AFFINE VERTEX: DIRECTIONAL SOFTMAX BOUNDS FOR TRANSFORMER VERIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer robustness verification requires bounds on attention that remain useful as they propagate through the network. We introduce Affine Vertex, a method that tightens affine softmax bounds while preserving the slopes supplied by an existing verifier. We use exact ranges of weighted softmax outputs to construct a convex lower bound on the residual, the difference between the weighted softmax output and the fixed linear term. Each tangent gives a valid bound, even if the search for a stronger tangent is stopped early. We evaluate these bounds without forming matrices of pairwise interactions between attention scores. For an attention row with K scores and a fixed number of optimization steps, this reduces the work from $O(K^2)$ to $O(K \log K)$ and the working storage from $O(K^2)$ to $O(K)$. On the official vision-transformer benchmark, Affine Vertex certifies 134 of 200 properties, compared with 127 for a matched control under the same computational budget. Experiments on independently trained models also show certification gains. Controlled comparisons demonstrate gains from convexification and further tangent optimization on selected properties. Independent interval checks validate sampled local bounds. These results show that refining softmax bounds can improve transformer verification, although tighter local bounds do not always yield additional certificates.

1 INTRODUCTION

Verification asks whether a model satisfies a requirement for every input within a specified set of allowed changes. For classification, the requirement is that the correct class remains ahead of every competing class. A model is robust to these changes when this requirement holds throughout the allowed set. A verifier establishes this guarantee by bounding the network's behavior over the entire set. Attention makes this difficult because changing one attention score can affect every softmax weight (Vaswani et al., 2017). Bounds must capture this interaction while remaining useful as they propagate through the rest of the network. If the bounds are too loose, the verifier may be unable to prove that the prediction remains correct, even when every allowed change preserves it.

Affine-bound verifiers address this problem by working backward from the prediction being checked (Zhang et al., 2018; Xu et al., 2020; Wei et al., 2023). At an attention operation, the relevant quantity is a weighted combination of softmax outputs. These weights describe how each output affects the downstream objective. An affine lower bound combines the attention scores linearly and adds a constant. Its coefficients describe how the bound changes with each score; together, these coefficients are called the slope. The constant is the intercept and shifts the bound upward or downward. This gives our central question: how much can we improve an existing softmax bound while keeping its slope?

We introduce **Affine Vertex** to raise this lower bound while keeping its coefficients unchanged. Our starting point is CROWN-LSE, which combines CROWN bound propagation with the log-sum-exp-based softmax bounds of Wei et al. (2023). Once the slope is fixed, increasing the intercept shifts the bound upward by the same amount everywhere. This makes the bound tighter, but we can only raise it so far: it must remain below the weighted softmax output for every score vector allowed by the verifier.

To find that limit, we subtract the chosen linear term from the weighted softmax output. We call the resulting difference the **residual**. Each score vector places a restriction on the intercept: the

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

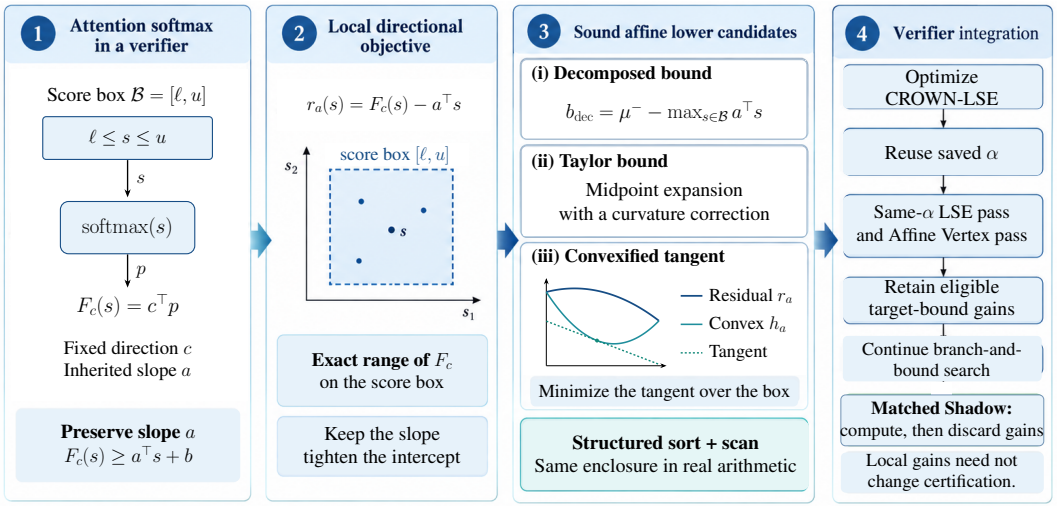


Figure 1: **Affine Vertex overview.** Exact ranges of $F_c(s) = c^\top \mathrm{softmax}(s)$ support conservative bounds on the residual $r_a(s) = F_c(s) - a^\top s$, with $\mu^- = \min_{\mathcal{B}} F_c$. Here s contains attention scores, c weights the softmax outputs, and a, b are the affine slope and intercept; α denotes the verifier's saved relaxation parameters. The strongest candidate supplies the intercept while the slope stays fixed. Curves are schematic. The dense and structured calculations give the same bounds in exact arithmetic, before numerical safeguards are applied. Matched Shadow computes the candidate improvements but does not retain them.

intercept cannot exceed the residual at that point. The verifier supplies a lower and upper limit for each attention score. Together, these intervals form the score box, which contains every possible score vector considered in the local calculation. The smallest residual within this box determines the largest valid intercept. A lower bound on this minimum gives us a safe choice, even if we cannot find the minimum exactly.

We can find the smallest weighted softmax output over the score box exactly. But our goal is to minimize the residual, which also subtracts the fixed linear term. A score vector that makes the softmax output small may also make the subtracted term small, so it need not give the smallest difference. We therefore use the exact softmax range as an ingredient in bounding the residual, rather than as a direct solution for the intercept.

Our main idea is to construct a convex function that stays below the residual throughout the score box. A tangent to this function also stays below it, so the tangent provides a lower bound on the residual. Because the tangent is linear, we can easily find its smallest value over the score box and use that value as a valid intercept. We first evaluate a tangent at the center of the score box, placing each score halfway between its lower and upper limits. We then search for tangent locations that give stronger bounds, retaining the best one found. Every tangent gives a valid bound, even if this search is stopped early. Figure 1 summarizes the construction and its use in the verifier.

The construction builds on established ideas in directional range computation and convex underestimation (Liu & Mendel, 2008; Adjiman et al., 1998), adapting them to the softmax residual. A direct implementation computes curvature bounds using matrices that contain an entry for every pair of attention scores. For a row of K scores, this requires $O(K^2)$ work and storage. We show that sorting and cumulative sums can compute the same bounds without forming these matrices. For a fixed number of tangent-location updates, the complete local algorithm therefore reduces the work per attention row to $O(K \log K)$ and the working storage to $O(K)$, while preserving the same bounds in exact arithmetic.

Our experiments show that refining softmax bounds can improve verification outcomes. Across independently trained transformers with several attention blocks, Affine Vertex certifies additional inputs compared with a matched control that computes the same candidate bounds but does not retain their improvements. On the official vision-transformer competition benchmark, it certifies 134 of

200 properties, compared with 127 for this control under the same budget. These results show that stronger local attention bounds can remain useful after propagation through the full network.

The component studies help explain where the gains come from. We compare the tangent construction with two simpler alternatives: one bounds the two terms of the residual separately, while the other uses a linear approximation with a correction to keep the bound valid. On selected benchmark cases, adding the tangent construction enables certification where these alternatives do not. Searching for stronger tangent locations also produces more successful certification runs on two selected properties than using only the tangent at the center of the score box. On a broader panel, the tested variants certify the same cases. The benefit therefore depends on whether a local improvement is large enough to resolve the full verification problem.

2 Related work

Neural network verification. Linear-relaxation methods such as CROWN propagate bounds through a network to certify its predictions under input perturbations (Zhang et al., 2018). Automatic bound-propagation frameworks extend this approach to broader computational graphs (Xu et al., 2020), while branch-and-bound methods split an inconclusive verification problem into smaller subproblems and compute tighter bounds for each (Shi et al., 2025). Our experiments use the α, β-CROWN framework (Xu et al., 2021; Wang et al., 2021) with GenBaB.

Softmax and transformer relaxations. Previous work has developed bounds that account jointly for the inputs to softmax, including convex and concave relaxations (Wei et al., 2023) and GaLiLeo's multidimensional affine bounds (Zhang et al., 2024). Other transformer-verification methods improve attention products or the bounds supplied to nonlinear operations (Shi et al., 2020; Bonaert et al., 2021; Zhang et al., 2025; Huang et al., 2026). Our work focuses on the weighted softmax expression encountered during backward propagation. For a fixed downstream direction and inherited slope, we seek a stronger intercept over the current score box. This refinement addresses one source of looseness and can complement improvements elsewhere in the network.

Directional ranges and convex underestimation. After exponentiating the scores, the directional softmax range becomes an interval-weighted-average problem. Its sorted-vertex solution is connected to established switch-point methods (Liu & Mendel, 2008). Our convex underestimator likewise builds on the classical αBB approach, which adds a quadratic correction to obtain a convex lower bound (Adjiman et al., 1998; Hladík, 2013). We specialize these ideas to the fixed-slope softmax residual and exploit its structure to compute the required enclosure without pairwise matrices.

3 Affine Vertex: strengthening an existing bound

Affine Vertex strengthens an existing affine bound while preserving its slope. This section establishes two results. First, tangents to a convex function below the residual give valid intercepts at every feasible tangent location, even if the search for stronger tangents is stopped early. Second, the same construction can be evaluated with $O(K \log K)$ work and $O(K)$ working storage per attention row for a fixed number of updates, replacing the quadratic work and storage of the dense formulation. We build the argument from the local intercept problem, explain how convexification produces valid tangent bounds, and then show how to evaluate the construction efficiently.

From attention to a verification objective. Let f denote the model's class-score function, y the correct class of input x, and $\mathcal{X}$ the set of allowed perturbed inputs x'. For classification, a verifier seeks positive lower bounds on $f_y(x') - f_t(x')$ for every $x' \in \mathcal{X}$ and competing class $t \neq y$. If all bounds are positive, robustness is certified; otherwise, they may be inconclusive. Affine Vertex improves the softmax bounds used while propagating these objectives through attention.

An attention row combines value vectors V_j as $o = \sum_j p_j V_j$, where s is the attention-score vector and $p = \mathrm{softmax}(s)$. The vector q describes how the attention output contributes to the verification objective. For fixed value vectors, define $c_j = q^\top V_j$. Then

$$q^\top o = \sum_j (q^\top V_j) p_j = c^\top p.$$

Thus c_j describes how attending to token j affects the objective; it may be positive, negative, or zero. In a full transformer, values also depend on the perturbed input. The surrounding verifier bounds those interactions and supplies the resulting coefficients to the local softmax problem.

