THRESHOLD-MASS CROWN: BOUNDING ATTENTION FOR TRANSFORMER VERIFICATION

Anonymous authors

ABSTRACT

Attention is difficult to bound because its weights change together: increasing one score changes the probability assigned to every token. A robustness verifier must account for these interactions and their effect on the network's output. We introduce Threshold-Mass CROWN, which uses the coefficients supplied by backward verification to bound a weighted sum of attention probabilities as a whole. The method sorts tokens by these coefficients and bounds the total attention assigned to tokens above each cutoff in the ordering. These bounds combine into one affine function of the attention scores. Propagating this function toward the input can preserve relationships between scores that independent score ranges discard. We prove that this affine function is a lower bound throughout the allowed score ranges. For an attention row with K tokens, sorting and cumulative sums take $O(K \log K)$ time and $O(K)$ working storage per coefficient vector. Across five independently trained one-block models, Threshold-Mass certifies 103 of 500 examples, compared with 49 for the CROWN-LSE baseline, at additional runtime cost. On a 200-property transformer benchmark, it certifies 143 properties. A control that performs the same additional calculation but returns the baseline bound certifies 127. The results show that retaining score relationships can improve certification, although stronger attention bounds alone remain insufficient for some more demanding architectures.

1 INTRODUCTION

Small changes to a transformer's input can affect its output. Robustness verification asks whether a requirement on that output holds for every allowed change to the input. For a classifier, the requirement is that the predicted class remains unchanged. We can establish this by showing that its output score stays above every competing class score throughout the allowed input region. Testing a few perturbed inputs cannot establish this guarantee; the verifier needs bounds that cover the entire region.

Attention makes this calculation difficult because its weights are coupled through softmax. Changing one attention score changes the probabilities across the row. The verifier must account for these changes and their effect on the final class scores. It must also account for relationships between attention scores that arise because they depend on the same input. Losing these relationships can make the bounds too loose to show that the predicted class stays ahead of its competitors.

Backward verification starts from the difference between the predicted class score and a competing class score. It works backward through the network to find a lower bound on that difference. At an attention operation, the calculation involves a weighted sum of attention probabilities. The coefficients specify how those probabilities enter the bound on the class-score difference. Existing approaches first bound individual attention probabilities and then combine the bounds using these coefficients (Wei et al., 2023; Zhang et al., 2024). We use the coefficients to construct the bound itself. They determine how we group tokens, allowing us to bound their combined contribution to the weighted sum.

We introduce **Threshold-Mass CROWN**. The method sorts tokens by their coefficients and forms groups containing all tokens above successive coefficient thresholds. We call the total attention assigned to each group a threshold mass. The original weighted sum can be rewritten exactly using these group probabilities and a constant term. Each group probability has a nonnegative multiplier, so

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.

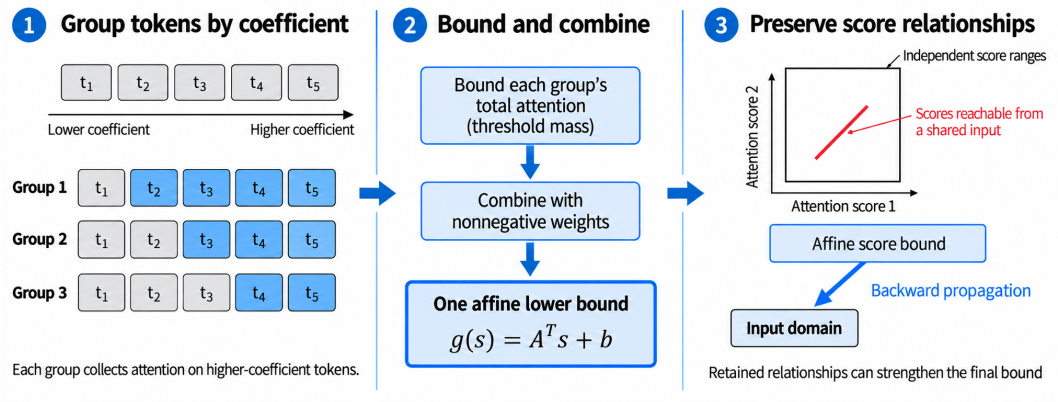

Figure 1: **Overview of Threshold-Mass CROWN.** Sorting tokens by their objective coefficients defines nested groups. The total attention assigned to each group is a threshold mass. Lower bounds on these masses combine with nonnegative weights into one affine score bound. Propagating this bound toward the input can retain score relationships and strengthen the final verification bound. The red line illustrates score pairs produced by a shared input within the independent score ranges; reachable sets need not be straight lines.

its lower bound can be safely combined with those of the other groups. Section 3 develops this idea through a three-token example before presenting the general construction.

The resulting bound is a weighted sum of attention scores plus a constant, an affine function. Keeping the scores in this expression lets the verifier continue backward through the computations that produced them. This matters because the scores share the same input: some combinations of score values may be impossible, even when each value lies within its individual range. Backward propagation can use this shared dependence to obtain a stronger bound.

A constant lower bound no longer depends on the attention scores, so it cannot carry these relationships into the earlier computations. This remains true even if the constant is the exact minimum over the independent score ranges. Our affine bound preserves that opportunity, but introduces approximation error. The final bound improves when the retained relationships compensate for this additional error. Figure 1 illustrates the idea.

We prove that the construction gives a valid lower bound throughout the allowed score ranges. We also show how to compute it efficiently. Computing each group's contribution separately repeats work because the groups overlap, requiring $O(K^2)$ operations for K tokens. Sorting and cumulative sums reuse the shared calculations to obtain the same bound in $O(K \log K)$ time and $O(K)$ working storage. These costs apply to one attention row and one coefficient vector. Section 3.6 gives the algorithm and its computational assumptions.

