes: SST-2 requires 60 full searches, while Yelp drops to 73 because fewer skipped tasks fail validation. The budget alone does not determine runtime. What matters is whether the skipped positions pass the candidate radius check.

**Table 2.** Summary of all-position scheduling results relative to the exhaustive cascade.

| Dataset | Budget | Tasks | Exhaustive Searches | Scheduled Searches | Reduction | Sentence Agreement | Measured Speedup |
|---|---|---|---|---|---|---|---|
| SST-2 | 1 | 362 | 362 | 31 | 91.4% | 30/30 | 3.03× |
| SST-2 | 2 | 362 | 362 | 60 | 83.4% | 30/30 | 2.66× |
| Yelp | 1 | 587 | 587 | 109 | 81.4% | 30/30 | 3.11× |
| Yelp | 2 | 587 | 587 | 73 | 87.6% | 30/30 | 3.25× |

**Table 3.** Fallback-enabled perturbation-position scheduling over all valid positions.

| Dataset | Task Budget | Examples | Returned Covering Certs | Directly Validated Examples | Fallback Examples | Avg. Time/Example |
|---|---|---|---|---|---|---|
| SST-2 | 1 | 30 | 30/30 | 29/30 | 1/30 | 347 s |
| SST-2 | 2 | 30 | 30/30 | 30/30 | 0/30 | 394 s |
| Yelp | 1 | 30 | 30/30 | 20/30 | 10/30 | 1572 s |
| Yelp | 2 | 30 | 30/30 | 27/30 | 3/30 | 1503 s |

In our fallback-enabled runs, scheduling returns a sentence-level certificate covering all eligible positions under the verifier's nonnegative margin acceptance rule for all 30 examples on both datasets. On SST-2, $k = 1$ requires fallback on only 1/30 examples and $k = 2$ validates all skipped positions directly. Yelp is harder: $k = 1$ requires fallback on 10/30 examples, while $k = 2$ reduces this to 3/30 and is slightly faster overall because it avoids many fallback searches. Gradient ranking helps because it proposes a useful candidate radius. The certificate still comes from validation and fallback, not from the ranking score.

The 30/30 completion rate is expected with fallback enabled, because tasks that are not certified during validation are handled by full radius search. The more important quantities are the returned sentence-level radii, agreement with exhaustive search where available, and the reduction in full radius searches. Table 4 shows that all returned radii are positive, while Tables 5 and 6 report the corresponding agreement, runtime, search, and validation costs. Because verification runtimes are heavy-tailed, Table 7 reports per-example distribution summaries in addition to aggregate runtimes.

For exhaustive all-valid verification, the paired per-example cascade speedup has median 1.51× on SST-2 (IQR 1.47–1.58) and median 1.37× on Yelp (IQR 1.31–1.40). For perturbation-position scheduling, per-example runtimes are also skewed: SST-2 $k = 1$ has median 161.7s (IQR 52.5–488.7), while Yelp $k = 1$ has median 741.5s (IQR 255.6–1897.2). Thus, we interpret total wall-clock speedups together with portable work metrics such as full searches, validation calls, fallback positions, and Backward calls avoided.

The validation/search cost column applies only to scheduling rows. It reports the average cost of one skipped-task validation call divided by the average cost of one full radius search in the same run.

**Table 4.** Sentence-level radii returned by fallback-enabled perturbation-position scheduling.

| Dataset | Task Budget | Min $\epsilon$ | Median $\epsilon$ | Max $\epsilon$ | Positive Radii |
|---|---|---|---|---|---|
| SST-2 | 1 | 0.0225 | 0.5000 | 1.5200 | 30/30 |
| SST-2 | 2 | 0.0225 | 0.5000 | 1.5200 | 30/30 |
| Yelp | 1 | 0.0125 | 0.3200 | 0.5400 | 30/30 |
| Yelp | 2 | 0.0125 | 0.3200 | 0.5400 | 30/30 |

**Table 5.** Scheduling versus exhaustive verification: agreement and observed single-run runtime.

| Dataset | Method | Task Agreement | Sentence Agreement | Total Time | Speedup |
|---|---|---|---|---|---|
| SST-2 | Exhaustive Backward | Reference | Reference | 50,567 s | 1.00× |
| SST-2 | Exhaustive cascade | 362/362 | 30/30 | 31,470 s | 1.61× |
| SST-2 | Scheduling $k = 1$ | N/A | 30/30 | 10,401 s | 3.03× |
| SST-2 | Scheduling $k = 2$ | N/A | 30/30 | 11,814 s | 2.66× |
| Yelp | Exhaustive Backward | Reference | Reference | 215,381 s | 1.00× |
| Yelp | Exhaustive cascade | 587/587 | 30/30 | 146,496 s | 1.47× |
| Yelp | Scheduling $k = 1$ | N/A | 30/30 | 47,151 s | 3.11× |
| Yelp | Scheduling $k = 2$ | N/A | 30/30 | 45,083 s | 3.25× |

**Table 6.** Scheduling versus exhaustive verification: work counts and validation cost.

| Dataset | Method | Full Searches | Validation Calls | Fallback Tasks | Validation/Search Cost in Scheduling |
|---|---|---|---|---|---|
| SST-2 | Exhaustive Backward | 362 | 0 | 0 | N/A |
| SST-2 | Exhaustive cascade | 362 | 0 | 0 | N/A |
| SST-2 | Scheduling $k = 1$ | 31 | 332 | 1 | 18.4% |
| SST-2 | Scheduling $k = 2$ | 60 | 302 | 0 | 21.0% |
| Yelp | Exhaustive Backward | 587 | 0 | 0 | N/A |
| Yelp | Exhaustive cascade | 587 | 0 | 0 | N/A |
| Yelp | Scheduling $k = 1$ | 109 | 557 | 79 | 22.9% |
| Yelp | Scheduling $k = 2$ | 73 | 527 | 13 | 13.6% |

**Table 7.** Per-example runtime distributions for perturbation-position scheduling.

| Dataset | Task Budget | Mean Time | Median Time | IQR | Max Time |
|---|---|---|---|---|---|
| SST-2 | 1 | 347 s | 161 s | 52–488 s | 2168 s |
| SST-2 | 2 | 393 s | 180 s | 64–502 s | 2560 s |
| Yelp | 1 | 1571 s | 741 s | 255–1897 s | 6314 s |
| Yelp | 2 | 1502 s | 862 s | 275–2261 s | 5244 s |

Position-task agreement compares the radius returned by exhaustive cascade with the radius returned by exhaustive Backward for each perturbation task. Sentence-radius agreement compares the final minimum sentence radius returned by scheduling with the final minimum sentence radius returned by exhaustive cascade on the same example. Position-task agreement is not applicable to scheduling rows because scheduling does not independently search every position; scheduling is compared using the final sentence-level radius. Agreement is exact equality of the floating-point radius values saved by the fixed search procedure; using tolerance $10^{-6}$ gives the same counts. In these all-valid comparisons, the mean and maximum absolute sentence radius differences are both 0.0 for SST-2 and Yelp at $k = 1$ and $k = 2$. No cap-saturation events occurred in the reported all-valid exhaustive or scheduling comparisons.