The verifier provides a slope a and a finite score box $\mathcal{B} = [\ell, u]$, where ℓ and u bound each score from below and above. We seek an intercept b such that $F_c(s) := c^\top \operatorname{softmax}(s) \geq a^\top s + b$ throughout the box. Subtracting the chosen linear term defines the residual $r_a(s) = F_c(s) - a^\top s$, whose minimum is the strongest admissible intercept:

$$b^*(a) = \min_{s \in \mathcal{B}} \{c^\top \operatorname{softmax}(s) - a^\top s\}. \tag{1}$$

We compute a lower bound on this minimum. Throughout, $c, a \in \mathbb{R}^K$ are fixed locally, $K \geq 1$, and $\ell \leq u$ are finite. A box encloses the scores without assuming probabilistic independence; infinite masking logits are outside these assumptions.

What directional ranges provide. The weighted softmax term has an exact range over the box. After exponentiating the scores, the expression becomes a weighted average: the coefficients c_j are the values being averaged, and the exponentiated scores provide the weights (Liu & Mendel, 2008). Each weight lies within an interval determined by the score bounds. To minimize this average, we favor larger weights for smaller coefficients and smaller weights for larger coefficients. After sorting the coefficients, we test each possible split, assigning upper weight endpoints before the split and lower endpoints after it. There are only $K + 1$ such choices, allowing us to compute the exact minimum μ^- in $O(K \log K)$ work. Applying the same procedure to $-c$ gives the maximum μ^+ (Appendix A.1).

These ranges immediately give a decomposed bound, $b_{\mathrm{dec}} = \mu^- - \max_{s \in \mathcal{B}} a^\top s$. However, the two extrema need not occur at the same scores. The scalar range is exact, but this residual bound can be loose. We therefore construct a bound that treats the two terms together.

Constructing a convex function below the residual. The residual need not be convex, so its tangents need not lie below it. We therefore construct a convex function below the residual and use that function's tangents instead. Following the αBB construction (Adjiman et al., 1998), we add a quadratic correction that stays nonpositive throughout the score box while supplying enough curvature to make the corrected function convex. The following bounds determine how much correction is needed.

Since the linear term has zero second derivative, we only need to bound the curvature of the weighted softmax term, $H(s) = \nabla^2 F_c(s)$. The diagonal bounds describe curvature along each score coordinate, while R_i bounds the total interaction with the other coordinates. We also bound each softmax probability over the score box. Together with the range of the weighted softmax output, these probability intervals let us bound the Hessian entries. In particular, we obtain:

$$H_{ii}^- \leq H_{ii}(s) \leq H_{ii}^+, \qquad \sum_{j \neq i} |H_{ij}(s)| \leq R_i \quad (s \in \mathcal{B}). \tag{2}$$

The diagonal shift compensates for this off-diagonal curvature:

$$d_i \geq \max\{0, R_i - H_{ii}^-\}, \qquad h_a(s) = r_a(s) + \frac{1}{2} \sum_i d_i (\ell_i - s_i)(u_i - s_i). \tag{3}$$

The correction is nonpositive inside the box, so $h_a \leq r_a$. It also makes the Hessian diagonally dominant with nonnegative diagonal, hence positive semidefinite. Thus h_a is convex. Appendix A.2 derives the curvature bounds and proves that h_a is a convex underestimator.

Turning a tangent into a valid affine bound. A tangent to a convex function lies below it. Because h_a lies below the residual, its tangent does too. Minimizing the tangent is simple: each coordinate chooses a box endpoint according to the sign of its coefficient. We call the point where the tangent is evaluated its anchor. For any anchor $z \in \mathcal{B}$, let $g = \nabla h_a(z)$. Minimizing this tangent over the box gives

$$b_{\tan}(z) = h_a(z) + \sum_i \begin{cases} g_i(\ell_i - z_i), & g_i \geq 0, \\ g_i(u_i - z_i), & g_i < 0. \end{cases} \tag{4}$$

The resulting affine bound is valid without finding an optimal anchor.

Theorem 3.1 (Fixed-slope affine certificate)**.** *Under the finite-box and fixed-coefficient assumptions above, suppose Eq. (2) holds and $D = \mathrm{diag}(d)$ satisfies Eq. (3). For every $z \in \mathcal{B}$, $b_{\tan}(z) \le b^*(a)$. Consequently, $F_c(s) \ge a^\top s + b_{\tan}(z)$ for every $s \in \mathcal{B}$.*

Proof. For every $s \in \mathcal{B}$, convexity gives

$$r_a(s) \ge h_a(s) \ge h_a(z) + \nabla h_a(z)^\top (s - z) \ge b_{\tan}(z).$$

Taking the minimum of r_a proves the intercept bound; adding $a^\top s$ gives the valid affine bound. □

Choosing anchors and retaining the strongest candidate. We start at the midpoint $m = (\ell + u)/2$ and search for stronger tangents by updating their location within the score box. In our experiments, the full variant uses eight updates and retains the strongest of the nine tangents evaluated, including the initial midpoint tangent. Every candidate remains valid regardless of the number of updates. Projection keeps each proposed anchor inside the box, where its tangent is valid.

The convexified tangent need not give the strongest bound on every box. We therefore also retain the decomposed bound b_{dec} and a Taylor bound b_{Tay}. The Taylor bound starts from a linear expansion of the residual at the midpoint and uses a curvature bound to control the remainder. Taking the maximum of these valid lower bounds preserves validity.

We call the combination of the decomposed and Taylor bounds D+T. VERTEX-0 adds the convexified tangent at the midpoint; VERTEX-8 also evaluates the eight updated anchors. Appendices A.4 and A.3 give the Taylor formula and anchor-update rule, respectively.

Computing the enclosure without pairwise matrices. We now have a valid intercept construction. The remaining challenge is to compute its curvature bounds efficiently. A direct implementation forms matrices with an entry for every pair of attention scores, requiring quadratic work and storage. These pairwise quantities have structure we can exploit: each off-diagonal row bound can be expressed as a weighted sum of distances between coefficients and a threshold. After sorting the coefficients, cumulative sums let us evaluate these expressions without constructing the matrices.

The coordinate probability bounds $p_i^- = \min_{s \in \mathcal{B}} p_i(s)$ and $p_i^+ = \max_{s \in \mathcal{B}} p_i(s)$ can likewise be computed by accumulating contributions from the other coordinates. We use log-sum-exp scans that exclude the current coordinate; these are called exclusive scans. Appendix A.5 gives the identities. The faster calculation changes how we compute the bound, but preserves its value in exact arithmetic.

Theorem 3.2 (Structured enclosure equivalence)**.** *For fixed finite c, a and a nonempty finite score box, Eqs. (26) and (27) give the same curvature bounds R_i and probability extrema $p_i^{\pm}$ as the dense calculation in exact arithmetic. With the same diagonal formulas, choice of diagonal correction, and deterministic anchor-update rule, the dense and structured rules, before numerical safeguards are applied, produce the same affine intercept on the given score box for any fixed update count $T \ge 0$.*

The proof covers ties, coefficient shifts, and $K = 1$ (Appendix A.5). For $K \ge 2$, the complete bound calculation for one attention row uses $O(K \log K + TK)$ work and $O(K)$ working storage when anchors are processed one at a time, hence $O(K \log K)$ work for fixed T. The dense version uses $O(K^2 + TK)$ work and $O(K^2)$ storage. These are per-row arithmetic bounds, not measured runtime or whole-verifier complexity. Appendix A.5 accounts for all operations and batching.

Algorithm 1 brings these steps together: compute the ranges and curvature enclosure, evaluate the candidate intercepts, and retain the strongest one for the inherited slope. Here $t_i = \mu^- + \mu^+ - c_i$ is the threshold used in the weighted-distance calculation, and Λ is the curvature bound used to choose the anchor-update step (Appendix A). The anchor update normally takes a projected-gradient step. When the curvature bound is too small to use this step, the implementation chooses a box endpoint according to the gradient signs. Appendix B gives the precise rule and numerical threshold.

Algorithm 1 One-row Affine Vertex with structured enclosures

Require: Fixed c, a, finite box $[\ell, u]$, update count $T \in \{0, 8\}$
1: Compute μ^-, μ^+ by sorted directional sweeps (Proposition A.1).
2: Compute p_i^-, p_i^+ by exclusive log-sum-exp scans (Eq. (27)).
3: Sort c with $v = p^+$; form weighted prefix/suffix sums and binary-search each t_i to compute R_i.
4: Compute $H_{ii}^{\pm}$, D, Λ, b_{dec}, b_{Tay}; set $z \leftarrow (\ell + u)/2$, $b \leftarrow \max\{b_{\mathrm{dec}}, b_{\mathrm{Tay}}\}$.
5: **for** $k = 0, \ldots, T$ **do**
6: Evaluate $h_a(z)$, $g = \nabla h_a(z)$; set $b \leftarrow \max\{b, b_{\tan}(z)\}$.
7: **if** $k < T$ **then**
8: Update z by the projected or endpoint anchor-update rule.
9: **end if**
10: **end for**
11: **Return** b for the unchanged slope a.

Appendix B records the numerical allowances and conservative conversion used in the implementation.

We next explain how the verifier uses the resulting intercepts.

4 Using Affine Vertex within a verifier

We combine the row bounds from Section 3 and check their effect on the network output. This refinement follows selected bound-optimization calls during branch-and-bound search.

Combining bounds across attention rows. Each output constraint and search subproblem can involve attention rows across queries and heads. For this complete group $\mathcal{R}$, Affine Vertex keeps each slope a_r and computes an intercept b_r. We compare their sum with the inherited constant B_{base} for the same group:

$$B_{\mathrm{ret}} = \max\left\{B_{\mathrm{base}}, \sum_{r \in \mathcal{R}} b_r\right\}. \tag{5}$$

The stored local comparisons use this same complete-group rule. Upper bounds follow by reversing the objective's sign and applying the lower-bound construction.

Deciding when to retain the completed bound. A stronger softmax bound may not improve the final output bound, so we compare these too. The host first uses α-CROWN (Xu et al., 2021) with LSE softmax bounds (Wei et al., 2023) to optimize its relaxation parameters. Let L_{opt} denote the resulting lower bound for the output constraint and search subproblem.

We save the relaxation parameters α and hold them fixed during two additional calculations. One uses LSE and returns L_{same}; the other uses Affine Vertex at the selected attention softmax nodes and returns L_V. We retain the Vertex result only if

$$L_V > L_{\mathrm{same}}, \qquad L_V > L_{\mathrm{opt}}, \qquad \text{at least one local objective improved in that pass.} \tag{6}$$

Otherwise, we return L_{opt}. A nonfinite candidate is an error and cannot provide a certificate. Selecting among sound bounds on the same output constraint and domain preserves soundness (Appendix A.6).

Two controls distinguish the benefit of retaining improvements from the cost of computing candidates.

Matched Shadow: retaining improvements. Shadow performs the same extra passes, candidate construction, comparisons, and output allocation as Affine Vertex, but restores L_{opt}.

Native CROWN-LSE: the fixed-budget comparison. The native control runs CROWN-LSE without the extra passes, keeping its official configuration and resource budget unchanged. Memory failures count as unsuccessful verifications. This measures completion under the specified budget.

Stronger bounds can change subsequent search decisions. A local improvement therefore does not guarantee more certificates before the time limit.