In a complete verifier, we evaluate Threshold-Mass alongside a baseline called CROWN-LSE. This baseline combines CROWN backward bound propagation with a log-sum-exp relaxation of softmax (Zhang et al., 2018; Wei et al., 2023). We compare the bounds after propagation to the input, where retained score relationships can affect the final bound on the difference between class scores. Because the additional calculation takes time, we measure both bound quality and the number of properties certified within a fixed budget.

We evaluate the method through complete verification experiments on transformer classifiers and local attention-bound comparisons in a language model. In five independently trained one-block classifiers, Threshold-Mass certifies 103 of 500 examples, compared with 49 for CROWN-LSE, at additional runtime cost. On the 200-property vision-transformer benchmark, it certifies 143 properties, compared with 127 for a control that computes the same candidate bound but uses the baseline bound. The language-model experiments examine local bound quality, while tests on more demanding transformer architectures show how loose bounds and memory limits can prevent certification.

2 RELATED WORK

Three lines of work lead to the construction studied here. Backward bound propagation provides the verification framework, softmax relaxations provide ways to bound coupled attention weights, and coefficient ordering reveals structure in a weighted softmax objective. Together they motivate a bound that uses the current objective to group tokens and retains an affine dependence on their scores.

Bound propagation and verification. CROWN constructs affine bounds and propagates them backward through neural networks (Zhang et al., 2018). Automatic bound-propagation frameworks extend this approach to more general computational graphs (Xu et al., 2020). Optimizing the relaxations and incorporating constraints from branch-and-bound splits further strengthen the verification procedure (Xu et al., 2021; Wang et al., 2021; Shi et al., 2025). Threshold-Mass supplies a softmax bound within this framework. The surrounding verifier continues to handle the other operations and the search over input subdomains. **Attention and softmax bounds.** Transformer verification has motivated specialized methods for bounding attention (Shi et al., 2020; Bonaert et al., 2021). Convex and concave softmax bounds account for interactions among scores (Wei et al., 2023), while GaLileo develops multidimensional affine relaxations (Zhang et al., 2024). These approaches provide bounds for individual softmax outputs that can be combined for the weighted objective supplied by backward verification. Threshold-Mass uses that objective's complete coefficient vector to choose the groups from which its affine bound is built. **Coefficient ordering and exact scalar bounds.** The unpublished Vertex-Softmax preprint uses coefficient ordering to compute exact weighted-softmax extrema on score boxes (Rezazadeh & Davoodi, 2026). It also examines the information lost when those boxes discard relationships between scores. The same ordering motivates our token groups. Bounding their probabilities produces an affine function that can continue through backward propagation. Appendix A.6 restates the exact scalar calculation to make the comparison self-contained.

3 THRESHOLD-MASS: BUILDING A BOUND FROM ATTENTION GROUPS

At an attention operation, backward verification must bound a weighted sum of attention probabilities. The verifier supplies the coefficients of this sum and a range for each attention score. Our task is to construct an affine lower bound that holds for every score vector within those ranges. Keeping the bound dependent on the scores allows the verifier to continue backward through the computations that produced them.

We construct this bound by grouping tokens according to their coefficients. A three-token example explains how to rewrite the weighted sum using group probabilities, whose lower bounds combine into a valid bound for the full expression. A second example shows why retaining score dependence can help after propagation. We then use cumulative sums to compute the same bound efficiently. Appendix C describes the numerical implementation, and Section 6 discusses its limitations.

3.1 THE LOCAL VERIFICATION PROBLEM

Problem statement. Consider one attention row containing $K \geq 1$ unmasked scores, with score vector $s \in \mathbb{R}^K$ and probability vector $p(s) = \mathrm{softmax}(s)$. The verifier supplies a coefficient vector $c \in \mathbb{R}^K$ and finite score bounds $\ell \leq u$. Write $\mathcal{B} = [\ell, u]$ for the score box. The expression to be bounded is

$$F_c(s) = c^\top \mathrm{softmax}(s) = \sum_{j=1}^{K} c_j p_j(s). \tag{1}$$

Our task is to construct a vector A_c and a constant B_c such that

$$g_c(s) = A_c^\top s + B_c \leq F_c(s) \qquad \text{for all } s \in \mathcal{B}. \tag{2}$$

The coefficients c are fixed for this local calculation, while the scores s vary within the supplied ranges. Once the affine bound is constructed, the verifier substitutes g_c for F_c and continues backward through the score computation. The vector A_c describes how the lower bound changes with the scores.

For example, suppose the attention row combines fixed value vectors V_j. Let w specify how the verification objective weights the output coordinates. The resulting expression is $\sum_j (w^\top V_j) p_j(s)$,

so $c_j = w^\top V_j$. In a complete network, the value vectors can also vary. The surrounding verifier handles that dependence before supplying the coefficients for the softmax bound.

3.2 Rewriting the objective as group probabilities

Sorting the coefficients lets us rewrite the objective exactly in terms of group probabilities with nonnegative multipliers. A three-token example shows how this rearrangement makes separate group bounds useful.

Consider three coefficients with $c_1 \le c_2 \le c_3$. Since the probabilities sum to one, their weighted sum can be written as

$$c_1p_1 + c_2p_2 + c_3p_3 = c_1 + (c_2 - c_1)(p_2 + p_3) + (c_3 - c_2)p_3.$$

The constant c_1 remains regardless of where attention is assigned. The second term accounts for the extra contribution from attending to token 2 or 3, and the third accounts for the further increase at token 3. Both coefficient differences are nonnegative.

Write $q_1 = p_2 + p_3$ and $q_2 = p_3$ for the two group probabilities. The construction needs a lower bound for each. Their nonnegative multipliers allow us to combine these bounds even when some original coefficients are negative.