For each dataset, exhaustive Backward is the strongest single-stage baseline in our verifier stack. The exhaustive cascade verifies the same valid positions with the early-exit cascade and is compared against exhaustive Backward to measure certificate agreement and Backward-call savings. The scheduling rows are then compared against exhaustive cascade, because scheduling uses the same cascade verifier but changes which tasks receive full radius search. Thus, the exhaustive cascade speedup is measured relative to exhaustive Backward, whereas the scheduling speedups are measured relative to exhaustive cascade.

Gradient ranking is an additional step used by scheduling, so Table 8 measures its cost on the same examples used in the all-valid scheduling experiments and reports verification pipeline totals after adding ranking time.

**Table 8.** Gradient-ranking cost and verification pipeline time for all-valid scheduling.

| Dataset and Budget | Ranking Time | Avg./Example | Median/Example | Verification Time | Pipeline Time | Ranking Overhead |
|---|---|---|---|---|---|---|
| SST-2, $k = 1$ | 1.95 s | 0.065 s | 0.056 s | 10,401 s | 10,403 s | 0.019% |
| SST-2, $k = 2$ | 1.95 s | 0.065 s | 0.056 s | 11,814 s | 11,816 s | 0.017% |
| Yelp, $k = 1$ | 0.59 s | 0.020 s | 0.007 s | 47,151 s | 47,152 s | 0.0013% |
| Yelp, $k = 2$ | 0.59 s | 0.020 s | 0.007 s | 45,082 s | 45,082 s | 0.0013% |

The ranking overhead is negligible relative to verification time: it is 0.019% and 0.017% of verification pipeline time on SST-2 for $k = 1$ and $k = 2$, respectively, and about 0.0013% on Yelp for both budgets.

Figures 2 and 3 provide a per-example view of the scheduling results. Speedup is measured relative to exhaustive all-position cascade verification on the same example, and fallback counts are skipped tasks that required full radius search. The speedup is positive on most examples, showing that the aggregate runtime reductions in Tables 5 and 6 are not caused by a single unusually easy case. SST-2 has consistently low fallback counts, especially at $k = 2$, where no skipped task requires fallback search. Yelp shows larger variability: several examples require many fallback searches, which reduces the achievable speedup even though the final radius still matches exhaustive cascade search. This behavior is consistent with the cost model in Proposition A1 in Appendix A: scheduling is most effective when skipped tasks validate cheaply at the candidate radius, and less effective when many skipped tasks require fallback search after validation.

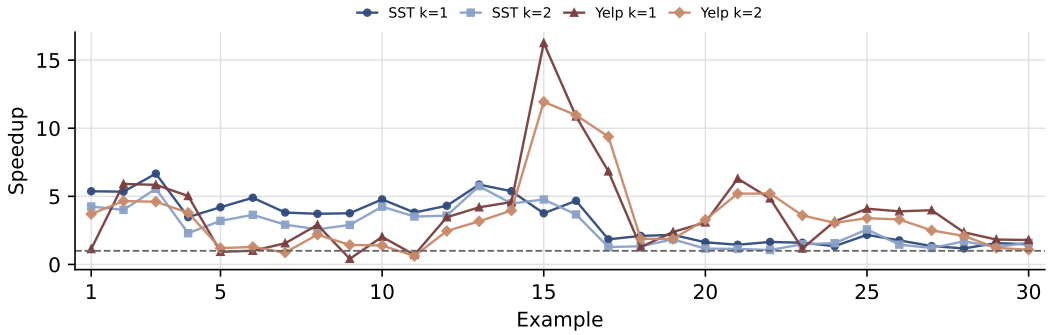



**Figure 2.** Per-example observed scheduling speedup relative to exhaustive all-position cascade verification. The dashed line indicates unit speedup.

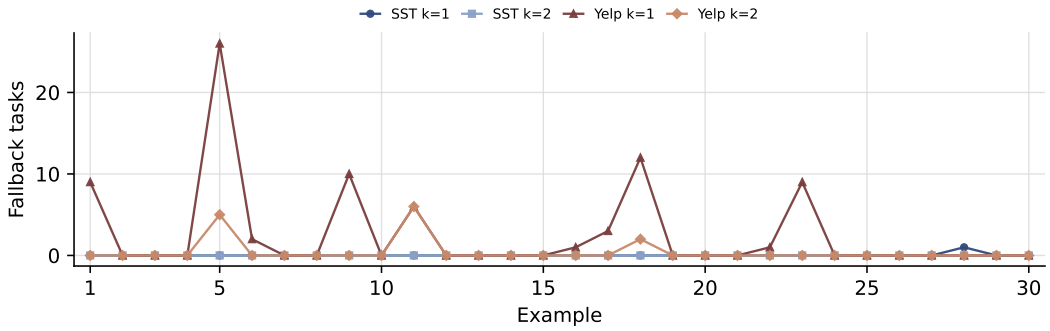



**Figure 3.** Per-example fallback task counts for perturbation-position scheduling.

#### 4.1.1. Expanded SST-2 Evaluation

To assess whether the scheduling behavior observed in the original 30-example SST-2 experiment persists over a larger sentence-level sample, we expanded the one-token $L_2$ all-valid evaluation to 100 SST-2 examples from the same preprocessed evaluation setting.

The results are summarized in Table 9. The same exhaustive cascade, gradient ranking, verifier configuration, and fixed four-refinement radius search protocol are used. Across the 100 examples, the exhaustive assessment contains 1485 valid perturbation-position tasks.

**Table 9.** Expanded 100-example SST-2 all-position scheduling results.

| Budget | Examples | Tasks | Exhaustive Searches | Scheduled Searches | Reduction | Sentence Agreement |
|---|---|---|---|---|---|---|
| $k = 1$ | 100 | 1485 | 1485 | 100 | 93.27% | 100/100 |
| $k = 2$ | 100 | 1485 | 1485 | 200 | 86.53% | 100/100 |

The expanded evaluation reproduces the exhaustive sentence-level radius for all 100 SST-2 examples at both scheduling budgets. The resulting sentence-level agreement rate is 100/100, with a 95% Clopper–Pearson exact confidence interval of 96.38–100.00% for both $k = 1$ and $k = 2$. The paired sentence radius differences are identically zero: the mean, median, and maximum absolute differences are all 0 for both budgets.

Efficiency statistics are also computed at the sentence level rather than treating individual perturbation positions as independent observations. At $k = 1$, the mean per-sentence reduction in full radius searches is 91.40%, with a 95% bootstrap confidence interval of 90.31–92.40%, while the median per-sentence reduction is 93.33%. At $k = 2$, the corresponding mean is 82.79% with a 95% bootstrap confidence interval of 80.62–84.80%, and the median is 86.67%.

The aggregate workload reductions are 93.27% at $k = 1$ and 86.53% at $k = 2$, reducing 1485 exhaustive full searches to 100 and 200, respectively. These aggregate values differ from the mean sentence-level reductions because sentences contain different numbers of valid perturbation positions: the sentence-level mean weights each evaluated sentence equally, whereas the aggregate reduction weights sentences according to their number of verification tasks. No fallback search or validation failure occurs in either configuration.

These results allow answering RQ1. In the matched 30-example cross-dataset evaluation, scheduling reproduces the exhaustive all-valid cascade radius on 30/30 SST-2 and 30/30 Yelp examples for both $k = 1$ and $k = 2$. The expanded SST-2 evaluation further reproduces the exhaustive sentence-level radius on 100/100 examples for both budgets. In this larger experiment, full radius searches decrease from 1485 to 100 at $k = 1$ and to 200 at $k = 2$, corresponding to reductions of 93.27% and 86.53%, respectively. Together, the matched cross-dataset results and the expanded SST-2 evaluation show that the observed scheduling agreement is not restricted to the original 30-example SST-2 cohort.