5 EXPERIMENTAL RESULTS

We test whether stronger attention bounds improve certification across transformer blocks and on the official benchmark, then examine which components contribute. All certification and component experiments use the structured implementation described in Section 3, with a 100-second budget; unsuccessful verifications remain in the totals. Appendix C gives the protocols.

Does the improvement survive multiple transformer blocks? We train Fashion-MNIST (Xiao et al., 2017) transformers with one to four attention-residual blocks, without normalization. We train five models at each depth using different random seeds. Architecture details appear in Appendix C.

For each model and input perturbation radius ϵ, measured in the ℓ_∞ norm, we evaluate the first 100 examples in the evaluation set. Each depth/radius combination therefore contains 500 examples. An example counts as certified only if the model predicts it correctly and the verifier establishes robustness; clean prediction errors and verifier failures remain in the denominator.

Table 1: Certification without normalization. Rates are percentages. Paired gains are full-minus-Shadow means $\pm$ sample SD across five training seeds, in percentage points (pp), not confidence intervals. Full denotes structured VERTEX-8; G-style denotes our independent integration of GaLiLeo-style bounds.

Blocks	ϵ	Native	G-style	Shadow	Full	Paired gain (pp)
1	$1/255$	64.2	64.2	64.2	64.4	$+0.2 \pm 0.4$
1	$2/255$	56.2	56.2	56.2	57.2	$+1.0 \pm 1.0$
2	$1/255$	47.4	48.6	47.4	52.0	$+4.6 \pm 3.6$
2	$2/255$	18.2	19.0	18.2	23.0	$+4.8 \pm 2.9$
3	$1/255$	37.6	39.0	37.6	44.8	$+7.2 \pm 5.4$
3	$2/255$	7.0	7.0	7.0	10.0	$+3.0 \pm 3.3$
4	$1/255$	31.2	33.0	31.6	37.0	$+5.4 \pm 3.0$
4	$2/255$	3.0	3.2	3.2	3.4	$+0.2 \pm 0.4$

At $1/255$, full Vertex improves certified accuracy over Shadow by 4.6, 7.2, and 5.4 percentage points at depths two, three, and four. Every seed at depths three and four has a positive net gain (Appendix D, Figure 2). These results show that the improvement can survive propagation through several blocks. They do not show that deeper models necessarily benefit more, because the models also differ in accuracy and verification difficulty.

We also evaluate mean-only normalization, which subtracts the feature mean without dividing by the input-dependent standard deviation. In this separate study, full Vertex certifies 174/240 examples versus 163/240 for Shadow, while retaining every property certified by Shadow. That study evaluates only correctly classified examples and is reported separately in Appendix E.

Does the method help on the official benchmark? The VNN-COMP 2023 ViT benchmark contains 100 properties for each of two CIFAR-10 (Krizhevsky, 2009) models: two blocks/five tokens and three blocks/seventeen tokens, both with three heads and BatchNorm. Its attack- and verification-filtered population measures benchmark coverage rather than ordinary test-set certified accuracy (Brix et al., 2023). We keep the official models, properties, native configuration, and budget unchanged.

In the matched comparison, full Vertex certifies 134 properties and Shadow certifies 127. Both methods certify 58/100 properties on the smaller model; full Vertex certifies 76/100 versus Shadow's 69/100 on the larger model. The seven additional certificates replace timeouts, while retaining every property certified by Shadow. This is a gain of 3.5 percentage points over Shadow.

In earlier matched benchmark checks, the two methods had identical initial bounds; the gains observed in those checks therefore arose during subsequent search.

In the separate native comparison, full Vertex certifies 134 properties and native CROWN-LSE certifies 113. All twenty-one additional Vertex successes occur on properties where the native run exhausts memory, which counts as unsuccessful verification under the fixed competition configuration.

Table 2: Official-property outcomes with structured full Vertex on RTX 5090 GPUs. Every row has 200 properties; all error counts are zero. The matched and native comparisons were conducted separately. GaLiLeo-style is an independent implementation, not the published verifier's official code. Memory exhaustion (OOM) is unsuccessful verification under the fixed configuration. The $2/255$ radius is derived, not the official radius.

Comparison / radius	Arm	Certified	Unknown/timeout	OOM
Matched / $1/255$	Shadow	127	73	0
	Full Vertex	134	66	0
Native / $1/255$	Native CROWN-LSE	113	55	32
	Full Vertex	134	66	0
	GaLiLeo-style	127	72	1
Matched / $2/255$	Shadow	0	200	0
	Full Vertex	0	200	0

All 113 native certificates are retained; the 55 native timeouts and the remaining eleven native OOM cases become Vertex timeouts.

The native comparison measures completion under the official configuration; Shadow measures the effect of retaining candidate improvements. These comparisons answer different questions, so their gains should not be added together. Neither comparison establishes intrinsic memory efficiency.

At the derived radius $2/255$, both full Vertex and Shadow time out on all 200 properties.

Which parts of the construction contribute? We test the contribution of convexification and anchor optimization in stages. First, we compare the decomposed candidate alone with decomposed and Taylor candidates together (D+T). Adding a convexified midpoint tangent gives VERTEX-0; further anchor updates give the full variant, VERTEX-8. Comparing these last two variants tests the added value of optimizing the anchor.

Every variant retains the inherited baseline. Shadow computes the full candidate but discards eligible improvements to the completed output bound. Each variant computes only the bounds included in that variant.

An initial comparison on four prespecified properties, 0, 50, 100, and 150, gives no separation: all five arms certify all four properties. To understand the observed benchmark gains, we then examine the seven properties on which full Vertex outperformed Shadow in the matched official comparison. Table 3(a) reports two separate comparisons, each with three repetitions at solver seed 100.

The component comparisons show where a midpoint tangent is sufficient and where further optimization helps. On properties 142, 154, and 181, midpoint and full certify every repetition, while Shadow and D+T certify none. These cases support the contribution of convexification without requiring further anchor updates. Property 192 is already certified by D+T, so the simpler bounds suffice there.

Further anchor optimization helps on two selected properties. On property 163, full certifies all three repetitions while midpoint certifies none. On property 175, full certifies all three while midpoint certifies one. Property 147 remains budget-sensitive, with one full success in each comparison. Because these cases were selected for gains, they do not estimate how often each component helps (Appendix F).

Replays at identical inputs also show that anchor optimization can tighten local bounds independently of changes in the search trajectory (Appendix H).

Do the component differences extend to a broader sample? On a broader panel of 32 properties, the tested variants certify exactly the same cases at both solver seeds: eight official properties and thirteen from deeper models (Table 3b). Full improves local bounds in all 42 runs in which the verifier evaluates the Affine Vertex refinement, out of 64 runs in total. These improvements produce no additional certificates.

Table 3: Component certification results using structured evaluation. (a) Seven benchmark properties certified by full Vertex but not Shadow; entries count three repetitions at seed 100. Comparisons A and B are separate. (b) Certified-property counts at each seed, 100 and 101; all variants certify identical sets. Deeper-model examples are clean-correct; n denotes group size.

(a) Selected-case attribution: certified repetitions

	A: Shadow / D+T / full			B: midpoint / full	
Property	Shadow	D+T	VERTEX-8	VERTEX-0	VERTEX-8
142	0/3	0/3	3/3	3/3	3/3
147	0/3	0/3	1/3	1/3	1/3
154	0/3	0/3	3/3	3/3	3/3
163	0/3	0/3	3/3	0/3	3/3
175	0/3	0/3	3/3	1/3	3/3
181	0/3	0/3	3/3	3/3	3/3
192	0/3	3/3	3/3	3/3	3/3

(b) Hash-selected panel: certified properties at each solver seed

Family	n	Shadow	D+T	VERTEX-0	VERTEX-8
Official	12	8	8	8	8
No normalization, depth 3	10	6	6	6	6
No normalization, depth 4	10	7	7	7	7

The panel contains twelve official properties and twenty correctly classified examples from depth-three and depth-four models. A fixed hash rule selects cases without consulting verification outcomes within the eligible pools. The panel contains none of the seven observed benchmark gains. Its deeper-model cases also have no full-versus-Shadow certification differences in the earlier benchmark records. The null finding therefore applies to this selected population; it does not establish that the variants are equivalent in general. Appendix F gives the selection procedure and exclusions. Appendix H analyzes the unsuccessful runs' returned bounds.

Checking the computed bounds. We independently check sampled bounds using interval arithmetic, which encloses numerical values while accounting for rounding. These checks validate sampled bounds returned by the dense and structured implementations without modifying their outputs. This evidence depends on valid incoming score boxes and the interval library; it does not validate every operation in a network certificate (Appendix G).

6 LIMITATIONS

The affine intercept remains conservative even though the directional range is exact: score boxes can include unreachable combinations, and the host must bound uncertain values and other operations. Successful certification results cover models without normalization, with mean-only normalization, and with BatchNorm; standard LayerNorm remains outside the validated experimental scope.

The unchanged native comparison does not establish the best baseline configuration or intrinsic memory efficiency. Our experiments evaluate the structured implementation on models with short attention rows; its practical scaling at larger token counts remains to be established. The complexity result does not by itself imply an end-to-end speedup.

7 CONCLUSION

Affine Vertex strengthens existing affine softmax bounds by refining their intercepts while preserving their slopes. Exact directional ranges support a convex construction whose tangents remain valid without optimizer convergence. A structured evaluation computes the same bounds with lower work and storage requirements. Experiments demonstrate additional certificates in the tested settings and identify cases where convexification and further anchor optimization contribute. Together, these results show how local softmax structure can strengthen the bounds used in transformer verification.

REPRODUCIBILITY STATEMENT

Appendices A–B provide the assumptions, proofs, and numerical safeguards underlying the method. Appendices C–H describe the experimental protocols and additional analyses. We provide code for reproducing the reported results in the supplementary material.

AI USE STATEMENT

The author developed the research idea and proofs. AI tools provided coding, debugging, writing, and editing support, including polishing the exposition and presentation of the proofs. The author reviewed and verified the resulting material and retains responsibility for the research questions, methods, proofs, experiments, interpretation, and final content.

ETHICS STATEMENT

This work studies mathematical bounds for transformer verification using public benchmark datasets. It does not involve human-subject experiments or the collection of private data. The method supports robustness verification under specified perturbations and assumptions; its certificates should not be interpreted as guarantees of overall model safety.

REFERENCES

C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A global optimization method, αBB, for general twice-differentiable constrained NLPs: I. theoretical advances. *Computers & Chemical Engineering*, 22(9):1137–1158, 1998. doi: 10.1016/S0098-1354(98)00027-1.

Gregory Bonaert, Dimitar I. Dimitrov, Maximilian Baader, and Martin T. Vechev. Fast and precise certification of transformers. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 466–481, 2021. doi: 10.1145/3453483.3454056.

Christopher Brix, Stanley Bak, Changliu Liu, and Taylor T. Johnson. The fourth international verification of neural networks competition (VNN-COMP 2023): Summary and results. *arXiv preprint arXiv:2312.16760*, 2023.

Milan Hladík. The effect of hessian evaluations in the global optimization αBB method. *arXiv preprint arXiv:1307.2791*, 2013.