The same rearrangement works for any row length. Let $c_{(1)} \le \cdots \le c_{(K)}$ denote the coefficients in sorted order, with the scores reordered to match. For $m = 1, \ldots, K-1$, let L_m contain the first m tokens and H_m the remaining tokens. Define the neighboring coefficient gap as $\Delta c_m = c_{(m+1)} - c_{(m)}$.

Each group competes for attention with the tokens outside it. The total probability assigned to the group is the sigmoid of the difference between the two groups' log-sum-exp values.

Lemma 3.1 (Decomposition into threshold masses)**.** *For $K \ge 2$, the weighted softmax expression satisfies*

$$F_c(s) = c_{(1)} + \sum_{m=1}^{K-1} \Delta c_m q_m(s), \qquad q_m(s) = \sum_{j \in H_m} p_j(s). \tag{3}$$

Every coefficient Δc_m is nonnegative. Each group probability also satisfies

$$q_m(s) = \sigma(\mathrm{LSE}(s_{H_m}) - \mathrm{LSE}(s_{L_m}))\,. \tag{4}$$

Here s_G contains the scores indexed by group G, $\mathrm{LSE}(z) = \log \sum_j e^{z_j}$, and $\sigma(z) = 1/(1 + e^{-z})$.

We can therefore bound each group probability in two steps: first bound the difference between the groups, then bound the sigmoid applied to that difference. Appendix A.1 proves both identities. Equal neighboring coefficients create zero gaps, so the corresponding terms can be skipped. If all coefficients are equal, the objective is already constant.

3.3 Bounding one group probability

We now need a lower bound for the attention assigned to a high-coefficient group. Equation (4) gives a way to build one: compare the two groups on a logarithmic scale, then pass that comparison through a sigmoid. The aim is to replace both steps by an affine expression in the original scores.

First, bound the difference between the groups. Write $d_m(s) = \mathrm{LSE}(s_{H_m}) - \mathrm{LSE}(s_{L_m})$ for the difference in (4). Increasing this difference increases the attention assigned to H_m. We therefore need an expression that stays below d_m throughout the score box.

For the high group, we need a lower bound on its log-sum-exp. This function is convex, so its tangent at the score-box midpoint gives an affine lower bound.

The low group's log-sum-exp is subtracted, so we need an upper bound for it. We first bound each exponential by the line joining its endpoint values. Adding these lines gives an upper bound on the sum of exponentials. Taking the logarithm preserves the inequality. Because the logarithm is concave,

its tangent gives an affine upper bound on this logarithm. Subtracting that upper bound from the high group's lower bound gives

$$\underline{d}_m(s) = v_m^\top s + \rho_m \le d_m(s). \tag{5}$$

Here v_m contains the score coefficients and ρ_m is the constant term. If a group contains one token, its log-sum-exp equals that token's score and needs no approximation. Appendix A.2 gives the group formulas and proves the inequalities.

Next, turn the difference into a probability bound. The sigmoid converts d_m into the group probability q_m. Choose a line $\lambda_m z + \kappa_m$ that lies below the sigmoid throughout the range of d_m, with $\lambda_m \ge 0$. We can then substitute the affine lower bound $\underline{d}_m$ from Equation (5):

$$q_m(s) \ge \lambda_m d_m(s) + \kappa_m \ge \lambda_m \underline{d}_m(s) + \kappa_m. \tag{6}$$

The first inequality uses the sigmoid line; the second uses its nonnegative slope. We denote the final affine expression by $\underline{q}_m(s)$. The line must be valid over the range of the true difference d_m. The lower approximation $\underline{d}_m$ may extend below that range, but the two inequalities above still apply.

Several lines satisfy the required inequality, so we need a practical selection rule. Our default considers six slopes in $[0, 1/4]$. The upper endpoint is the maximum sigmoid derivative, while slope zero gives a constant lower bound. For each slope, we compute the largest intercept that keeps the line below the sigmoid throughout the required interval.

We evaluate $\underline{d}_m$ at the score-box midpoint to obtain a reference value. Among the candidate sigmoid lines, we choose the one with the largest value there. This reference guides the choice; every candidate remains valid over the full required interval. Appendices A.2, A.3, and B.1 give the reference value, intercept calculation, and selection rule.

Returning to the three-token example. For $q_2 = p_3$, the high group contains only token 3. We therefore bound the difference $s_3 - \mathrm{LSE}(s_1, s_2)$ by subtracting the low group's affine upper bound from s_3. We then apply the selected sigmoid line, whose slope is nonnegative. The resulting affine lower bound $\underline{q}_2(s)$ retains dependence on the original scores.

For $q_1 = p_2 + p_3$, the corresponding difference is $\mathrm{LSE}(s_2, s_3) - s_1$; applying the same construction gives $\underline{q}_1(s)$. Combining the two group bounds gives

$$g_c(s) = c_1 + (c_2 - c_1)\underline{q}_1(s) + (c_3 - c_2)\underline{q}_2(s).$$

The nonnegative coefficient differences ensure that $g_c(s) \le F_c(s)$ throughout the score box. The result remains a function of the scores, allowing their shared input dependence to be used during propagation.

3.4 Combining the bounds

For any number of tokens, the group bounds combine in the same way. Collecting their score coefficients and constant terms gives one affine function, whose validity we now state for the full objective.

Substituting (6) into the threshold decomposition produces

$$g_c(s) = c_{(1)} + \sum_{m=1}^{K-1} \Delta c_m \left[\lambda_m (v_m^\top s + \rho_m) + \kappa_m\right].$$

Collecting the score coefficients and constants gives A_c and B_c in (2).