Table 10 compares the effect of gradient, left-to-right, and random ranking on scheduling efficiency.

Among the completed ranking-ablation runs, all methods recover the exhaustive all-valid cascade radius on SST-2. Validation and fallback ensure that the ranking heuristic does not determine soundness; instead, the ranking affects cost through the number of fallback searches. Among completed configurations, gradient ranking gives the lowest cost: at $k = 1$, it triggers only one fallback position, compared with 37 for left-to-right and 89 for the completed random seed run. This suggests that the gradient score is useful as a proposal heuristic in the completed SST-2 configurations, while soundness comes from validation and fallback rather than from the heuristic itself.

We additionally attempted five random-ranking seeds. These runs did not produce certificate disagreements in the completed examples, but some runs exhausted GPU memory during selected-task radius search before completing the full dataset-level evaluation. These failures indicate that the ranking strategy can influence resource usage in the current implementation and hint at the computational efficiency of our gradient ranking method

as a practical heuristic that provided the lowest observed verification cost among the completed configurations.

**Table 10.** Ranking ablation for perturbation-position scheduling on SST-2.

| Ranking | Task Budget | Exhaustive Agreement | Fallback Positions | Full Searches Avoided | Total Time |
|---|---|---|---|---|---|
| Gradient | 1 | 30/30 | 1 | 331 | 10,401 s |
| Left-to-right | 1 | 30/30 | 37 | 295 | 12,475 s |
| Random (seed 0) | 1 | 30/30 | 89 | 243 | 19,599 s |
| Gradient | 2 | 30/30 | 0 | 302 | 11,814 s |
| Left-to-right | 2 | 30/30 | 16 | 286 | 13,427 s |
| Random (seed 0) | 2 | 30/30 | 40 | 262 | 20,593 s |

#### 4.1.2. Scheduler Component Ablations

To isolate the contribution of fallback search, we compare the complete scheduler with an ablated variant in which a failed skipped-task validation returns `unknown` instead of triggering additional radius search. This corresponds to the two fallback modes in Algorithm 1: $\rho$ = search for the complete method and $\rho$ = unknown for the no-fallback variant. All other components, including gradient ranking, the selected-task budget, and the IBP $\rightarrow$ Forward $\rightarrow$ Backward verifier cascade, are unchanged. Because the no-fallback variant does not make any additional acceptance decision after failed validation, it remains verifier-conditionally sound; however, it may fail to return a sentence-level radius. The resulting comparison is reported in Table 11.

**Table 11.** Fallback-search ablation on the 30-example Yelp one-token $L_2$ experiment.

| Budget | Variant | Completed | Unknown | Full Searches | Reduction | Sentence Agreement |
|---|---|---|---|---|---|---|
| $k = 1$ | Full method | 30/30 | 0 | 109 | 81.43% | 30/30 |
| $k = 1$ | No fallback | 20/30 | 10 | 30 | 94.89% | 20/20 |
| $k = 2$ | Full method | 30/30 | 0 | 73 | 87.56% | 30/30 |
| $k = 2$ | No fallback | 27/30 | 3 | 60 | 89.78% | 27/27 |

The ablation exposes the cost–completion trade-off introduced by fallback search. At $k = 1$, validation fails for 10 of the 30 Yelp examples. Without fallback, these cases remain unknown and only 20/30 examples produce a sentence-level certificate. Enabling fallback searches the 79 failed perturbation tasks, increasing the number of full searches from 30 to 109 but recovering a completed result for all 30 examples. The measured runtime of the full method is 47,152 s, whereas the corresponding no-fallback runtime, derived from the execution traces by excluding fallback-search work, is 34,531 s. At $k = 2$, validation fails for only three examples, corresponding to 13 failed perturbation tasks. Fallback therefore increases the number of full searches from 60 to 73 while raising completion from 27/30 to 30/30. The measured full-method runtime is 45,082 s, compared with a trace-derived no-fallback runtime of 40,675 s.

Importantly, whenever the no-fallback variant returns a radius, it agrees with the exhaustive sentence-level reference: 20/20 examples for $k = 1$ and 27/27 for $k = 2$. Thus, fallback search does not serve as an additional acceptance heuristic; rather, it resolves validation failures that would otherwise remain unknown. The results show that the search–validate–fallback design trades additional verification work for complete coverage of the evaluated perturbation tasks.

To further examine the robustness of the scheduling strategy under longer verified sequences and an adaptive radius search resolution, we performed an additional 20-example

SST-2 sensitivity experiment with a maximum verification length of 64 tokens. Unlike the main fixed-resolution experiments, this evaluation uses the tolerance-based binary refinement procedure described above, where refinement continues until the interval between the certified and uncertified radius bounds is below the specified tolerance. The experiment uses the same IBP → Forward → Backward cascade. The evaluation contains 289 valid one-token perturbation tasks, with tokenized lengths ranging from 8 to 42 tokens, including examples beyond the original length-32 setting. At budget $k = 1$, scheduling reproduces the exhaustive sentence-level cascade radius on all 20 examples while reducing the number of full radius searches from 289 to 20, corresponding to a 93.08% reduction. No validation failures or fallback searches are required in this experiment. This result provides a controlled sensitivity analysis showing that the proposed search–validate procedure remains effective under a longer-sequence setting and a different radius search strategy.

#### 4.1.3. Controlled Depth Scaling

To examine whether the scheduling behavior persists as verification cost increases with model depth, we compare the one- and two-layer SST-2 models on the same fixed 10 sentences and 115 valid perturbation-position tasks. All architectural and verification settings are held fixed except for Transformer depth. Table 12 reports the scheduling results.

**Table 12.** Matched SST-2 scheduling results for one- and two-layer Transformers.

| Layers | Examples | Tasks | Agreement | Full Searches | Reduction | Fallback | Scheduler Time |
|---|---|---|---|---|---|---|---|
| 1 | 10 | 115 | 10/10 | 10 | 91.30% | 0 | 1627 s |
| 2 | 10 | 115 | 10/10 | 13 | 88.70% | 3 | 6407 s |

The scheduler reproduces the exhaustive cascade sentence-level radius on all 10 examples at both depths. Increasing depth substantially raises the cost of verification: relative to the one-layer model, the two-layer model requires 3.15× the pure-Backward runtime, 4.63× the exhaustive cascade runtime, and 3.94× the scheduler runtime. Nevertheless, the scheduler continues to avoid most full radius searches, reducing 115 exhaustive searches to 10 for the one-layer model and 13 for the two-layer model. The three additional two-layer searches arise from fallback processing in one example.

This experiment provides evidence that the scheduling mechanism remains effective when Transformer depth and verification cost increase within the evaluated architecture. It does not establish scalability to standard pretrained or large language models.

### *4.2. Multi-Position Perturbation Assessment*

For the two-token experiment, each perturbation task is an unordered pair of valid perturbable positions under the product perturbation set in Equation (4). This setting is substantially more expensive because the number of perturbation tasks grows quadratically with the number of valid positions.

Table 13 applies the same validation-and-fallback scheduling protocol to unordered position pairs.