Pei Huang, Dennis Wei, Omri Isac, Haoze Wu, Min Wu, and Clark Barrett. Parameterized abstract interpretation for transformer verification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 35500–35508, 2026. doi: 10.1609/aaai.v40i42.40860.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Feilong Liu and Jerry M. Mendel. Aggregation using the fuzzy weighted average as computed by the Karnik–Mendel algorithms. *IEEE Transactions on Fuzzy Systems*, 16(1):1–12, 2008. doi: 10.1109/TFUZZ.2007.896229.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. In *International Conference on Learning Representations*, 2020.

Zhouxing Shi, Qirui Jin, Zico Kolter, Suman Jana, Cho-Jui Hsieh, and Huan Zhang. Neural network verification with branch-and-bound for general nonlinearities. In *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 315–335. Springer, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Shiqi Wang, Huan Zhang, Kaidi Xu, Xue Lin, Suman Jana, Cho-Jui Hsieh, and J. Zico Kolter. Beta-CROWN: Efficient bound propagation with per-neuron split constraints for neural network robustness verification. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://papers.nips.cc/paper/2021/hash/fac7fead96dafceaf80c1daffeae82a4-Abstract.html.

Dennis Wei, Haoze Wu, Min Wu, Pin-Yu Chen, Clark Barrett, and Eitan Farchi. Convex bounds on the softmax function with applications to robustness verification. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 6853–6878. PMLR, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Automatic perturbation analysis for scalable certified robustness and beyond. In *Advances in Neural Information Processing Systems*, 2020.

Kaidi Xu, Huan Zhang, Shiqi Wang, Yihan Wang, Suman Jana, Xue Lin, and Cho-Jui Hsieh. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2011.13824.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems*, 2018.

Yunruo Zhang, Lujia Shen, Shanqing Guo, and Shouling Ji. GaLiLeo: General linear relaxation framework for tightening robustness certification of transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 21797–21805, 2024.

Yunruo Zhang, Tianyu Du, Shouling Ji, and Shanqing Guo. DROWN: Towards tighter LiRPA-based robustness certification. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 6212–6229. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-main.415/.

A MATHEMATICAL CONSTRUCTION AND PROOFS

This appendix gives the full mathematical arguments behind the method. Following Section 3, we begin with a directional range and a first affine bound, construct the convexified tangent bound, and then add the complementary Taylor candidate. We next establish structured equivalence and complexity, before proving the aggregation rules used in Section 4. Table 4 collects the notation used throughout.

The arguments in this appendix assume exact arithmetic. Section 5 and Appendix G describe numerical checks on sampled local problems. Throughout, $K \geq 1$, $\ell, u, c, a \in \mathbb{R}^K$, and $\ell \leq u$ coordinatewise. The box $\mathcal{B} = [\ell, u]$ is nonempty and finite. Coefficients may be positive, negative, or zero; repeated coefficients and zero-width intervals are allowed. Both c and a are fixed during a local calculation.

Table 4: Local notation. Query/head rows are indexed by r when several row objectives are aggregated.

Symbol	Definition
$K, \mathcal{B}$	Number of keys and finite score box $[\ell, u] \subset \mathbb{R}^K$
$p(s), F_c(s)$	$\mathrm{softmax}(s)$ and $c^\top p(s)$
c, a	Fixed downstream coefficient vector and inherited score slope
$r_a(s), b^*(a)$	$F_c(s) - a^\top s$ and its minimum over $\mathcal{B}$
μ^-, μ^+	Exact minimum and maximum of F_c over $\mathcal{B}$
p_i^-, p_i^+	Exact coordinate probability extrema over $\mathcal{B}$
m, w	Midpoint $(\ell + u)/2$ and half-width $(u - \ell)/2$
v_i, M, Δ, t_i	p_i^+, $\mu^- + \mu^+$, $\mu^+ - \mu^-$, and $M - c_i$ for the structured radius
W_L, C_L, W_R, C_R	Weighted prefix/suffix sums at the split $c_j < t_i$
$E_i(x)$	Exclusive log-sum-exp $\log \sum_{j \neq i} e^{x_j}$
$H(s)$	Hessian $\nabla^2 F_c(s)$, also the Hessian of r_a
H_{ii}^-, H_{ii}^+, R_i	Diagonal enclosures and absolute off-diagonal row-sum bound
$D = \mathrm{diag}(d)$	Nonnegative shift satisfying $d_i \geq \max(0, R_i - H_{ii}^-)$
$h_a(s)$	$r_a(s) + \frac{1}{2} \sum_i d_i(\ell_i - s_i)(u_i - s_i)$
$z_k, g(z_k), \Lambda$	Feasible anchor, gradient of h_a, and shifted-Hessian eigenvalue upper bound
T	Number of anchor updates: zero for VERTEX-0, eight for VERTEX-8
$b_{\mathrm{dec}}, b_{\mathrm{Tay}}, b_{\tan}$	Decomposed, Taylor, and convexified-tangent intercepts
$B_{\mathrm{base}}, B_{\mathrm{ret}}$	Inherited and retained biases of a complete local objective group
$L_{\mathrm{opt}}, L_{\mathrm{same}}, L_V$	Completed target/domain bounds from the optimized, detached LSE, and Vertex passes

A.1 DIRECTIONAL RANGES AND THE FIRST AFFINE BOUND

The range calculation becomes simpler after exponentiating the scores: the weighted softmax output is then a ratio of two linear functions. The proof uses the optimal ratio to identify which interval endpoint each coordinate should choose.

The map $s \mapsto y = e^s$ is a coordinatewise bijection from $\mathcal{B}$ to the positive box $\mathcal{B}_y = \prod_i [L_i, U_i]$, where $L_i = e^{\ell_i}$ and $U_i = e^{u_i}$. Thus

$$F_c(s) = R_c(y) := \frac{c^\top y}{\mathbf{1}^\top y}, \qquad \mathbf{1}^\top y > 0. \tag{7}$$

Minimizing this weighted average favors greater weight on smaller coefficients. At an optimum, an entry is set to its upper endpoint when its coefficient is below the optimal average, and to its lower endpoint when above it.

We do not know that average in advance, but sorting the coefficients lets us test every possible split. Sort $c_{(1)} \leq \cdots \leq c_{(K)}$ and permute the endpoints accordingly. The candidate at split q and the

resulting range are as follows.

$$\tau_q = \frac{\sum_{i\le q} c_{(i)}U_{(i)} + \sum_{i>q} c_{(i)}L_{(i)}}{\sum_{i\le q} U_{(i)} + \sum_{i>q} L_{(i)}}, \qquad 0 \le q \le K. \tag{8}$$

Proposition A.1 (Directional score-box range)**.** *For the fixed direction and finite box above,* $\mu^- := \min_{s\in\mathcal{B}} F_c(s) = \min_q \tau_q$ *and* $\mu^+ := \max_{s\in\mathcal{B}} F_c(s) = -\min_{s\in\mathcal{B}} F_{-c}(s)$*. A sorted prefix/suffix sweep evaluates each range in* $O(K \log K)$ *time and* $O(K)$ *working storage.*

Proof of Proposition A.1. Continuity on the compact box gives a minimizer y^* and minimum $\rho^* = R_c(y^*)$. Define $\phi(y) = \sum_i (c_i - \rho^*)y_i$. For all $y \in \mathcal{B}_y$,

$$R_c(y) - \rho^* = \frac{\phi(y)}{\mathbf{1}^\top y} \ge 0, \tag{9}$$

with equality at y^*. Therefore the minimum of the linear function ϕ over the box is zero. A coordinatewise minimizer selects U_i whenever $c_i < \rho^*$ and L_i whenever $c_i > \rho^*$. Coordinates with $c_i = \rho^*$ do not affect ϕ and may use either endpoint. After sorting the coefficients, one minimizing vertex has upper endpoints on a prefix and lower endpoints on the suffix. Its ratio is one of the $K+1$ values τ_q in Eq. (8). Every τ_q is feasible and hence at least ρ^*; at least one equals ρ^*. This proves the minimum formula. Replacing c by $-c$ proves the maximum formula. Sorting costs $O(K \log K)$; prefix sums of U_i, c_iU_i and suffix sums of L_i, c_iL_i evaluate the $K+1$ ratios in $O(K)$ further work and storage. □

This is the sorted switch rule for a normalized interval-weighted average (Liu & Mendel, 2008). It is not an exact solver for r_a when $a \neq 0$. Any scalar lower bound valid throughout the same box is at most μ^-. Information about which score combinations the network can actually reach may allow tighter bounds on a smaller set. The theorem concerns the full score box.

The range immediately gives a first intercept. Since $F_c(s) \ge \mu^-$ everywhere, we can subtract the largest possible value of $a^\top s$ on the box. This separates the two terms of the residual:

$$b_{\text{dec}} = \mu^- - \sum_i \begin{cases} a_i u_i, & a_i \ge 0, \\ a_i \ell_i, & a_i < 0. \end{cases} \tag{10}$$

For every $s \in \mathcal{B}$, $F_c(s) \ge \mu^-$ and $a^\top s \le \max_{z\in\mathcal{B}} a^\top z$. Subtraction proves Eq. (10).

This bound minimizes the two terms separately. We next construct a bound that treats the residual as a whole.

A.2 Curvature bounds and convexification

To make the residual convex, we need bounds on the diagonal Hessian entries and on the sum of the absolute off-diagonal entries in each row. We derive these from the softmax derivatives, using probability intervals and the directional range to bound the terms that vary with the scores.

Let $p = \text{softmax}(s)$ and $\mu = c^\top p$. The softmax derivative is $\partial_j p_i = p_i(\mathbf{1}_{i=j} - p_j)$. Differentiating the weighted sum yields

$$\partial_i F_c = p_i(c_i - \mu),\ H_{ii} \quad = p_i(c_i - \mu)(1 - 2p_i),\ H_{ij} = -p_i p_j(c_i + c_j - 2\mu), \qquad i \neq j. \tag{11}$$

The monotonicity of p_i in its own score and in every other score gives

$$p_i^- = \frac{e^{\ell_i}}{e^{\ell_i} + \sum_{j\neq i} e^{u_j}}, \qquad p_i^+ = \frac{e^{u_i}}{e^{u_i} + \sum_{j\neq i} e^{\ell_j}}. \tag{12}$$

Each endpoint is attained at the indicated score vertex, including when intervals are degenerate.

For the diagonal, let $q(p) = p(1-2p)$ and define

$$q_i^- = \min\{q(p_i^-), q(p_i^+)\}, \qquad q_i^+ = q\big(\text{clip}(1/4, p_i^-, p_i^+)\big). \tag{13}$$

The quadratic q is concave with maximum at $1/4$, so these enclose its range on the probability interval. Let $e_i^- = c_i - \mu^+$ and $e_i^+ = c_i - \mu^-$. The minimum and maximum of

$$\{q_i^- e_i^-, q_i^- e_i^+, q_i^+ e_i^-, q_i^+ e_i^+\} \tag{14}$$

give H_{ii}^- and H_{ii}^+. These $e_i^\pm$ are coefficient deviations, distinct from the probability weights $v_i = p_i^+$ and the diagonal shifts d_i. For off-diagonal terms an enclosure is

$$A_{ij}^+ = p_i^+ p_j^+ \max\{|c_i + c_j - 2\mu^-|, |c_i + c_j - 2\mu^+|\}, \quad R_i = \sum_{j \neq i} A_{ij}^+. \tag{15}$$

The nonnegative probability factors are bounded above coordinatewise, and the absolute affine function of μ attains its interval maximum at an endpoint. Dependencies among p_i, p_j, μ are ignored only to enlarge the enclosure. Equations (2) therefore hold in real arithmetic.