Theorem 3.2 (Validity of the Threshold-Mass bound)**.** *Let $c \in \mathbb{R}^K$ be fixed and let $\ell, u \in \mathbb{R}^K$ satisfy $\ell \le u$. Use the group bounds described above and choose each sigmoid slope in $[0, 1/4]$. For each slope, compute the maximal valid intercept over the exact range of d_m, as described in Appendix A.3. Then $g_c(s) \le F_c(s)$ throughout $[\ell, u]$. For $K = 1$, the exact bound is the constant c_1. Applying the lower-bound construction to $-c$ and negating the result gives an upper bound.*

The guarantee allows the verifier to substitute g_c for the weighted softmax expression throughout the score box. It holds for any permitted slope choice; finding the best slope is unnecessary for validity. Appendix A.4 proves it by combining the group inequalities. We next examine when the retained score dependence can improve the propagated bound.

3.5 WHY KEEPING SCORE DEPENDENCE CAN HELP

An affine bound can give a stronger result after propagation than the exact constant obtained from independent score ranges. The reason is that propagation can recover relationships between scores that those ranges omit. A two-score example makes this mechanism explicit.

Suppose two scores have the form $s_1(x) = x$ and $s_2(x) = x - a$. Their difference is always $-a$, so the second softmax probability stays constant as x changes. Separate score intervals nevertheless allow combinations in which one score takes its largest value and the other its smallest. Such combinations may not come from any common input.

Minimizing over the independent score intervals includes these extra combinations. An affine function involving $s_2 - s_1$ can preserve the relationship when the score expressions are substituted. If the entire permitted range of $s_2 - s_1$ is negative, the sigmoid is convex there, so its tangent gives a valid lower bound. Proposition 3.3 makes this advantage precise.

Proposition 3.3 (A strict gain from retaining score dependence)**.** *Let $a > 2r > 0$, $\mathcal{X} = [-r, r]$, $s(x) = (x, x - a)$, and $c = (0, 1)$. The coordinate ranges give $\mathcal{B} = [-r, r] \times [-r - a, r - a]$. In real arithmetic, the six-candidate Threshold-Mass construction returns an affine function g_c with $g_c(s) \le F_c(s)$ throughout $\mathcal{B}$ and*

$$\min_{x \in \mathcal{X}} g_c(s(x)) = \sigma(-a) > \sigma(-a - 2r) = \min_{s \in \mathcal{B}} F_c(s). \tag{7}$$

The returned function retains the difference $s_2 - s_1$. Substituting $s_1 = x$ and $s_2 = x - a$ cancels x, recovering the constant attention probability. The box minimum instead includes score pairs that no input can produce. Appendix A.7 gives the full parameterized statement and proof, including why the six-candidate rule selects the required tangent.

The gain comes from using the score relationship during propagation. If the affine bound were minimized immediately over the same independent box, it could not exceed the exact scalar minimum. In a larger network, approximation errors in the group bounds and subsequent propagation toward the input can reduce or eliminate this gain. The relevant comparison is therefore between completed bounds, obtained after propagation to the input and minimization over the input domain.

3.6 COMPUTING THE BOUND WITH CUMULATIVE SUMS

Our direct reference construction explicitly forms the score coefficients for every threshold, using $O(K^2)$ work and storage. Cumulative sums recover the same affine bound with $O(K \log K)$ work and $O(K)$ working storage. These costs apply to one attention row and one backward coefficient vector. The saving comes from reusing calculations across overlapping groups.

Each low group is a prefix, the first tokens in sorted order, and each high group is the remaining suffix. Forward and reverse cumulative sums reuse the exponential and midpoint terms shared by these groups. Those sums provide the group bounds, the ranges of d_m, and the reference values. Two further cumulative sums assemble the score coefficients. We call this computation the scan.

The scan changes how contributions are accumulated, while keeping the group bounds and sigmoid-line choices unchanged. It uses the same reference values, exponential endpoint bounds, and exact formulas for one-token groups. Candidate ordering and tie-breaking are also unchanged.

Theorem 3.4 (Equivalent construction with linear storage)**.** *Use the construction choices above. Let the number of candidate sigmoid lines be independent of K, and resolve ties deterministically. The prefix–suffix construction in Appendix B.2 returns the same A_c and B_c as the direct threshold sum in real arithmetic. For one row and one backward coefficient vector, it requires $O(K \log K)$ operations and $O(K)$ auxiliary storage, counting elementary scalar functions as unit-cost operations.*

Sorting accounts for the logarithmic factor. Different accumulation orders can produce floating-point differences, which Appendix C examines numerically. Appendix B.2 gives the scan formulas and proof, and Algorithm 1 summarizes the construction.

Algorithm 1 Threshold-Mass for one attention row

Input: Fixed coefficients c and finite score bounds $\ell \leq u$.

1. Sort c, applying the same permutation to ℓ and u, and break ties by original index. If all coefficients are equal, return their common value with zero slope.
2. Use forward and reverse cumulative sums to obtain the shared quantities for each group bound, the range of d_m, and its reference value.
3. For each nonzero coefficient gap, construct the six candidate sigmoid lines using their valid intercepts. Select the first line attaining the largest value at the reference value.
4. Weight each group bound by its coefficient gap. Use the scan formulas to accumulate the score coefficients into A_c and the constant terms, including $c_{(1)}$, into B_c.
5. Restore the original coordinate order and return (A_c, B_c).

Output: An affine lower bound $A_c^\top s + B_c$ satisfying (2) in real arithmetic.

4 Using Threshold-Mass in a verifier

This section explains how a verifier computes and uses an additional Threshold-Mass bound. Substitution and selection preserve validity. Search experiments measure the additional calculation's benefit and cost, while a shadow control tests the effect of using its result.