On SST-2, all skipped pairs validate directly at the candidate radius, so no fallback pair search is needed. On Yelp, fallback is required on 10/30 examples for $k = 1$ and 7/30 examples for $k = 2$, but the final sentence-level pair radius still agrees with exhaustive pairwise cascade on all examples. Thus, perturbation-position scheduling recovers the exhaustive pairwise cascade radius while reducing full pair radius searches from 1089 to 30 on SST-2 and from 1301 to 117 on Yelp at $k = 1$.

**Table 13.** Two-token product-$L_2$ perturbation-position scheduling.

| Dataset | Task Budget | Examples | Final Agreement | Directly Validated | Fallback Examples | Full Searches | Total Time |
|---|---|---|---|---|---|---|---|
| SST-2 | 1 | 30 | 30/30 | 30/30 | 0/30 | 30 | 3678 s |
| SST-2 | 2 | 30 | 30/30 | 30/30 | 0/30 | 60 | 4114 s |
| Yelp | 1 | 30 | 30/30 | 20/30 | 10/30 | 117 | 7411 s |
| Yelp | 2 | 30 | 30/30 | 23/30 | 7/30 | 77 | 5638 s |

As supporting evidence for the underlying radius query mechanism, Table 14 evaluates the early-exit cascade against pure Backward verification on the same two-token task set. Across 30 examples, this produces 1089 pair tasks on SST-2 and 1301 pair tasks on Yelp; these pair counts measure verification workload rather than independent statistical samples. The cascade matches the Backward-certified radius on every pair in both datasets. On SST-2, it reduces Backward calls by 32.33% and gives an observed 1.39× total-time speedup. On Yelp, it reduces Backward calls by 20.23% and gives an observed 1.20× speedup. The cascade still helps in the pair setting, but the gain is smaller, especially on Yelp.

**Table 14.** Two-token product-$L_2$ early-exit cascade verification.

| Setting | Examples | Pair Tasks | Cert. Agreement | Pure Backward | Cascade Time | Speedup | Calls Saved |
|---|---|---|---|---|---|---|---|
| SST-2, length $\leq 16$ | 30 | 1089 | 1089/1089 | 10,801 calls/29,983 s | 21,596 s | 1.39× | 3492/10,801 |
| Yelp, length $\leq 16$ | 30 | 1301 | 1301/1301 | 12,865 calls/40,567 s | 33,884 s | 1.20× | 2603/12,865 |

The two-token experiment addresses the perturbation-order part of RQ3. The cascade continues to match Backward-certified radii for all pair tasks on both datasets, and perturbation-position scheduling recovers the exhaustive pairwise cascade radius on all examples while greatly reducing the number of full pair radius searches. Ranking sensitivity is evaluated separately in the one-token SST-2 scheduling ablation. Although this experiment follows the prior multi-position $L_2$ protocol, extending fallback-enabled scheduling to three- and four-token perturbations remains future work because the number of position combinations grows combinatorially and GPU memory becomes limiting.

### *4.3. Qualitative AGNews2 Case Study*

To test whether the same behavior appears outside SST-2 and Yelp, Table 15 reports a three-position case study on five curated AGNews2 binary topic examples using a deeper two-layer Transformer model.

Across the five cases, scheduling recovers the exhaustive cascade radius for both budgets. At $k = 1$, it performs 5 full searches instead of 15, reducing full searches by 66.7% and giving a 2.16× aggregate speedup over exhaustive cascade. At $k = 2$, it performs 10 full searches, reducing full searches by 33.3% and giving a 1.22× aggregate speedup. No skipped task requires fallback search in these cases. Because this case study evaluates only three reference positions per example, $k = 2$ leaves only one skipped position to validate; therefore its savings are smaller than $k = 1$ and it is slower than exhaustive cascade in two cases. The result should be read as qualitative evidence on a different dataset and deeper model, not as a replacement for the 30-example SST-2/Yelp evaluation.

**Table 15.** Three-position AGNews2 case study: scheduling versus exhaustive cascade on a two-layer Transformer.

| Case | Type | Budget | Cascade Radius | Scheduled Radius | Agreement | Scheduled Time | Speedup |
|---|---|---|---|---|---|---|---|
| 0 | Short high-margin | 1 | 0.38 | 0.38 | Yes | 413 s | 1.52× |
| 0 | Short high-margin | 2 | 0.38 | 0.38 | Yes | 724 s | 0.87× |
| 1 | Low-margin | 1 | 0.0775 | 0.0775 | Yes | 2335 s | 2.27× |
| 1 | Low-margin | 2 | 0.0775 | 0.0775 | Yes | 4453 s | 1.19× |
| 2 | Sports high-margin | 1 | 0.92 | 0.92 | Yes | 5871 s | 1.37× |
| 2 | Sports high-margin | 2 | 0.92 | 0.92 | Yes | 9699 s | 0.83× |
| 3 | Sports low-margin | 1 | 0.14 | 0.14 | Yes | 2783 s | 2.78× |
| 3 | Sports low-margin | 2 | 0.14 | 0.14 | Yes | 4696 s | 1.65× |
| 4 | Long many-position | 1 | 0.80 | 0.80 | Yes | 3098 s | 3.11× |
| 4 | Long many-position | 2 | 0.80 | 0.80 | Yes | 6065 s | 1.59× |

### *4.4. Certificate Agreement and Efficiency of the Verifier Cascade*

As a supporting analysis, we examine the verifier-stage cascade used as the underlying radius query mechanism. The cascade is configurable, but our main implementation uses IBP → Forward → Backward. Backward is the final cascade stage and provides the matched-radius reference. As shown below, the practical savings come almost entirely from Forward certifications; IBP is retained as a cheap optional stage but certifies no candidate radii in our evaluated settings.

Table 16 evaluates the early-exit verifier cascade `IBP` → `Forward` → `Backward` against pure Backward verification on matched one-token perturbation-position tasks. Because the cascade uses an OR over verifier stages, radius agreement with pure Backward is not a formal consequence unless Backward pointwise dominates the earlier stages. We do not rely on such a dominance theorem. Instead, we report radius agreement empirically and use verifier-stage call counts to measure the runtime benefit.

**Table 16.** Matched one-token early-exit cascade comparison with observed single-run runtimes.

| Dataset | Norm | Tasks | Cert. Agreement | Backward Time/Calls | Cascade Time | Speedup | Forward Certs | Calls Saved |
|---|---|---|---|---|---|---|---|---|
| SST-2 | $L_1$ | 90 | 90/90 | 8614 s/923 | 3011 s | 2.86× | 572 | 572/923 |
| SST-2 | $L_2$ | 90 | 90/90 | 11,819 s/983 | 7603 s | 1.55× | 354 | 354/983 |
| SST-2 | $L_\infty$ | 90 | 90/90 | 8087 s/671 | 7602 s | 1.06× | 44 | 44/671 |
| Yelp | $L_1$ | 90 | 90/90 | 31,723 s/1103 | 19,378 s | 1.64× | 386 | 386/1103 |
| Yelp | $L_2$ | 90 | 90/90 | 26,614 s/942 | 18,956 s | 1.40× | 241 | 241/942 |
| Yelp | $L_\infty$ | 90 | 90/90 | 18,801 s/663 | 18,109 s | 1.04× | 1 | 1/663 |

Across all six settings, the cascade matches the Backward-certified radius on every matched task. Since we do not prove pointwise dominance between Forward and Backward bounds, this agreement is treated as an empirical result rather than a formal consequence of using Backward as the final fallback. IBP certifies no candidate radii, whereas Forward avoids 62.0%, 36.0%, and 6.6% of Backward calls on SST-2 under $L_1$, $L_2$, and $L_\infty$, respectively. On Yelp, the corresponding reductions are 35.0%, 25.6%, and 0.15%.