The curvature bounds determine a diagonal correction that makes the residual convex while keeping it below the original objective.

Proposition A.2 (Convex residual underestimator)**.** *Under Eq. (2), choose d and h_a as in Eq. (3), and set $D = \mathrm{diag}(d)$.*

Then $h_a(s) \leq r_a(s)$ throughout $\mathcal{B}$, and h_a is convex on $\mathcal{B}$.

Proof. For each $s \in \mathcal{B}$, the diagonal of $H(s) + D$ obeys

$$H_{ii}(s) + d_i \geq H_{ii}^- + \max\{0, R_i - H_{ii}^-\} \geq R_i \geq \sum_{j \neq i} |H_{ij}(s)|. \tag{16}$$

The matrix is symmetric and diagonally dominant with nonnegative diagonal; its Gershgorin lower endpoints are nonnegative, so it is positive semidefinite. The added quadratic in Eq. (3) contributes exactly D to the Hessian and is nonpositive on the box. Thus $h_a \leq r_a$ and h_a is convex. This is the diagonal αBB construction with $\alpha_i = d_i/2$ (Adjiman et al., 1998; Hladík, 2013).

□

We now have a convex function below the residual. Its tangents provide the valid affine bounds established next.

A.3 Tangent certificates and anchor selection

A tangent to the convex underestimator lies below the residual. Minimizing that tangent over the score box gives the intercept in Theorem 3.1.

Proof of Theorem 3.1. For any $s, z \in \mathcal{B}$, convexity yields

$$r_a(s) \geq h_a(s) \geq h_a(z) + \nabla h_a(z)^\top (s - z). \tag{17}$$

Writing $g = \nabla h_a(z)$, minimization of the last affine expression chooses $s_i = \ell_i$ for $g_i \geq 0$ and $s_i = u_i$ for $g_i < 0$. Its minimum is precisely $b_{\tan}(z)$. Hence every tangent candidate is at most $b^*(a)$. Adding $a^\top s$ proves the stated affine inequality. □

Choosing feasible anchors. Validity holds at every feasible anchor; the remaining choice is which anchors give stronger bounds. The simplest anchor is the midpoint $m = (\ell + u)/2$. It is always feasible and needs no search. To seek a stronger tangent, the full variant starts at $z_0 = m$ and takes projected-gradient steps on h_a, keeping the best tangent encountered.

We use these curvature bounds to choose the step size. The following Λ bounds the largest Hessian eigenvalue of h_a, and g is its gradient:

$$\Lambda = \max\{0, \max_i (H_{ii}^+ + d_i + R_i)\}, \qquad g_i(z) = p_i(z)(c_i - F_c(z)) - a_i + d_i(z_i - m_i), \tag{18}$$

When the curvature bound is positive, we use its reciprocal as the step size and project the result back into the box: $z_{k+1} = \Pi_{\mathcal{B}}(z_k - g(z_k)/\Lambda)$. Otherwise, we choose a feasible box endpoint. Appendix B specifies the numerical safeguards and the near-zero-curvature case.

Eight updates evaluate nine anchors, including the midpoint. Retaining the strongest candidate prevents additional anchors from weakening the local bound when the inputs and convexifier are unchanged. Their cost may matter under a verification time limit.

Differentiating h_a gives the gradient in Eq. (18). The upper Gershgorin endpoint of row i of its Hessian is at most $H_{ii}^{+} + d_i + R_i$. Hence the stated Λ bounds every eigenvalue above. When $\Lambda > 0$, it supplies a Lipschitz-gradient bound on the box for the projected-gradient proposal. Projection keeps each proposed anchor inside the box, where its tangent is valid. The endpoint rule also produces a feasible anchor. Neither rule needs to find an optimal anchor for the bound to be valid.

The convexified tangent need not be the strongest candidate on every box. We therefore retain a complementary bound based directly on the residual.

A.4 The Taylor candidate and combining bounds

The Taylor candidate approximates the residual at the midpoint and uses a lower curvature bound to control the remainder. This provides another valid intercept to compare with the decomposed and tangent candidates.

Let $w = (u - \ell)/2$ be the half-width, and let p_i^{-}, p_i^{+} be the coordinate probability extrema over $\mathcal{B}$. A suitable curvature bound is obtained from

$$\theta_i = \mathrm{clip}(1/2, p_i^{-}, p_i^{+}), \quad \beta = \min\{1/2,\, 2 \max_i \theta_i(1-\theta_i)\}, \quad \kappa = (\mu^{+} - \min_i c_i)\beta. \tag{19}$$

The Hessian satisfies $\nabla^2 F_c(s) \succeq -\kappa I$ on the box. Taylor's lower remainder bound therefore gives

$$b_{\mathrm{Tay}} = r_a(m) - \sum_i |\nabla r_a(m)_i| w_i - \frac{\kappa}{2} \sum_i w_i^2. \tag{20}$$

To prove the Taylor bound, take any direction $\xi \in \mathbb{R}^K$ and differentiate $F_c(s + t\xi)$ twice. With $\bar{\xi} = p^{\top}\xi$, collecting the softmax derivative terms gives

$$\xi^{\top} H(s) \xi = \sum_i p_i (c_i - \mu)(\xi_i - \bar{\xi})^2. \tag{21}$$

One can also verify this identity by expanding the square: its three terms are $\sum_i p_i(c_i - \mu)\xi_i^2$, $-2(p^{\top}\xi)\sum_i p_i(c_i - \mu)\xi_i$, and zero, respectively, because $\sum_i p_i(c_i - \mu) = 0$. These agree with Eq. (11). Since $c_i - \mu \geq -(\mu^{+} - \min_j c_j)$,

$$\xi^{\top} H(s) \xi \geq -(\mu^{+} - \min_j c_j)\, \xi^{\top}\big(\mathrm{diag}(p) - pp^{\top}\big)\xi. \tag{22}$$

The absolute row sum of the covariance matrix is $2p_i(1 - p_i)$. Its spectral norm is consequently at most $2\max_i p_i(1 - p_i) \leq 1/2$. Maximizing $p_i(1 - p_i)$ on $[p_i^{-}, p_i^{+}]$ gives the θ_i and β in Eq. (19). It follows that $H(s) \succeq -\kappa I$ throughout $\mathcal{B}$.

The segment from m to s stays inside the convex box. Taylor's integral formula for r_a, whose Hessian is H, gives

$$\begin{aligned} r_a(s) &= r_a(m) + \nabla r_a(m)^{\top}(s - m) + \int_0^1 (1 - t)(s - m)^{\top} H(m + t(s - m))(s - m)\, dt \\ &\geq r_a(m) + \nabla r_a(m)^{\top}(s - m) - \frac{\kappa}{2}\|s - m\|_2^2 \\ &\geq r_a(m) - \sum_i |\nabla r_a(m)_i| w_i - \frac{\kappa}{2}\sum_i w_i^2. \end{aligned} \tag{23}$$

This is Eq. (20). The bound remains valid for $K = 1$ and for zero-width coordinates.

Each candidate bounds the same residual on the same box, so we can retain their maximum. For a finite nonempty anchor set $\mathcal{Z}$,

$$b_{\mathcal{Z}} = \max\{b_{\mathrm{dec}}, b_{\mathrm{Tay}}, \max_{z \in \mathcal{Z}} b_{\tan}(z)\} \leq b^{*}(a). \tag{24}$$

The midpoint variant VERTEX-0 uses $\mathcal{Z} = \{m\}$; the full variant VERTEX-8 uses the midpoint and eight updated anchors. Both retain the decomposed and Taylor candidates as well.

Validity of candidate combination. The decomposed, Taylor, and tangent candidates each lower-bound the same residual minimum $b^*(a)$. Their maximum therefore also lower-bounds $b^*(a)$, proving Eq. (24). □

This completes the bound construction. We next show how to evaluate it without forming pairwise matrices.

A.5 Structured evaluation, equivalence, and complexity

The remaining question is how much the curvature construction costs. A direct calculation appears to need every pair of scores. For $i \neq j$, the softmax Hessian entry is $H_{ij} = -p_i p_j(c_i + c_j - 2F_c(s))$. Using $v_i = p_i^+$, the dense calculation bounds the off-diagonal contribution in row i by

$$R_i = v_i \sum_{j \neq i} v_j \max\{|c_i + c_j - 2\mu^-|, |c_i + c_j - 2\mu^+|\}. \tag{25}$$

The pairwise expression has useful structure: the larger of the two distances to the doubled range endpoints, $2\mu^-$ and $2\mu^+$, equals the distance to their center plus their half-separation. Set $M = \mu^- + \mu^+$, $\Delta = \mu^+ - \mu^- \geq 0$, and $t_i = M - c_i$. Applying $\max(|x - \Delta|, |x + \Delta|) = |x| + \Delta$ gives

$$R_i = v_i \left[\sum_j v_j |c_j - t_i| - v_i |c_i - t_i| + \Delta\Big(\sum_j v_j - v_i\Big) \right]. \tag{26}$$

The main sum is now a weighted sum of distances from t_i. After sorting c, split the coefficients at $c_j < t_i$. Prefix and suffix sums of v_j and $v_j c_j$ give it as $t_i W_L - C_L + C_R - t_i W_R$, where W sums the weights and C sums the weighted coefficients on either side. Binary search locates each split. Subtracting the self term excludes $j = i$; ties contribute zero.

Probability extrema can also be computed without a pairwise matrix. Each coordinate needs a sum over all the other coordinates. Exclusive prefix and suffix log-sum-exp scans provide these leave-one-out sums in linear work. With $E_i(x) = \log \sum_{j \neq i} e^{x_j}$ and $\mathrm{lae}(x, y) = \log(e^x + e^y)$, the endpoints are

$$p_i^- = e^{\ell_i - \mathrm{lae}(\ell_i, E_i(u))}, \qquad p_i^+ = e^{u_i - \mathrm{lae}(u_i, E_i(\ell))}. \tag{27}$$

For $K = 1$, the excluded sum is empty, $E_1 = -\infty$, and both probabilities are one. These rearrangements reduce computation while preserving the same curvature and probability bounds. Theorem 3.2 states exactly what is preserved.

Exact-arithmetic equivalence. The structured calculation must reproduce the enclosure used by the dense construction, not just produce a similarly sized bound. The proof proceeds in three steps: rewrite each curvature bound R_i as a weighted sum of distances, compute each probability endpoint using a sum over the other coordinates, and carry these equalities through the remaining bound calculation.

The radius identity only requires finite $c \in \mathbb{R}^K$, nonnegative $v \in \mathbb{R}^K$, and finite $\mu^- \leq \mu^+$. For its use as a Hessian enclosure, v_i must upper-bound $p_i(s)$ and $[\mu^-, \mu^+]$ must contain $F_c(s)$ throughout the box. Before numerical allowances are added, these are the exact extrema defined above. No normalization of v is assumed: coordinate probability maxima generally occur at different score vectors.