At each softmax node, the verifier supplies coefficients and score ranges, substitutes the affine bound, and continues propagation. Validity requires sound score bounds and propagation rules. Contributions from different rows combine before minimization, so their score relationships can remain available. Section 3.5 illustrates how such a relationship can yield exact cancellation.

For classification, a margin is the correct-class score minus a competing-class score. Minimizing its propagated affine bound over the input domain gives a completed margin bound. The smallest bound across competing classes is the minimum margin bound. A positive value certifies robustness.

Comparing completed bounds. The verifier first optimizes CROWN-LSE's relaxation parameters, which control its affine approximations. These parameters remain fixed during the additional Threshold-Mass pass.

Each class comparison is a constraint requiring a positive margin throughout the current domain. Let $L_{\text{CROWN-LSE}}$ and L_{TM} be the two completed bounds for the same margin. When only lower bounds are requested, the selected bound is

$$L_{\text{sel}} = \max\{L_{\text{CROWN-LSE}}, L_{\text{TM}}\}. \tag{8}$$

Both bounds concern the same margin and input domain, so their maximum remains valid.

When affine coefficients or upper bounds are also requested, the verifier returns all outputs from one method. In this case, it uses Threshold-Mass only if every lower bound matches or exceeds the baseline. Otherwise, it returns the baseline. Missing intermediate bounds, unsupported tensor dimensions, and NaN or infinite values also trigger this fallback. Appendix B.3 gives the guarantees.

Search and the shadow control. Branch-and-bound splits an unresolved input domain and repeats the bound calculation on smaller subdomains. Stronger bounds can establish the property sooner, but their extra cost can offset that benefit under a time budget.

A shadow control computes Threshold-Mass but returns the baseline bound. Matching computation, intermediate state, properties, and time budget helps isolate the benefit of using the bound. Comparison with the original CROWN-LSE verifier, which performs no extra pass, measures whether the extra work pays off. Both comparisons require complete runs because search paths can differ.

5 Experimental results

The one-block and benchmark studies show stronger bounds and more certificates. Controls test the benefit of using the bound and retaining score dependence. Local language-model tests assess bound quality, and timings assess construction cost. More demanding architectures yield few certificates and memory failures.

One-block verification. Across five training seeds, Threshold-Mass certifies 103 of 500 examples, compared with 49 for CROWN-LSE. The paired runtime ratio is $1.64\times$, with CROWN-LSE as the reference.

Five independently trained Fashion-MNIST models each contain one attention–residual–ReLU–MLP block without LayerNorm. Methods use the same weights and first 100 evaluation examples per model, at ℓ_∞ radius 0.02. Certification requires a correct prediction on the unperturbed input and verified robustness. Incorrect predictions and verification failures remain among the 500 attempts.

A reference-optimal variant searches all permitted slopes and chooses the sigmoid lower line with the highest value at the same reference point.

Table 1: Matched one-block results over 500 attempts. Each example's lower bound is the minimum across completed class-margin bounds. Mean lower bounds use the 358 examples classified correctly before perturbation (clean-correct). Runtime ratios are paired within training seed.

Method	Certified / 500	Mean lower bound	Runtime / CROWN-LSE
CROWN-LSE	49	-4.5446	$1.00\times$
Threshold-Mass	103	-0.3574	$1.64\times$
Reference-optimal slope variant	103	-0.3595	$1.74\times$

Threshold-Mass improves all 358 clean-correct bounds over CROWN-LSE. Across seeds, certificate-rate gains range from 2 to 22 percentage points, averaging 10.8 points (Appendix D.1).

Official benchmark. Threshold-Mass certifies 143 properties versus 127 for its shadow on the 200-property VNN-COMP 2023 vision-transformer suite (Brix et al., 2023). All 127 shadow certificates are retained, giving an eight-percentage-point gain. Appendix D.2 gives the full outcomes, paired comparisons, and timing.

Table 2: Certification on the 200-property benchmark with a 100-second budget per property. The shadow computes the candidate Threshold-Mass bound but returns the baseline bound.

Method	Certified / 200	Rate
Threshold-Mass shadow	127	63.5%
Threshold-Mass CROWN	143	71.5%

A separate comparison examines completed bounds on the original input domain, before branch-and-bound splits it. Of 200 properties, 39 are certified before this calculation, leaving 161. Threshold-Mass improves the minimum margin bound on all 161 properties relative to optimized CROWN-LSE, with a mean improvement of 0.1347. Appendix D.3 also reports separate selected-property controls that distinguish the benefit of using the bound from the effect of merely computing it.

Construction choices. Two comparisons examine the construction choices: slope selection in the one-block study and affine versus constant bounds on the benchmark. In the one-block study, the reference-optimal variant certifies the same 103 examples as the default, with a slightly weaker mean propagated bound and higher runtime (Table 1). We therefore retain the six-candidate default.

Threshold-Mass retains all exact-scalar certificates and adds 33 (Table 3). This control replaces the affine bound with the tightest constant over the score box to test the benefit of retained score dependence. Appendix D.5 gives the protocol and separate root-bound comparisons.

Language-model attention bounds. Threshold-Mass improves most local bounds on attention rows recorded from Qwen2.5-3B-Instruct (Qwen Team, 2025). We compare local bounds with CROWN-LSE using fixed coefficient vectors, without propagation through the complete model.

The sampled rows contain 64 or 128 tokens. Evaluating 128 rows at four perturbation radii gives 512 cases, of which 448 have nonzero coefficient vectors. On these 448 cases, Threshold-Mass improves 409 bounds, ties 38, and worsens one relative to CROWN-LSE, using tolerance 10^{-6}.