Consequently, speedups are largest under $L_1$ (2.86× on SST-2 and 1.64× on Yelp), moderate under $L_2$, and negligible under $L_\infty$. The Yelp $L_\infty$ result, where Forward certifies only one candidate, shows that the cascade provides little benefit when cheaper stages

rarely succeed. Thus, the cascade is applicable across norms and datasets, but its efficiency gain depends on the certification effectiveness of its cheaper stages. The reported wall-clock speedups are hardware-dependent; reductions in Backward calls provide the more portable efficiency measure.

The small runtime difference between Forward → Backward and IBP → Forward → Backward should not be overinterpreted, because these long-running experiments were not repeated; the stable work metric is that both configurations make the same number of Forward and Backward calls, and IBP certifies no candidates in this setting.

Table 17 isolates the contribution of each cascade stage on matched SST-2 one-token $L_2$ tasks.

**Table 17.** Cascade-stage ablation on matched one-token $L_2$ SST-2 tasks.

| Cascade | Cert. Agreement | Total Time | Speedup | Forward Calls | Backward Calls |
|---|---|---|---|---|---|
| Backward only | Reference | 11,820 s | 1.00× | 0 | 983 |
| Forward → Backward | 90/90 | 7679 s | 1.54× | 983 | 629 |
| IBP → Forward → Backward | 90/90 | 7603 s | 1.55× | 983 | 629 |

We perform the stage-order ablation on SST-2 because its purpose is to isolate whether IBP contributes beyond the Forward stage. Cross-dataset cascade behavior is reported separately in Table 16.

The ablation shows that Forward is responsible for the avoided Backward calls: both cascades reduce Backward calls from 983 to 629 and match the Backward radius on all 90 tasks. IBP certifies no candidate radii in this setting. Because IBP is very cheap, adding it does not materially change total runtime, but the practical savings come from Forward certifications rather than IBP.

These results answer RQ2. Early-exit staging preserves the Backward-certified radius on all matched tasks while reducing Backward calls. The ablation shows that the savings are due to Forward certifications: IBP certifies no candidate radii in this setting and contributes little beyond negligible overhead.

## 5. Discussion

The main source of savings is that sentence-level certified-radius assessment requires many related verification tasks: each eligible token position, or position pair, must be considered because the final sentence radius is the minimum over tasks. Exhaustive assessment therefore repeats full radius search even for positions that are not radius-limiting. Perturbation-position scheduling exploits this structure. The selected tasks provide a candidate sentence radius, and skipped tasks only need to be checked at that single candidate radius unless validation fails. This changes the computation from "search everywhere" to "search selected tasks, validate the rest, and fallback only where needed."

The ablations show that ranking affects efficiency but not certificate acceptance. Gradient ranking is useful because it tends to select positions that produce a conservative candidate radius, which makes skipped-task validation more likely to succeed. When ranking is weaker, as in left-to-right or random selection, fallback searches increase and the runtime benefit decreases. Fallback is therefore essential: it preserves coverage of all eligible tasks, while the ranking heuristic controls how much work is avoided.

The SST-2 and Yelp results show different fallback behavior. On SST-2, scheduling reduces full searches from 362 to 31 at $k = 1$ and to 60 at $k = 2$; fallback is almost absent, with only one fallback task at $k = 1$ and none at $k = 2$. Yelp is more variable: full searches decrease from 587 to 109 at $k = 1$ and to 73 at $k = 2$, but fallback is required more often, especially at $k = 1$. This explains why the best budget is dataset-dependent. Increasing $k$

adds selected-task searches, but it can also reduce fallback enough to improve total runtime, as observed on Yelp.

The AGNews2 case study shows the same pattern on a deeper two-layer model: $k = 1$ gives consistent savings, while $k = 2$ is less useful when only three reference positions are evaluated.

The method is useful when many related robustness checks must be run on the same model, for example when auditing a classifier or comparing verification settings. It does not replace tighter or faster Transformer verifiers; it changes how many full searches are needed when using one. A stronger verifier backend could be placed inside the same scheduling loop.

For RQ1, fallback-enabled perturbation-position scheduling reproduces the exhaustive sentence-level radius on all evaluated SST-2 and Yelp examples under the fixed four-refinement search protocol. Most importantly, the expanded SST-2 evaluation reproduces the exhaustive result on 100/100 sentences at both scheduling budgets, reducing 1485 exhaustive full searches to 100 at $k = 1$ and 200 at $k = 2$. The matched cross-dataset evaluation additionally reproduces the exhaustive radius on 30/30 SST-2 and 30/30 Yelp examples at both budgets.

For RQ2, the early-exit cascade reproduces the Backward-certified radii on all matched one-token and two-token tasks while reducing expensive Backward calls. The cascade-stage ablation shows that the observed savings arise from Forward certifications, whereas IBP contributes little beyond negligible overhead in the evaluated setting.

For RQ3, the results show that scheduling efficiency depends on dataset, task budget, ranking heuristic, perturbation norm, perturbation order, and model depth. SST-2 generally requires less fallback work than Yelp, while gradient ranking gives the lowest observed cost among the completed ranking configurations. The two-token setting preserves agreement while increasing the verification workload because the number of perturbation tasks grows quadratically. The cascade experiments across $L_1$, $L_2$, and $L_\infty$ show that verifier-stage behavior also depends on the perturbation norm. In the matched depth experiment, increasing the Transformer from one to two layers substantially increases verification runtime, while the scheduler still reproduces the exhaustive sentence-level radius on 10/10 examples and reduces 115 exhaustive full searches to 13 for the two-layer model. Finally, the AGNews2 case study provides supporting evidence on a different dataset and a two-layer Transformer: in the three-position case-study setting, $k = 1$ recovers the exhaustive three-position cascade radius on all five cases while reducing full searches by 66.7%.

## 6. Conclusions and Future Work

Certified robustness assessment of Transformer text classifiers is computationally demanding because sentence-level certificates require evaluating multiple perturbation-position tasks. This work introduced verifier-aware perturbation-position scheduling, a search–validate–fallback mechanism that reduces the number of full radius searches while preserving coverage of all eligible perturbation tasks. Unlike methods that improve the underlying verifier, the proposed approach operates as an external scheduling layer.

Across the evaluated settings (expanded all-position evaluation with 100 SST-2 examples; cross-dataset Yelp evaluation with 30 examples), the scheduler reproduced the corresponding exhaustive sentence-level outcomes cutting repeated verification work. Additional controlled experiments examine one- and two-layer Transformers, two-token perturbations, model depth, longer verified sequences, and an adaptive radius-search strategy.

The presented results are promising and provide a solid basis for extending the current experimental setting. The evaluated models are small verification-oriented Transformers using no-variance layer normalization, and the experiments focus on continuous norm-

bounded embedding perturbations. The main experiments use a fixed four-step binary refinement protocol; the additional 20-example SST-2 sensitivity experiment provides evidence under an adaptive tolerance-based search, but does not constitute a comprehensive study of search-resolution choices. The reported guarantees are also conditional on the soundness and numerical behavior of the underlying verifier implementation.