Proof of Theorem 3.2. Let $M = \mu^- + \mu^+$ and $\Delta = \mu^+ - \mu^- \geq 0$. For any real x, if $x \geq 0$ then $|x + \Delta| = |x| + \Delta$ and $|x - \Delta| \leq |x| + \Delta$. The case $x < 0$ follows by symmetry. Thus

$$\max\{|x - \Delta|, |x + \Delta|\} = |x| + \Delta. \tag{28}$$

With $x = c_i + c_j - M = c_j - t_i$, where $t_i = M - c_i$, the two arguments on the left are $c_i + c_j - 2\mu^+$ and $c_i + c_j - 2\mu^-$. Consequently,

$$\begin{aligned} R_i &= v_i \sum_{j \neq i} v_j \big(|c_j - t_i| + \Delta\big) \\ &= v_i \left[\sum_j v_j |c_j - t_i| - v_i |c_i - t_i| + \Delta\Big(\sum_j v_j - v_i\Big) \right]. \end{aligned} \tag{29}$$

The subtraction removes both parts of the self contribution: $v_i|c_i - t_i|$ from the absolute-deviation sum and Δv_i from the weight sum.

Let π sort c nondecreasingly, and write $c_{(j)} = c_{\pi(j)}$ and $v_{(j)} = v_{\pi(j)}$. For a split $q \in \{0, \dots, K\}$, define

$$\begin{aligned} W_L(q) &= \sum_{j \leq q} v_{(j)}, & C_L(q) &= \sum_{j \leq q} v_{(j)} c_{(j)}, \\ W_R(q) &= \sum_{j > q} v_{(j)}, & C_R(q) &= \sum_{j > q} v_{(j)} c_{(j)}. \end{aligned} \tag{30}$$

Empty sums are zero. Binary search gives $q_i = \#\{j : c_j < t_i\}$. Splitting the absolute value at t_i yields

$$\begin{aligned} \sum_j v_j |c_j - t_i| &= \sum_{j \leq q_i} v_{(j)} (t_i - c_{(j)}) + \sum_{j > q_i} v_{(j)} (c_{(j)} - t_i) \\ &= t_i W_L(q_i) - C_L(q_i) + C_R(q_i) - t_i W_R(q_i). \end{aligned} \tag{31}$$

Terms with $c_j = t_i$ are zero, so placing ties on either side gives the same value. The same permutation must carry c and v. Together, Eqs. (29) and (31) give exactly the dense radius without forming any pairwise array.

For coordinate probabilities, define $\mathrm{lae}(x, y) = \log(e^x + e^y)$, with $\log 0 = -\infty$. For finite $x_1, \dots, x_K$, form

$$\begin{aligned} P_0 &= -\infty, & P_j &= \mathrm{lae}(P_{j-1}, x_j), \quad j = 1, \dots, K, \\ S_{K+1} &= -\infty, & S_j &= \mathrm{lae}(x_j, S_{j+1}), \quad j = K, \dots, 1. \end{aligned} \tag{32}$$

Induction gives $P_j = \log \sum_{h \leq j} e^{x_h}$ and $S_j = \log \sum_{h \geq j} e^{x_h}$. Therefore

$$E_i(x) = \mathrm{lae}(P_{i-1}, S_{i+1}) = \log \sum_{j \neq i} e^{x_j}. \tag{33}$$

Substitution in Eq. (27) reproduces Eq. (12). The value $-\infty$ represents the logarithm of an empty sum; it is not an infinite input score or an attention mask.

Both implementations now have the same $p_i^\pm$, $\mu^\pm$, and R_i. Applying the same diagonal interval formulas gives identical $H_{ii}^\pm$. In particular, before adding numerical allowances, the selection

$$d_i^0 = \max(0, R_i - H_{ii}^-), \qquad \Lambda^0 = \max\{0, \max_i (H_{ii}^+ + d_i^0 + R_i)\} \tag{34}$$

produces the same shift and smoothness bound. Any other identical deterministic selection from these quantities has the same conclusion. The decomposed and Taylor values also coincide. Starting at the same midpoint, induction over a common deterministic anchor policy gives identical anchors and tangent values, including the same anchor-update rule when the curvature bound is zero. Their retained maximum is therefore identical. This proves equivalence of the formulas for a single score box before numerical allowances are added, not exactness of the residual optimum or equality of floating-point evaluations. □

Coefficient shifts must also shift the range. For a common offset γ, let $\widetilde{c} = c - \gamma \mathbf{1}$. Since $\mathbf{1}^\top p = 1$, $F_{\widetilde{c}} = F_c - \gamma$, so the consistent range is $\widetilde{\mu}^\pm = \mu^\pm - \gamma$. Hence

$$\widetilde{M} = M - 2\gamma, \qquad \widetilde{\Delta} = \Delta, \qquad \widetilde{t}_i = t_i - \gamma, \qquad \widetilde{c}_j - \widetilde{t}_i = c_j - t_i. \tag{35}$$

The radius and coefficient deviations $c_i - \mu^{\pm}$ are invariant. The implementation uses $\gamma = c_1$ when computing R_i. It subtracts the same offset from both range endpoints before forming the center. If centering were applied to the entire residual calculation, its intercept would have to be translated back: $r_{a,\tilde{c}} = r_{a,c} - \gamma$, so a lower intercept for the centered residual becomes one for the original residual after adding γ. Centering only inside the radius routine needs no intercept translation. These identities do not guarantee cancellation-free finite-precision evaluation.

Singletons and degenerate boxes. For $K = 1$, both exclusive sums are empty, $p_1^- = p_1^+ = 1$, and the exact directional range is $\mu^- = \mu^+ = c_1$. The radius, Hessian diagonal, and minimal unpadded shift are zero. The residual is $c_1 - a_1 s_1$; its tangent already yields the exact intercept $c_1 - \max_{s_1 \in [\ell_1, u_1]} a_1 s_1$. The identity also holds when any subset of score intervals has zero width. No strict ordering of coefficients, positive box width, or nonzero slope is needed.

Work and storage. Having established that the two evaluations produce the same mathematical bound, we now account for the work and storage required. We include initialization and every anchor evaluation. This avoids attributing the sorting cost alone to a procedure that also computes probability bounds, curvature bounds, and tangent candidates.

Complexities count unit-cost arithmetic, comparisons, and elementary functions at fixed precision. Prefix/suffix operations are work-efficient scans, and sorting is assumed to use $O(K \log K)$ comparisons and $O(K)$ auxiliary storage. These are arithmetic-operation and scalar-storage bounds, not bit complexity for arbitrary-precision interval arithmetic or guarantees about a particular device's elapsed time.

Directional-range sweeps and the radius sort each use $O(K \log K)$ work. The exclusive probability scans, weighted prefix/suffix sums, diagonal intervals, shift, smoothness bound, midpoint, and decomposed/Taylor initialization each use $O(K)$ work. The K binary searches use $O(K \log K)$ work in total. Each of the $T + 1$ tangent evaluations and T projected updates costs $O(K)$. Every stage uses only linear-size arrays.

The radius routine sorts the coefficients once. The directional-range calculation may sort the coefficients separately; reusing a single ordering is not required for the stated complexity bound. The scalar log-domain fallback also uses prefix/suffix reductions of linear size. Checking inputs, converting numerical precision, adding residual allowances, and evaluating the residual at the midpoint each require at most $O(K)$ additional work and storage per row. The local calculation uses one score box and does not construct or evaluate candidate subdivisions.

Including initialization, a bound valid uniformly for $K \geq 1$ is

$$O\big(K \log(K+1) + (T+1)K\big) \text{ work}, \qquad O(K) \text{ working storage}. \tag{36}$$

For $K \geq 2$ this is $O(K \log K + TK)$, compared with $O(K^2 + TK)$ for the dense reference. The constant-update variant $T = 8$ evaluates nine anchors, not eight; $T = 0$ still evaluates the midpoint and constructs the full convexifier. Retaining only the current anchor and best tangent value makes storage independent of T. Explicitly recording all anchors for diagnostics adds $O((T + 1)K)$ storage; the stated storage bound assumes that anchors are processed one at a time and that no automatic-differentiation graph is retained.

For N row objectives of length K held in memory at once, the corresponding bounds are $O(N[K \log(K+1) + (T+1)K])$ work and $O(NK)$ working storage, excluding optional traces and host state. With heterogeneous lengths the work sums over rows. Complete-group scalar aggregation is linear in its row count. Processing output constraints in batches limits memory use while keeping all query and head rows for each constraint together. Query rows, heads, specifications, layers, and branch-and-bound domains determine how many such calls the host performs. Its attention products, intermediate-bound construction, optimization, and search are not assigned the complexity of the bound calculation for one row.

A.6 Combining rows within the verifier

The preceding results concern one attention row. To use them within the verifier, we must preserve validity when combining rows and handling upper bounds. The following arguments support the integration in Section 4.

For rows $r \in \mathcal{R}$, summing valid inequalities gives

$$\sum_{r\in\mathcal{R}} F_{c_r}(s_r) \geq \sum_{r\in\mathcal{R}} a_r^\top s_r + \sum_{r\in\mathcal{R}} b_r. \tag{37}$$

It holds over the product of row boxes and therefore over any correlated reachable subset. If the inherited aggregate B_{base} is valid for the same slopes and domain, Eq. (5) is valid. For an upper affine bound, apply the lower construction to $-F_c(s) - (-a)^\top s = -r_a(s)$ and negate its intercept.

After propagating the local bounds to the network output, the verifier compares the completed lower bounds. Taking the maximum of sound completed lower bounds on the same scalar target and domain proves the validity of the target-level selection in Eq. (6). This statement does not compare separately optimized search trajectories.

B Numerical evaluation

This appendix describes the floating-point evaluation of the construction. The allowances below are engineering safeguards, not universal rounding-error bounds. Appendix G independently checks sampled bounds returned by the implementation under stated assumptions.

The local calculation converts its inputs to float64. The surrounding verifier keeps its relaxation parameters in float32. The scalar sweep subtracts $\max_i u_i$ before exponentiating. This common rescaling leaves the ratios unchanged and prevents positive exponential overflow. If the range calculation encounters underflow or a nonfinite candidate, it switches to a calculation in the log domain; a common shift alone would not resolve underflow or cancellation in signed sums. Coordinate probability ranges use log-sum-exp calculations; the structured calculation uses the exclusive scans in Eq. (32).

Let ϵ_{mach} denote machine epsilon at the working precision. To allow for numerical error in the curvature calculation, we add

$$16\epsilon_{\text{mach}}(1 + R_i + |H_{ii}^-|) \tag{38}$$

to $\max(0, R_i - H_{ii}^-)$. We also subtract an allowance from each candidate lower bound, scaled by the coefficients and score bounds:

$$\delta = 10^{-9}\left(1 + \sum_i |c_i| + \sum_i |a_i| \max\{|\ell_i|, |u_i|\}\right). \tag{39}$$

Complete row groups are combined before conservative conversion to the verifier's lower precision. This conversion does not correct any enclosure error in preceding operations.