Table 3: Exact-scalar comparison on 200 properties, with 110 shared certificates. Method-only counts give additional properties certified by just one method.

Method	Certified / 200	Method-only certificates
Exact scalar	110	0
Threshold-Mass	143	33

Setting every sigmoid slope to zero gives constant group bounds, which improve fewer cases than the default. This variant bounds each group separately. The exact-scalar control instead minimizes the original weighted-softmax expression jointly. Appendix D.7 compares the variants.

Construction cost and numerical checks. Cumulative sums speed up construction on longer rows; 360 high-precision checks find no bound violations.

At 1,024 scores per row, a batch of 32 rows takes 3.72 milliseconds with the scan and 2,606.76 milliseconds with the direct implementation. The scan is slightly slower at two scores per row. Tests reach 2,048 scores, with direct comparisons up to 1,024. These timings measure local construction rather than complete verification (Appendix D.6).

Checks cover narrow and wide score intervals, equal coefficients, and a common constant added to all scores. Appendix C reports the numerical agreement and remaining floating-point limitations.

More demanding architectures. Adding transformer blocks or LayerNorm yields few certificates despite stronger bounds, while longer attention rows encounter memory failures.

We compare with independent implementations of the ER, LSE, and LSE* softmax relaxations of Wei et al. (2023). For each example, the reference is the largest of their three completed minimum-margin bounds. The comparison includes clean-correct examples for which Threshold-Mass and all three references return finite bounds. Across five seeds, Threshold-Mass improves the reference on every included example. These gains yield only two certificates among 500 two-block examples and none with pre-LayerNorm.

With 49 tokens, Threshold-Mass exhausts memory on each of the 335 clean-correct examples (Appendix D.8).

6 Limitations

Complete verification remains limited by bound quality, memory, and numerical guarantees. Score boxes can admit combinations that no input produces, and approximations throughout the network can weaken the final bound.

The theorems assume real arithmetic and valid bounds from the surrounding verifier. Numerical safeguards do not cover every rounding error in the construction or complete verifier (Appendix C).

7 Conclusion

Threshold-Mass uses the verification objective to group tokens and combines bounds on their attention probabilities into an affine function of the scores. Cumulative sums compute this bound with near-linear work and linear working storage. The one-block and benchmark experiments show that retaining score dependence can improve certification. On more demanding architectures, stronger attention bounds alone remain insufficient.

REPRODUCIBILITY STATEMENT

The implementation file is provided as supplementary material. The package contains a standalone PyTorch implementation of the Threshold-Mass affine bound, including the cumulative-sum scan, installation instructions, a usage example, and tests on synthetic inputs. It allows the local construction to be evaluated independently of a complete verifier; the tests check agreement with the direct construction and the scan's storage requirements. Appendices A–C give the formulas, assumptions, proofs, and numerical safeguards, and Appendix D describes the evaluation sets, hardware, metrics, and experimental protocols. Reproducing the full verification experiments additionally requires the host verifier, datasets, and trained checkpoints, which are not included in the standalone package.

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

Gregory Bonaert, Dimitar I. Dimitrov, Maximilian Baader, and Martin 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. URL https://files.sri.inf.ethz.ch/website/papers/pldi21-transformers.pdf.

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. URL https://arxiv.org/abs/2312.16760.

Rudy Bunel, Jingyue Lu, Ilker Turkaslan, Philip H. S. Torr, Pushmeet Kohli, and M. Pawan Kumar. Branch and bound for piecewise linear neural network verification. *Journal of Machine Learning Research*, 21(42):1–39, 2020. URL https://jmlr.org/papers/v21/19-468.html.

Robert Joseph George, Jennifer Cruden, Xiangru Zhong, Huan Zhang, and Anima Anandkumar. TorchLean: Formalizing neural networks in Lean. arXiv preprint arXiv:2602.22631v1, 2026. URL https://arxiv.org/abs/2602.22631v1.

Qwen Team. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115v2, 2025. URL https://arxiv.org/abs/2412.15115v2. Revised January 3, 2025; first posted December 2024.

Navid Rezazadeh and Arash Gholami Davoodi. Vertex-Softmax: Tight transformer verification via exact softmax optimization. arXiv preprint arXiv:2605.10974v1, 2026. URL https://arxiv.org/abs/2605.10974v1. Unpublished preprint; not a peer-reviewed publication.

Zhouxing Shi, Huan Zhang, Kai-Wei Chang, Minlie Huang, and Cho-Jui Hsieh. Robustness verification for transformers. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJxwPJHFwS.

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. doi: 10.1007/978-3-031-90643-5_17. URL https://arxiv.org/abs/2405.21063.

Attila Szász, Balázs Bánhelyi, and Márk Jelasity. No soundness in the real world: On the challenges of the verification of deployed neural networks. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 58088–58105. PMLR, 2025. URL https://proceedings.mlr.press/v267/szasz25a.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, pp. 29909–29921, 2021. URL https://proceedings.neurips.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. URL https://proceedings.mlr.press/v206/wei23c.html.

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*, volume 33, pp. 1129–1141, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/hash/0cbc5671ae26f67871cb914d81ef8fc1-Abstract.html.

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://openreview.net/forum?id=nVZtXBI6LNn.

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*, volume 31, 2018. URL https://papers.neurips.cc/paper_files/paper/2018/hash/d04863f100d59b3eb688a11f95b0ae60-Abstract.html.

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. doi: 10.1609/aaai.v38i19.30180. URL https://ojs.aaai.org/index.php/AAAI/article/view/30180.

Xingjian Zhou, Keyi Shen, Andy Xu, Hongji Xu, Cho-Jui Hsieh, Huan Zhang, and Zhouxing Shi. SoundnessBench: A soundness benchmark for neural network verifiers. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=UuYYldVLH3.