Future work can build on these results by investigating larger pretrained architectures, integrating additional and more recent Transformer verification backends, considering substantially longer sequences, exploring broader radius search configurations, and extending the perturbation models to cover a wider range of linguistic transformations.

**Author Contributions:** Conceptualization, V.S., R.B., L.L., D.M.G. and F.B.; methodology, V.S., A.P. and F.B.; software, V.S.; writing—original draft, V.S., L.L. and F.B.; writing—review and editing, R.B., L.L., D.M.G., A.P. and F.B.; visualization, V.S. and L.L.; supervision, R.B., D.M.G. and F.B.; funding acquisition, D.M.G. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the Spanish Government under Grants: PID2024-157191OB-C21 funded by MICIU/AEI/10.13039/501100011033 and ERDF/EU, and PID2022-140554OB-C32 funded by MCIN/AEI/10.13039/501100011033; and funded by the Comunidad de Madrid under Grants: SEGVAUTO 5Gen-CM (TEC-2024/ECO-277) and DIGIMATER-CM (TEC-2024/TEC-102).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The implementation, experiment scripts, figure generation scripts, command lines, environment specifications, raw JSON result logs, fixed selected-example manifests, and trained model checkpoints used in this study are publicly available in the project repository at https://github.com/Elios-Lab/Transformer-Position-Scheduling-Certification (accessed on 6 August 2026). The revised artifact provides the fixed selected-example manifests, checkpoint SHA-256 checksums, and the exact software environment used for the reported experiments. The results reported in this revision correspond to archived release `paper-revision-artifact-v2` and Git commit `aed9f862ed02c7d5e12aa5eb6ddb66b7538cc121`. The exact SST-2, Yelp, and AGNews2 selected-example identifiers are included in that release. SST-2, Yelp, and AG News are third-party public datasets available from their original sources cited in this article.

**Conflicts of Interest:** The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## Appendix A. Proof Details and Radius-Search Procedure

This appendix gives the radius search details and proof arguments omitted from the main text. These details support the verifier-conditional guarantees stated in Section 3; they do not introduce additional acceptance rules beyond the underlying verifier decisions.

Algorithm A1 returns a certified lower endpoint under the fixed expansion-plus-binary search protocol. The soundness of the returned endpoint does not require monotonicity, because the returned value is accepted only after a verifier query certifies it. Monotonicity is required only when interpreting the procedure as an approximation to the largest certifiable radius or when comparing independent finite-resolution searches. Radius search first checks the acceptance condition at radius zero. If the zero-radius margin does not satisfy the chosen threshold $\tau$, the procedure returns unknown rather than a certified radius. This matters for $\tau > 0$, where a correctly classified clean input need not have margin at least $\tau$.

**Algorithm A1** Certified radius search for one perturbation task

**Require:** Input $x_0$, label $y$, perturbation position set $Q$, acceptance threshold $\tau$, initial radius $\epsilon_0$, maximum radius $\epsilon_{\max}$, iterations $T$

All verifier queries use acceptance threshold $\tau$
Query verifier cascade at radius 0
**if** 0 is not certified **then**
  **return** unknown
**end if**
$\ell \leftarrow 0, u \leftarrow \epsilon_0$
Query verifier cascade at radius $u$
**if** $u$ is certified **then**
  **while** $u < \epsilon_{\max}$ and $u$ is certified **do**
    $\ell \leftarrow u$
    $u \leftarrow \min(2u, \epsilon_{\max})$
    Query verifier cascade at radius $u$
  **end while**
  **if** $u = \epsilon_{\max}$ and $u$ is certified **then**
    **return** certified lower endpoint $u$ {cap saturation}
  **end if**
**else**
  **while** $u > 10^{-12}$ and $u$ is not certified **do**
    $u \leftarrow u/2$
    Query verifier cascade at radius $u$
  **end while**
  **if** $u \leq 10^{-12}$ and $u$ is not certified **then**
    **return** certified lower endpoint $\ell$
  **end if**
  $\ell \leftarrow u, u \leftarrow 2u$
**end if**
**for** $t = 1$ to $T$ **do**
  $m \leftarrow (\ell + u)/2$
  Query verifier cascade at radius $m$
  **if** $m$ is certified **then**
    $\ell \leftarrow m$
  **else**
    $u \leftarrow m$
  **end if**
**end for**
**return** certified lower endpoint $\ell$

**Lemma A1** (Verifier-conditional radius search soundness)**.** *Assume every verifier cascade acceptance decision is sound at threshold $\tau$. If Algorithm A1 returns a radius $\ell$ rather than unknown, then $\ell$ is a certified lower endpoint for the perturbation task $Q$ at threshold $\tau$.*

**Proof.** Algorithm A1 first queries radius zero and returns unknown if zero is not certified. Therefore the initial lower endpoint $\ell = 0$ is certified whenever the search continues. The algorithm updates $\ell$ only after a verifier cascade query certifies the queried radius. If the expansion reaches the cap while certified, the returned cap value has also been certified by a verifier query. Hence every non-unknown return value was directly certified by the cascade. By the assumed soundness of verifier cascade acceptance decisions, it is a certified lower endpoint for task $Q$ at threshold $\tau$. □

**Proof of Theorem 1.** The ranking score is used only to prioritize perturbation tasks for early radius search; it is not used as evidence of robustness. By Lemma A1, selected-task and fallback-search endpoints returned by Algorithm A1 are certified lower endpoints. By Lemma 1, every skipped task accepted during validation is certified at $\epsilon_{\text{cand}}(k)$. Since the

scheduling procedure returns a radius only when all selected-task and fallback searches return certified lower endpoints, every searched task used in the returned minimum has a certified lower endpoint.

For each selected task $Q \in S_k$, radius search returns a certified lower endpoint $\epsilon_Q$. The candidate radius is

$$\epsilon_{\text{cand}}(k) = \min_{Q \in S_k} \epsilon_Q.$$

Thus every selected task is certified at $\epsilon_{\text{cand}}(k)$.

For each skipped task $Q \in \mathcal{Q} \setminus S_k$, the algorithm validates $\epsilon_{\text{cand}}(k)$ using the cascade. Let $F_k$ be the skipped tasks that are not certified at $\epsilon_{\text{cand}}(k)$. If $F_k = \emptyset$, then all tasks in $\mathcal{Q}$ are certified at $\epsilon_{\text{cand}}(k)$, and the claim follows.

If $F_k \neq \emptyset$, fallback radius search is run for every $Q \in F_k$, returning certified lower endpoints $\epsilon_Q$. The algorithm returns

$$\epsilon_{\text{sched}}(k) = \min\left(\epsilon_{\text{cand}}(k), \min_{Q \in F_k} \epsilon_Q\right).$$

For selected tasks, $\epsilon_{\text{sched}}(k) \leq \epsilon_{\text{cand}}(k) \leq \epsilon_Q$. For skipped tasks certified during validation, $\epsilon_{\text{sched}}(k) \leq \epsilon_{\text{cand}}(k)$. For fallback tasks $Q \in F_k$, $\epsilon_{\text{sched}}(k) \leq \epsilon_Q$ by construction.