Anchor-update safeguards. The projected anchor update in Section 3 uses the curvature bound to set the step size. The implementation takes this branch when $\Lambda > \epsilon_{\text{mach}}$ and protects the divisor by the smallest positive normal value. Otherwise it proposes ℓ_i for $g_i \geq 0$ and u_i for $g_i < 0$. Both rules produce feasible anchors; these choices affect tightness, not the validity of a tangent at a feasible anchor.

B.1 Numerical allowances for structured evaluation

Structured evaluation follows the formulas of Theorem 3.2 but combines terms in a different order. We add allowances to its probability and radius calculations to reduce the risk of numerical error. The probability intervals computed by the scans are widened by $\varepsilon_p = 16K\epsilon_{\text{mach}}$: the lower endpoint is reduced by ε_p and clipped below at zero, and the upper is increased and clipped above at one.

Let $\widehat{v}$ denote the resulting upper values. In the radius routine, $\widetilde{c} = c - c_1\mathbf{1}$ and the range endpoints are shifted consistently, as in Eq. (35). If $W = \sum_j \widehat{v}_j$, the additional radius allowance is

$$\varepsilon_{R,i} = 64K\epsilon_{\text{mach}}\left[1 + \widehat{v}_i W\left(\max_j |\widetilde{c}_j| + |\widetilde{t}_i| + |\Delta|\right)\right]. \tag{40}$$

The radius is clipped below at zero after adding this allowance. We retain the same diagonal-shift and residual allowances. We also retain downward rounding when converting the summed lower

bound to the verifier's lower precision. Signed weighted sums and self-term subtraction can still suffer cancellation, and an extreme finite input can overflow intermediate arithmetic.

Theorem 3.2 establishes equality in exact arithmetic before numerical allowances are added. With these allowances and floating-point rounding, the dense and structured calculations can produce different diagonal corrections, anchors, and intercepts. Appendix G reports checks on sampled outputs and on the diagonal corrections used to compute them. The outputs are checked without modification.

B.2 Conditions for interpretation

The local guarantee relies on information supplied by the rest of the verifier. The following conditions make that boundary explicit.

The exact-arithmetic guarantee assumes that the incoming boxes contain all possible scores and that the local coefficients and slopes are fixed. Masking with infinite logits requires a separate sound convention and is not covered by replacing infinity with an arbitrary finite number. A bound calculation that returns an invalid or nonfinite candidate cannot certify a property. The surrounding verifier must still provide valid baseline intercepts and score intervals. It must also correctly bound uncertain attention values and products, combine bounds downstream, and handle the other nonlinear operations. Checking sampled local intercepts does not validate the floating-point calculations throughout the full network.

C Experimental methods

The experiments evaluate certification on two model populations and use component comparisons to distinguish the sources of improvement.

Models and evaluation populations. The official benchmark contains 200 properties on two vision transformers. The Fashion-MNIST study without normalization uses five training seeds at each of four depths and two perturbation radii. Each model has 16 tokens, hidden width 16, two attention heads, and a ReLU MLP of width 64 in each block. Each depth/radius result includes the first 100 evaluation examples for each model. The mean-only-normalization study uses three seeds at each depth and twenty correctly classified examples per model (Appendix E).

The selected-case study evaluates seven benchmark properties with three repetitions at solver seed 100. The broader study evaluates 32 properties at solver seeds 100 and 101 (Appendix F). Repeating a solver run or testing several inputs from one checkpoint does not provide additional independently trained models.

Verifier and computational resources. All Affine Vertex certification experiments use structured evaluation, including the component comparisons and mean-only-normalization study. Dense evaluation is retained as a reference for the separate numerical checks. All studies use RTX 5090 GPUs. The benchmark comparisons use α, β-CROWN (Xu et al., 2021; Wang et al., 2021) with GenBaB. The study without normalization assigns four CPU threads per process; the component studies assign one. Methods compared on a component property share the same physical GPU, software, model, and configuration.

The component implementation processes batches of 32 output specifications, whereas the benchmark implementation does not impose this batch limit on the Affine Vertex calculation. Both combine all query and head rows belonging to each objective. The native benchmark configuration remains unchanged.

Budget and outcomes. All comparisons use a configured 100-second solver limit; an iteration already in progress may return after that limit. Unknown outcomes, timeouts, memory exhaustion, and errors count as uncertified. Certification rates retain every example in the specified evaluation population. Timing comparisons restricted to successful runs do not establish a general speedup.

Paired comparisons. Each comparison pairs the same model, input property, and perturbation radius. Benchmark totals and repeated component evaluations remain separate. In earlier official

matched checks, both methods had identical initial bounds for all 200 properties and all 1,800 target constraints. Shadow computes the same candidates but does not retain their improvements. For the local comparisons, we reevaluate stored problems with the same coefficients, slopes, score boxes, and row grouping. This fixes the local bound calculation, but not the relaxation state of the complete network.

Returned-bound summaries. Root bounds are computed before branch-and-bound splits the problem. For models without normalization, we report the first finite final bound, falling back to a root bound if no finite final bound is available. The mean-only study uses the finite optimized root bound, falling back to the initial root bound. The unsuccessful-component analysis uses the finite bound returned by branch-and-bound. These quantities may describe different stopping stages; a finite bound alone does not establish certification.

D ADDITIONAL RESULTS WITHOUT NORMALIZATION

This appendix explains how we summarize variation across independently trained models and compare returned bounds. These summaries supplement the certification rates in the main text; they do not change which examples count as certified.

The five seeds at each depth use the same first-100-example evaluation rule. Table 1 gives certification rates for all four methods and the variation in the Vertex-minus-Shadow gain across training seeds. For seed s, let q_s^V and q_s^S be the percentages certified by full Vertex and Shadow. We first compute the gain within each seed, then summarize those five gains as

$$\bar{g} = \frac{1}{5}\sum_{s=0}^{4}(q_s^V - q_s^S), \qquad s_g = \sqrt{\frac{1}{4}\sum_{s=0}^{4}(q_s^V - q_s^S - \bar{g})^2}. \tag{41}$$

The SD describes variation across trained models; it is not a standard error or confidence interval. Each seed contributes 100 examples, so averaging the five seed rates gives the same result as pooling all 500 examples.

Table 5: Full-minus-Shadow returned-bound statistics on examples for which all four methods return finite bounds. Means first average within each seed, then equally across five seeds. The 95% interval resamples those five means; positive/tied/negative counts use tolerance 10^{-9}. Requiring finite bounds only from Vertex and Shadow gives the same counts and displayed means. Negative bound differences are not certification losses.

Depth	ϵ	Finite pairs	Mean gain	Median gain	95% interval	+/tie/−
1	$1/255$	344	$+0.0019$	$+0.0000$	$[+0.000, +0.004]$	24/320/0
1	$2/255$	344	$+0.0152$	$+0.0000$	$[+0.006, +0.025]$	30/312/2
2	$1/255$	362	$+0.3401$	$+0.0000$	$[+0.117, +0.646]$	128/227/7
2	$2/255$	362	$+0.2108$	$+0.0268$	$[-0.116, +0.455]$	237/95/30
3	$1/255$	358	$+0.7301$	$+0.0100$	$[+0.416, +1.123]$	196/151/11
3	$2/255$	328	$+1.1000$	$+0.0961$	$[+0.527, +1.805]$	296/14/18
4	$1/255$	366	$+0.6628$	$+0.0174$	$[+0.205, +1.244]$	249/97/20
4	$2/255$	335	$+0.5716$	$+0.0426$	$[+0.395, +0.753]$	322/5/8

Whether a run provides a finite bound and whether it certifies an input are separate questions. A run that ends in memory exhaustion or an error may still have an earlier finite root bound. That bound does not make the property certified. Incorrect clean predictions and verifier failures remain in the certification denominator.

Across 2,872 verification calls on correctly classified examples per method, OOM/error counts are 7/73 for full Vertex, 13/73 for Shadow, 536/79 for native CROWN-LSE, and 12/73 for the GaLiLeo-style integration. These call totals cover multiple depths and radii and are not an independent pooled benchmark.

As a sensitivity check, we repeat the bound comparison using only the requirement that Vertex and Shadow return finite values. The other comparators need not be finite. For every depth and radius,

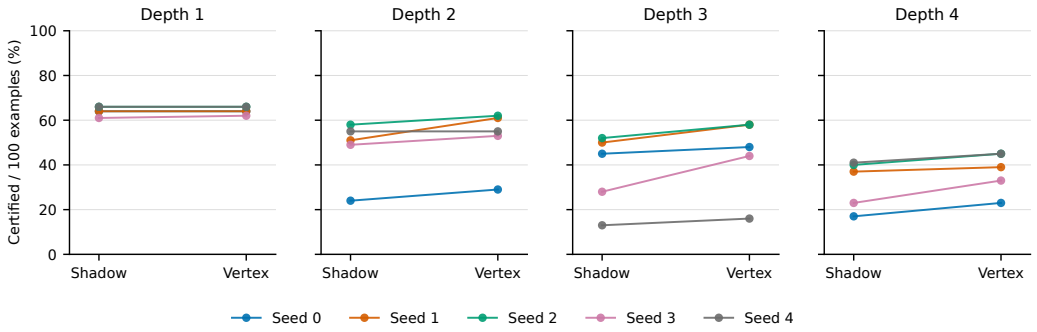


Figure 2: Certification rates at $1/255$ for each training seed. Each endpoint uses all 100 evaluation examples for that checkpoint. Lines pair Shadow and full Vertex within a seed; they do not represent a trajectory across depths. Coincident outcomes overlap.

this check gives the same number of finite pairs and the same mean difference, averaged across seeds, to four decimal places as requiring finite bounds from all four methods. Matching summaries do not establish that the retained examples are identical.

The returned-bound distributions are skewed: many per-seed medians are zero despite positive means, and scales vary across checkpoints. Table 5 includes decreases as well as increases; for example, Vertex returns a smaller bound in thirty depth-two comparisons at $2/255$. Separate searches can follow different paths and stop at different stages, so the differences do not measure local improvements at a fixed verifier state or guarantee a larger returned bound on every property.

Memory failures in native CROWN-LSE affect which examples have finite bounds from all four methods; certification rates retain the full evaluation population.

E Results with mean-only normalization

This study evaluates transformers with affine mean-only normalization using the same structured Affine Vertex implementation as the main experiments.

For feature vector $v \in \mathbb{R}^d$, mean-only normalization is

$$\mathrm{MeanOnly}(v) = \gamma \odot \left(v - \frac{1}{d} \sum_{j=1}^{d} v_j \mathbf{1} \right) + \beta, \tag{42}$$

with fixed learned scale and bias during verification. It omits division by the input-dependent standard deviation and is affine. This setting is distinct from both no normalization and standard LayerNorm; simplified normalization also appears in Shi et al. (2020).

The twelve Fashion-MNIST models use depths one through four and training seeds zero through two. They have sixteen 7×7 patches, hidden width sixteen, two heads, and ReLU-MLP width sixty-four.

The fixed training recipe for the mean-only study uses the first 12,000 training images, eight AdamW epochs, batch size 256, learning rate 0.0015, weight decay 0.0001, and gradient clipping at 1.0.