A MATHEMATICAL FOUNDATIONS AND PROOFS

The technical appendices answer three questions: why is the bound valid, when can retaining score dependence help, and how can the bound be computed and used efficiently? Here we establish validity in the same order as the construction: rewrite the objective, bound one group probability, and combine the group bounds. We then compare the affine bound with the strongest constant bound and prove the separation result from the main text. The final two subsections examine slope selection, constant group bounds, and behavior under common score shifts. Appendix B derives the cumulative-sum computation and its use in the verifier; Appendix C addresses rounding and numerical checks.

We work in real arithmetic with finite score intervals, including intervals of zero width, called degenerate intervals. The incoming coefficient vector c is fixed during each local substitution. A sound bound is one that is valid throughout the stated domain. An affine minorant is an affine lower bound; we also call its graph a plane and its constant term the bias. Each nontrivial threshold divides the tokens into two nonempty, disjoint groups. We introduce the needed quantities as the argument develops; Table 4 provides a reference.

Let $\mathcal{X}$ be a nonempty input domain, $s : \mathcal{X} \to \mathbb{R}^K$ its score map, and $\mathcal{S} = s(\mathcal{X}) \subseteq \mathcal{B} = [\ell, u]$ the set of score vectors that the allowed inputs can produce. We use

$$p_j(s) = \frac{e^{s_j}}{\sum_{i=1}^{K} e^{s_i}}, \qquad \mathrm{LSE}(z) = \log \sum_i e^{z_i}, \qquad \sigma(z) = \frac{1}{1+e^{-z}}. \tag{9}$$

The local objective is $F_c(s) = c^\top p(s)$, and $V_{\mathcal{B}}(c)$ denotes its exact minimum over the score box. These definitions are shared by the validity and comparison arguments below.

Table 4: Notation and parameter domains. Scanned quantities are defined in Appendix B.2.

Symbol	Definition and domain
$K \geq 1$	Number of unmasked scores in one softmax row.
$\mathcal{X}, s, \mathcal{S}$	Nonempty input domain, score map $s : \mathcal{X} \to \mathbb{R}^K$, and reachable set $\mathcal{S} = s(\mathcal{X})$.
$\ell, u, \mathcal{B}$	Finite vectors with $\ell \leq u$ and sound box $\mathcal{B} = [\ell, u] \supseteq \mathcal{S}$.
$c \in \mathbb{R}^K$	Fixed incoming backward coefficient vector.
$p(s)$, $F_c(s)$	Softmax probabilities and $c^\top p(s)$.
σ, LSE	Logistic sigmoid and log-sum-exp; see (9).
$V_{\mathcal{B}}(c)$	Exact scalar minimum of F_c on the independent score box.
$g, G_{\mathcal{S}}, P_{\mathcal{X}}(g)$	Sound affine minorant; its infimum on $\mathcal{S}$; any finite sound lower bound on that infimum obtained by propagation toward the input.
π, $c_{(j)}$	Ascending sorting permutation and coefficient $c_{\pi(j)}$.
$L_m, H_m, \Delta c_m$	Low prefix, high suffix, and nonnegative coefficient gap, $1 \leq m < K$.
q_m, d_m	High-group probability mass and true high-minus-low LSE difference.
t, ξ	Midpoint $(\ell + u)/2$ and shared exponential shift $\max_j u_j$.
h_G, r_G	Affine lower and upper LSE functions on group G.
$\underline{d}_m = v_m^\top s + \rho_m$	Affine lower bound on d_m, not necessarily confined to its exact interval.
D_m^-, D_m^+	Exact minimum and maximum of d_m on $\mathcal{B}$.
$r_m^\star$	Scalar selection reference $\underline{d}_m(t) \in [D_m^-, D_m^+]$.
λ_m, κ_m	Nonnegative sigmoid slope in $[0, 1/4]$ and maximal valid fixed-slope intercept.
Λ	Declared ordered constant-size map from (D^-, D^+, r) to candidate slopes.
A_c, B_c	Score coefficient vector and bias of the joint affine return.
A, ε, R	Separation-family parameters: $A > \varepsilon > 0$, $R = (A - \varepsilon)/2$.

A.1 PROOF OF THE ORDERED-MASS IDENTITY

The first step is to replace a weighted sum of individual probabilities by a sum of group probabilities. This matters because the gaps between sorted coefficients are nonnegative: once each group has a lower bound, those bounds can be added safely. The identity is exact, including for tied or negative coefficients. We prove it by expanding each coefficient into its smallest value plus successive gaps, then expressing each group mass through a sigmoid.

Let π sort the coefficients in ascending order. At threshold m, the first m tokens form the low group and the remaining tokens form the high group:

$$L_m = \{\pi(1), \ldots, \pi(m)\}, \quad H_m = \{\pi(m+1), \ldots, \pi(K)\}, \quad \Delta c_m = c_{(m+1)} - c_{(m)}. \tag{10}$$

Proof of Lemma 3.1. For notational convenience, let $p_{(j)} = p_{\pi(j)}$. Since $c_{(j)} = c_{(1)} + \sum_{m<j} \Delta c_m$ and $\sum_j p_{(j)} = 1$,

$$\begin{aligned}\sum_{j=1}^{K} c_{(j)} p_{(j)} &= c_{(1)} + \sum_{j=1}^{K} p_{(j)} \sum_{m<j} \Delta c_m \\ &= c_{(1)} + \sum_{m=1}^{K-1} \Delta c_m \sum_{j=m+1}^{K} p_{(j)}.\end{aligned}$$

This proves (3). Put $Z_H = \sum_{j \in H_m} e^{s_j} > 0$ and $Z_L = \sum_{j \in L_m} e^{s_j} > 0$. Then

$$q_m = \frac{Z_H}{Z_H + Z_L} = \frac{1}{1 + \exp(\log Z_L - \log Z_H)} = \sigma(\mathrm{LSE}(s_{H_m}) - \mathrm{LSE}(s_{L_m}))\,.$$

This proves (4). □

The coefficients in c need not be distinct. A zero gap contributes zero under any ordering within a tie. If all coefficients are equal, every gap is zero and the exact result is the common coefficient. For $K = 1$, the same empty-sum convention applies. We can now focus on bounding one group probability; the weighted sum will be assembled after that bound is established.