For any perturbation task $Q$, the perturbation sets are nested:

$$B_p^Q(x_0, \epsilon') \subseteq B_p^Q(x_0, \epsilon) \qquad \text{whenever } \epsilon' \leq \epsilon.$$

Therefore, certification at a radius also certifies any smaller radius. Hence every task $Q \in \mathcal{Q}$ is certified at the returned radius $\epsilon_{\text{sched}}(k)$. □

**Theorem A1** (Conditional finite-resolution exhaustive-search comparison)**.** *This comparison result is not needed for certificate soundness; it only characterizes agreement with an independently executed finite-resolution exhaustive search. Assume the same deterministic radius search procedure, verifier cascade, and threshold $\tau$ are used for both scheduling and exhaustive search. Assume also that the compared searches terminate with valid certified/uncertified brackets rather than cap saturation, and that the verifier cascade decision predicate $D_Q(\epsilon)$ is monotone over queried radii for every task $Q \in \mathcal{Q}$. Let $\epsilon_{\text{full}}$ be the sentence-level radius returned by independent exhaustive search over $\mathcal{Q}$, and let $\epsilon_{\text{sched}}$ be the radius returned by scheduling with fallback. Define*

$$\delta_{\max} = \begin{cases} \max_{Q \in V_k} \delta_Q, & V_k \neq \emptyset, \\ 0, & V_k = \emptyset, \end{cases}$$

*where $V_k$ is the set of skipped tasks that validate successfully at $\epsilon_{\text{cand}}$, and $\delta_Q$ is the final search interval width that independent exhaustive search would obtain for task $Q$. Then*

$$0 \leq \epsilon_{\text{sched}} - \epsilon_{\text{full}} \leq \delta_{\max}.$$

**Proof.** Selected tasks are fully searched in both procedures. Any skipped task that does not certify at $\epsilon_{\text{cand}}$ is also fully searched by fallback, so such tasks cannot be ignored by scheduling. The only difference from an independently executed exhaustive finite-resolution search is a skipped task that validates successfully at $\epsilon_{\text{cand}}$: scheduling records $\epsilon_{\text{cand}}$ as a certified lower endpoint, while an independent binary search may not query that exact radius.

For any such validated task $Q \in V_k$, monotonicity implies that its certified lower endpoint is at least $\epsilon_{\text{cand}}$. The independent finite-resolution search can therefore return a

lower endpoint at most $\delta_Q$ below this value. Taking the maximum over validated skipped tasks gives

$$0 \leq \epsilon_{\text{sched}} - \epsilon_{\text{full}} \leq \delta_{\max}.$$

If $V_k = \emptyset$, then every skipped task is handled by fallback search, so scheduling and exhaustive search use the same deterministic search results and $\delta_{\max} = 0$. □

*Appendix A.1. Remark on Monotone Closure*

If the implemented verifier cascade decision predicate is not monotone over queried radii, one can define a monotone closure

$$\overline{D}_Q(\epsilon) = 1 \quad \Longleftrightarrow \quad \exists \epsilon' \geq \epsilon \text{ queried such that } D_Q(\epsilon') = 1.$$

This closure is sound because certification at a larger radius semantically certifies every smaller radius for the same perturbation task $Q$. Our reported implementation does not explicitly apply this closure; therefore Theorem A1 is stated as a conditional finite-resolution comparison, while soundness of returned radii is independent of the monotonicity assumption but remains conditional on verifier-stage soundness as stated in Theorem 1.

*Appendix A.2. Search Saturation*

If the expansion phase reaches $\epsilon_{\max}$ and the verifier certifies that radius, Algorithm A1 returns the certified cap value $\epsilon_{\max}$. This value is sound but should be interpreted as a cap-saturated lower endpoint rather than a tightly bracketed approximation to the largest certifiable radius. In our reported finite-resolution comparisons, agreement is therefore understood with respect to the implemented search protocol and its returned certified lower endpoints.

**Proposition A1** (Verification cost)**.** *Let $C_{\text{search}}(Q)$ be the cost of a full cascade radius search for task $Q$, and let $C_{\text{val}}(Q, \epsilon_{\text{cand}})$ be the cost of validating the candidate radius at a skipped task. Exhaustive verification costs*

$$C_{\text{full}} = \sum_{Q \in \mathcal{Q}} C_{\text{search}}(Q).$$

*Scheduling with fallback costs*

$$C_{\text{sched}} = \sum_{Q \in S_k} C_{\text{search}}(Q) + \sum_{Q \in \mathcal{Q} \setminus S_k} C_{\text{val}}(Q, \epsilon_{\text{cand}}) + \sum_{Q \in F_k} C_{\text{search}}(Q).$$

*Let $V_k = (\mathcal{Q} \setminus S_k) \setminus F_k$ be the skipped tasks that validate successfully. Then*

$$C_{\text{full}} - C_{\text{sched}} = \sum_{Q \in V_k} [C_{\text{search}}(Q) - C_{\text{val}}(Q, \epsilon_{\text{cand}})] - \sum_{Q \in F_k} C_{\text{val}}(Q, \epsilon_{\text{cand}}).$$

*Thus scheduling helps when the full searches avoided on successfully validated tasks outweigh validation overhead, including wasted validation on tasks that later require fallback.*

## References

1. Li, L.; Ma, R.; Guo, Q.; Xue, X.; Qiu, X. BERT-Attack: Adversarial Attack Against BERT Using BERT. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Virtual, 16–20 November 2020; pp. 6193–6202.
2. Joos, S.; Van Hamme, T.; Verheyen, W.; Preuveneers, D.; Joosen, W. Triple Down on Robustness: Understanding the Impact of Adversarial Triplet Compositions on Adversarial Robustness. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 14. [CrossRef]