Evaluation uses a fixed ordering of 2,000 test examples. The first twenty correctly classified examples per checkpoint form the verification panel, so the reported rates describe robustness among correctly classified examples. These training and evaluation settings apply only to the mean-only study.

Across the 240 paired properties, full Vertex certifies 174 and Shadow certifies 163, with no property certified by Shadow alone. All other outcomes are unknown; root bounds are finite and no verifier errors occur.

All 240 properties are evaluated on RTX 5090 GPUs.

Table 6: Mean-only study at $1/255$. Each depth has sixty properties on correctly classified examples, twenty per seed. Gain compares finite optimized root bounds, using the initial root bound when needed. SD is across three seed means, not a confidence interval.

Blocks	Mean gain $\pm$ seed SD	Bound pos./tie/neg.	Full Vertex	Shadow
1	0.000295 ± 0.000511	3/57/0	57/60	57/60
2	0.190057 ± 0.223004	15/45/0	48/60	46/60
3	0.493936 ± 0.353405	34/26/0	41/60	36/60
4	1.485309 ± 0.794865	46/14/0	28/60	24/60
All	0.542399 ± 0.247023	98/142/0	174/240	163/240

To summarize bound improvements without treating each input as an independent trained model, we first average within each checkpoint. For depth d and training seed s, let Δ_{dsi} be the Vertex-minus-Shadow difference in the root bound selected by this rule for example i. The table uses

$$\overline{\Delta}_{ds} = \frac{1}{20}\sum_{i=1}^{20}\Delta_{dsi}, \qquad \overline{\Delta}_{d} = \frac{1}{3}\sum_{s=0}^{2}\overline{\Delta}_{ds}, \qquad \mathrm{SD}_d = \sqrt{\frac{1}{2}\sum_{s=0}^{2}(\overline{\Delta}_{ds} - \overline{\Delta}_{d})^2}. \tag{43}$$

The overall summary first averages eighty examples within each seed. The comparison tolerance is 10^{-9}.

At depth four the median gain is 0.1914, well below the mean, and seed one has a positive mean gain without another certification. The increase in mean gain with depth therefore does not by itself show that depth causes larger certification gains. Some checkpoints select overlapping images, so 240 pairs are not 240 independent model replications.

F Contribution of the bound components

The component experiments distinguish the contribution of convexification from that of optimizing the tangent location. Selected cases explain observed gains; a broader panel examines whether certification differences also occur outside those cases.

F.1 Selected-case comparisons

The selected properties are 142, 147, 154, 163, 175, 181, and 192, exactly the seven properties certified by full Vertex but not Shadow in the matched benchmark. Comparison A evaluates Shadow, D+T, and full Vertex three times each at solver seed 100. Methods share the same physical RTX 5090, one CPU thread, software, configuration, and initial constraint vector within each property; execution order rotates.

Table 3 reports the outcomes. Comparison A supports a contribution from convexification beyond the simpler candidates on selected properties. Because these cases were selected for their observed gains, they do not estimate how often convexification helps across the benchmark.

Comparison B independently evaluates midpoint and full Vertex on the same seven properties, with three repetitions per method. This tests the additional contribution of anchor optimization; Table 3 shows improved certification counts on two selected properties. Comparison A contains 63 runs and comparison B contains 42; their repetitions are not pooled. Local-bound differences are examined separately in Appendix H.

F.2 Broader-panel comparison

The broader panel contains six properties from each official model and two correctly classified examples from each depth-three and depth-four checkpoint across five training seeds. Properties are selected deterministically by hashing their identifiers, without consulting verification outcomes within the eligible pools. The official pool excludes the four prespecified pilot cases (0, 50, 100, 150) and the three gains known at selection (147, 154, 181). The four additional gains (142, 163, 175, 192) are outside the selected panel.

Shadow, D+T, midpoint, and full Vertex are evaluated at solver seeds 100 and 101, giving 256 evaluations of 32 properties. Each property's initial constraint vector agrees across methods at both seeds. All variants certify exactly the same eight official, six depth-three, and seven depth-four properties at each seed. The methods also agree within each combination of training seed and depth.

The selected deeper-model examples also show no full-versus-Shadow certification difference in the benchmark evaluation. The null finding therefore applies to this panel and does not establish equivalence across the full benchmark. Solver repetitions and examples within a checkpoint are not independent trained-model replications.

G INDEPENDENT VALIDATION OF LOCAL BOUNDS

We check each sampled intercept against a lower bound computed independently for the same local problem. The intercept returned by the implementation is left unchanged. The checker treats the stored floating-point inputs as exact values. It uses outward rounding to construct a lower bound L on the sum of residuals for a complete group, allowing each row to vary over its score box. The condition $b \leq L$ validates the returned aggregate intercept b under the checker's assumptions. If $L < b$, the check is inconclusive because the reference may be too loose. The checker uses mpmath 1.3.0 interval arithmetic and the construction proved above; neither the library nor the host verifier is machine-checked here.

Table 7: Numerical-validation coverage. Every listed output passes the check at 128-bit precision without being lowered. Counts denote complete groups or synthetic cases, not independent network properties. Structured checks reuse the eighty stored local problems from the dense checks, so they are not new samples. The final column distinguishes checks of the recorded diagonal corrections from checks using corrections constructed independently.

Implementation	Population	Cases	Rows	Shift/anchor evidence
Dense	Four prespecified properties	22	834	Independent sufficient shift; feasible reconstructed anchors
Dense	Synthetic stress inputs	64	128	Independently enclosed residual outputs
Dense	Successful properties 154/181	18	918	Actual recorded shifts and anchors; row and aggregate checks
Dense	Four failed official properties	40	–	Independent aggregate lower-reference checks
Structured	80 captures + 32 synthetic cases	112	2416	Actual supplied diagonal shifts and emitted outputs

Several rows belong to each group, and several groups can come from the same property. All checks remain conditional on valid incoming boxes and the interval library, and do not cover every applied cut, its full proof dependency chain, baseline validity, or downstream accumulation.

Validation populations. The dense synthetic cases cover float32/float64, $K \in \{2, 3, 4, 16\}$, degenerate/narrow/wide boxes, common shifts of ± 1000, and coefficient scales from 10^{-6} to 10^{6}.

For the prespecified properties, validation uses an independently sufficient diagonal shift and feasible reconstructed anchors. It validates the returned bound, but does not check the original intermediate diagonal corrections and anchors. The smallest difference between the independent reference and the returned intercept is approximately 4.20×10^{-8}.

A second population tests the actual diagonal shifts and anchors on successful properties 154 and 181 under the same configured budget. Both row outputs and aggregate candidates validate; twelve aggregates exceed their inherited baseline. The supplied shifts are not increased for validation.

Analytic controls. Analytic controls check the directional enclosure against all eight vertices of a three-coordinate box, a constant-softmax residual with nonzero linear slope, and acceptance/rejection of valid/deliberately invalid intercepts. A midpoint control checks that zero updates still constructs the convexifier and can improve on D+T.

Structured implementation. The structured algorithm computes the same enclosure in exact arithmetic, but its different order of operations and additional numerical allowances require a separate implementation check. On the tested RTX 5090, the structured and dense calculations return bitwise-identical aggregate intercepts for all eighty stored groups. The separate interval suite in Table 7 checks the diagonal corrections used by the implementation and the bounds it returns.

The interval checks here provide complementary evidence about sampled local outputs. They do not establish universal floating-point equality with the dense implementation or a general rounding-error guarantee.

H Local bound strength and unresolved certification

Local tightening and successful verification are different outcomes. We examine whether anchor optimization improves the same local problem, how tightly that problem can be bounded, and whether improved search bounds cross the certification threshold.

H.1 Does anchor optimization tighten local bounds?

Each variant receives identical coefficients c, slopes a, and score bounds ℓ, u. Contributions from all query/head rows in each objective group are summed before comparison with the inherited aggregate, as in Eq. (5). This fixes the local problem; the comparison is not between final network bounds from separate searches.

Table 8: Local improvements after retaining the inherited baseline. Groups are nested within properties, not independent network examples. The two comparisons improve on the same number of groups within each population.

Population	Groups	Full $>$ midpoint	Midpoint $>$ D+T
Four prespecified properties	22	15	15
Two successful properties	18	12	12
Four unsuccessful properties	40	8	8

Full Vertex improves some local bounds beyond midpoint in every population (Table 8). Appendix G reports the dense-reference and structured-validation checks on these same groups.

Sampling scope. The unsuccessful properties are 14, 29, 53, and 85 on the smaller official model. Each contributes ten groups with $K = 5$ from a search that remains unknown under the 100-second budget. Two early groups come from one attention layer and improve after baseline retention; eight come from the other layer and retain the baseline. Later evaluations of the improving layer are not sampled.

For each property, we compare how much midpoint and full improve on the baseline. In the per-property medians, midpoint retains 81–91% of full's gain. Each median, however, uses only the two early groups that improve. This statistic is not representative of a balanced sample of layers or search stages. These cases do not cover the deeper models without normalization.

H.2 How much local residual uncertainty remains?

Exact directional ranges contribute to the convexifier itself. On all twenty-two prespecified groups, replacing only these ranges with bounds derived from individual probability intervals or from the smallest and largest coefficients weakens the candidate. The decomposed and Taylor candidates are held fixed. This demonstrates a local contribution without establishing additional certifications.

We next bound how much further the intercept could improve for the same fixed local problem.

Let L be an independently certified lower bound on a group's fixed-slope residual minimum b^* and U an upper bound obtained by evaluating the residual at a feasible point with outward rounding. Then $L \le b^* \le U$. On the twenty-two prespecified groups, $U - L$ has median 0.00253400 and range $[0.00007831, 0.32169527]$.

To obtain the upper bound U, we evaluate the residual at feasible points found by numerical search. These feasible values bound the minimum from above; they do not prove a lower bound over the box. For the sampled local problems from unsuccessful verification runs, we search from three fixed starting points using L-BFGS-B, allowing at most one hundred iterations. We then evaluate the resulting feasible points independently with outward rounding.

Table 9: Median ceilings on further local improvement for unsuccessful properties, relative to the stronger of the baseline and candidate bounds. A feasible residual value supplies the upper bound; the ceiling need not be attainable.

Property	14	29	53	85
Median upper ceiling	0.127239	0.077187	0.0758219	0.153219

A loose feasible upper bound can leave a large ceiling (Table 9), so these brackets do not identify whether anchors, convexification, score correlations, or upstream propagation limit certification. Predicting how the final output bound would change requires holding the relaxation state of the complete network fixed.

H.3 Do improved search bounds cross the certification threshold?

Eleven properties in the broader 32-property panel remain uncertified by every component variant at both solver seeds. Full Vertex returns a larger bound than Shadow and midpoint in all twenty-two property/seed comparisons, but every bound remains negative. The median full-minus-midpoint difference is 0.000941992; the median change relative to the magnitude of the negative Shadow bound is 1.98%. The median difference between full and Shadow in the number of branch-and-bound subproblems processed is zero.

These are finite bounds returned by branch-and-bound, without substituting root bounds. Under the verifier's convention they are relative to its certification threshold, not maximal certification margins. Because separate searches can follow different trajectories and stop at different states, these differences do not identify the cause of unresolved certification or establish a throughput advantage.