3. Zhou, H.; Wang, Z.; Wang, H.; Chen, D.; Mu, W.; Zhang, F. Evaluating the Validity of Word-level Adversarial Attacks with Large Language Models. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL 2024*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 4902–4922. [CrossRef]
4. Wawrowski, Ł.; Biczyk, P.; Ślęzak, D.; Sikora, M. Adversarial Attacks Detection Method for Tabular Data. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 112. [CrossRef]
5. Huang, P.S.; Stanforth, R.; Welbl, J.; Dyer, C.; Yogatama, D.; Gowal, S.; Dvijotham, K.; Kohli, P. Achieving Verified Robustness to Symbol Substitutions via Interval Bound Propagation. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 4083–4093. [CrossRef]
6. Jia, R.; Raghunathan, A.; Göksel, K.; Liang, P. Certified Robustness to Adversarial Word Substitutions. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 4129–4142. [CrossRef]
7. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Long Beach, CA, USA, 4–9 December 2017.
8. Shi, Z.; Zhang, H.; Chang, K.W.; Huang, M.; Hsieh, C.J. Robustness Verification for Transformers. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 26 April–1 May 2020.
9. Liu, H.; Zhang, Z.; Zhao, J. Precise Verification of Transformers Through ReLU-Catalyzed Abstraction Refinement. In *Computer Aided Verification*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2026; Volume 16683, pp. 527–550. [CrossRef]
10. Wei, D.; Wu, H.; Wu, M.; Chen, P.Y.; Barrett, C.; Farchi, E. Convex Bounds on the Softmax Function with Applications to Robustness Verification. In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), Valencia, Spain, 25–27 April 2023; Volume 206, pp. 6853–6878.
11. Zhang, Y.; Shen, L.; Guo, S.; Ji, S. GaLileo: General Linear Relaxation Framework for Tightening Robustness Certification of Transformers. *Proc. AAAI Conf. Artif. Intell.* **2024**, *38*, 21797–21805. [CrossRef]
12. Zhang, Y.; Du, T.; Ji, S.; Guo, S. DROWN: Towards Tighter LiRPA-Based Robustness Certification. In Proceedings of the 31st International Conference on Computational Linguistics (COLING), Abu Dhabi, United Arab Emirates, 19–24 January 2025; pp. 6212–6229.
13. Huang, P.; Wei, D.; Isac, O.; Wu, H.; Wu, M.; Barrett, C. Parameterized Abstract Interpretation for Transformer Verification. *Proc. Aaai Conf. Artif. Intell.* **2026**, *40*, 35500–35508. [CrossRef]
14. Feng, B.; Tang, T.; Wang, Y.; Chen, Z.; Wang, Z.; Yang, S.; Xie, Y.; Ding, Y. Faith: An Efficient Framework for Transformer Verification on GPUs. In Proceedings of the USENIX Annual Technical Conference (USENIX ATC), Carlsbad, CA, USA, 11–13 July 2022; pp. 167–182.
15. Ye, M.; Gong, C.; Liu, Q. SAFER: A Structure-Free Approach for Certified Robustness to Adversarial Word Substitutions. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL), Virtual, 5–10 July 2020.
16. Wang, W.; Tang, P.; Lou, J.; Xiong, L. Certified Robustness to Word Substitution Attack with Differential Privacy. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT), Virtual, 6–11 June 2021.
17. Zhang, X.; Hong, H.; Hong, Y.; Huang, P.; Wang, B.; Ba, Z.; Ren, K. Text-CRS: A Generalized Certified Robustness Framework against Textual Adversarial Attacks. In Proceedings of the IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 20–22 May 2024.
18. Katz, G.; Barrett, C.; Dill, D.L.; Julian, K.; Kochenderfer, M.J. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In Proceedings of the International Conference on Computer Aided Verification (CAV), Heidelberg, Germany, 24–28 July 2017.
19. Tjeng, V.; Xiao, K.; Tedrake, R. Evaluating Robustness of Neural Networks with Mixed Integer Programming. In Proceedings of the International Conference on Learning Representations (ICLR), New Orleans, LA, USA, 6–9 May 2019.
20. Bunel, R.; Lu, J.; Turkaslan, I.; Torr, P.H.S.; Kohli, P.; Kumar, M.P. Branch and Bound for Piecewise Linear Neural Network Verification. *J. Mach. Learn. Res.* **2020**, *21*, 1–39.
21. Liao, B.; Cheng, C.H.; Esen, H.; Knoll, A. Are Transformers More Robust? Towards Exact Robustness Verification for Transformers. In *Computer Safety, Reliability, and Security*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2023.
22. Mirman, M.; Gehr, T.; Vechev, M. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In Proceedings of the International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018.
23. Gowal, S.; Dvijotham, K.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelović, R.; Mann, T.; Kohli, P. On the Effectiveness of Interval Bound Propagation for Training Verifiably Robust Models. *arXiv* **2018**, arXiv:1810.12715.

24. Zhang, H.; Weng, T.W.; Chen, P.Y.; Hsieh, C.J.; Daniel, L. Efficient Neural Network Robustness Certification with General Activation Functions. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 2–8 December 2018.
25. Singh, G.; Gehr, T.; Püschel, M.; Vechev, M. An Abstract Domain for Certifying Neural Networks. In Proceedings of the ACM SIGPLAN Symposium on Principles of Programming Languages (POPL), Cascais, Portugal, 13–19 January 2019.
26. Xu, K.; Shi, Z.; Zhang, H.; Wang, Y.; Chang, K.W.; Huang, M.; Kailkhura, B.; Lin, X.; Hsieh, C.J. Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–12 December 2020.
27. Xu, K.; Zhang, H.; Wang, S.; Wang, Y.; Jana, S.; Lin, X.; Hsieh, C.J. Fast and Complete: Enabling Complete Neural Network Verification with Rapid and Massively Parallel Incomplete Verifiers. In Proceedings of the International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
28. Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.J.; Kolter, J.Z. Beta-CROWN: Efficient Bound Propagation with Per-Neuron Split Constraints for Neural Network Robustness Verification. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Virtual, 6–14 December 2021; Volume 34.
29. Ugare, S.; Banerjee, D.; Misailovic, S.; Singh, G. Incremental Verification of Neural Networks. *Proc. ACM Program. Lang.* **2023**, *7*, 1920–1945. [CrossRef]
30. Zhang, G.; Fukuda, K.; Zhang, Z.; Bandara, H.M.N.D.; Chen, S.; Zhao, J.; Sui, Y. Efficient Neural Network Verification via Order Leading Exploration of Branch-and-Bound Trees. In Proceedings of the 39th European Conference on Object-Oriented Programming (ECOOP), Bergen, Norway, 30 June–4 July 2025; Volume 333, pp. 36:1–36:29. [CrossRef]
31. Elsaleh, R.; Davis, L.; Wu, H.; Katz, G. Incremental Neural Network Verification via Learned Conflicts. *arXiv* **2026**, arXiv:2603.12232.
32. Wang, Y.; Yang, Y.; He, D.; He, K. Robustness-Aware Word Embedding Improves Certified Robustness to Adversarial Word Substitutions. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL*; Association for Computational Linguistics: Toronto, ON, Canada, 2023.
33. Zhang, Y.; Albarghouthi, A.; D'Antoni, L. A One-Layer Decoder-Only Transformer Is a Two-Layer RNN: With an Application to Certified Robustness. *arXiv* **2024**, arXiv:2405.17361.
34. Mangal, R.; Păsăreanu, C.S. A Cascade of Checkers for Run-Time Certification of Local Robustness. In Proceedings of the Workshop on Formal Methods for ML-Enabled Autonomous Systems (FoMLAS), Haifa, Israel, 31 July–1 August 2022; pp. 15–28.
35. König, M.; Hoos, H.H.; van Rijn, J.N. Speeding Up Neural Network Robustness Verification via Algorithm Configuration and an Optimised Mixed Integer Linear Programming Solver Portfolio. *Mach. Learn.* **2022**, *111*, 4565–4584. [CrossRef]
36. Maleki, M.; Sidibomma, R.; Adibi, A.; Samavi, R. Cascading Robustness Verification: Toward Efficient Model-Agnostic Certification. *arXiv* **2026**, arXiv:2602.04236.
37. Fischer, M.; Sprecher, C.; Dimitrov, D.I.; Singh, G.; Vechev, M. Shared Certificates for Neural Network Verification. In *Computer Aided Verification (CAV)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13371, pp. 127–148. [CrossRef]
38. Muthukumar, R.; Sulam, J. Adversarial Robustness of Sparse Local Lipschitz Predictors. *SIAM J. Math. Data Sci.* **2023**, *5*, 920–948. [CrossRef]
39. Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C.D.; Ng, A.Y.; Potts, C. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642.
40. Zhang, X.; Zhao, J.; LeCun, Y. Character-Level Convolutional Networks for Text Classification. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Montreal, QC, Canada, 7–10 December 2015.


**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.