A.2 LSE INEQUALITIES, EXACT RANGES AND THE REFERENCE VALUE

The sigmoid identity reduces a group probability to the difference between two log-sum-exp values. To turn it into an affine bound, we need three ingredients: an affine lower approximation to that difference, the interval containing its true values, and a reference value for choosing the sigmoid line. The lemmas below supply these ingredients in order.

Constructing the affine difference. A lower bound on the high-minus-low difference requires a lower bound for the high group and an upper bound for the low group. Convexity gives the first one directly. Set $t = (\ell + u)/2$; the tangent to the high-group LSE at this midpoint is

$$h_G(s_G) = \mathrm{LSE}(t_G) + w_G^\top (s_G - t_G), \qquad w_G = \mathrm{softmax}(t_G), \qquad h_G \le \mathrm{LSE}(s_G). \tag{11}$$

For the low group, we first bound each exponential from above by the line joining its endpoint values, called the endpoint chord. We will then bound the logarithm of their sum from above by a tangent. The common shift $\xi = \max_j u_j$ rescales the exponentials without changing the represented LSE. The chord slopes and midpoint values are

$$\alpha_j = \begin{cases} \dfrac{e^{u_j - \xi} - e^{\ell_j - \xi}}{u_j - \ell_j}, & \ell_j < u_j, \\ e^{\ell_j - \xi}, & \ell_j = u_j, \end{cases} \qquad y_j = \frac{e^{\ell_j - \xi} + e^{u_j - \xi}}{2}. \tag{12}$$

Adding the chord midpoint values gives $Y_G = \sum_{j \in G} y_j > 0$. Taking a logarithm and its tangent at Y_G produces the affine upper bound

$$r_G(s_G) = \xi + \log Y_G + \frac{\sum_{j \in G} \alpha_j (s_j - t_j)}{Y_G} \;\ge\; \mathrm{LSE}(s_G). \tag{13}$$

For a one-token group $G = \{j\}$, use $r_G(s_G) = s_j$ exactly. The affine difference in (5) is $h_{H_m}(s_{H_m}) - r_{L_m}(s_{L_m})$.

The next lemma verifies both sides of this construction. Its proof uses convexity for the high tangent, then the exponential chords and concavity of the logarithm for the low upper bound.

Lemma A.1 (Affine LSE bounds)**.** *For every nonempty group G and midpoint $t_G \in [\ell_G, u_G]$,* (11) *is a global affine LSE lower bound. Equation* (13)*, with* (12)*, is an affine upper bound on the group box. Replacing its singleton instance by s_j is sound and no looser.*

Proof. The LSE gradient is $w = \mathrm{softmax}(z)$ and its Hessian is $\mathrm{diag}(w) - ww^\top$. For any vector v, its quadratic form is the variance of v_j under weights w_j, hence nonnegative. LSE is convex, so its first-order inequality at t_G gives (11).

For a nondegenerate coordinate interval, convexity of the exponential gives

$$e^{s_j - \xi} \le y_j + \alpha_j(s_j - t_j) \qquad (\ell_j \le s_j \le u_j).$$

On a degenerate interval, the right side agrees at the only feasible coordinate, which is all that is required; the tangent there is not asserted to be a global exponential upper bound. Every feasible chord value is positive. Summing gives a positive affine function

$$Y_G(s_G) = Y_G + \sum_{j \in G} \alpha_j(s_j - t_j) \quad \text{with} \quad \sum_{j \in G} e^{s_j - \xi} \le Y_G(s_G).$$

Monotonicity of $\log$ and its concavity yield

$$\mathrm{LSE}(s_G) \le \xi + \log Y_G(s_G) \le \xi + \log Y_G + \frac{Y_G(s_G) - Y_G}{Y_G},$$

which is (13). If $G = \{j\}$, then $\mathrm{LSE}(s_G) = s_j$ exactly; this is no larger than any sound upper plane. □

Subtracting these two planes therefore gives the affine lower difference $\underline{d}_m$ needed by the construction. The shift ξ affects its numerical evaluation, but not the represented bound in real arithmetic.

Finding the true difference range. The sigmoid line must be valid wherever the true LSE difference can lie. The groups use disjoint coordinates, so the endpoints can be found by moving the high and low groups in opposite directions. The exact range of the true difference is

$$D_m^- = \mathrm{LSE}(\ell_{H_m}) - \mathrm{LSE}(u_{L_m}), \qquad D_m^+ = \mathrm{LSE}(u_{H_m}) - \mathrm{LSE}(\ell_{L_m}). \tag{14}$$

Lemma A.2 (Exact range of a disjoint LSE difference)**.** *For nonempty disjoint H, L, the exact minimum and maximum of* $\mathrm{LSE}(s_H) - \mathrm{LSE}(s_L)$ *on $[\ell, u]$ are*

$$\mathrm{LSE}(\ell_H) - \mathrm{LSE}(u_L), \qquad \mathrm{LSE}(u_H) - \mathrm{LSE}(\ell_L).$$

Proof. Each LSE is strictly increasing in every coordinate. The difference is minimized by setting $s_H = \ell_H$ and $s_L = u_L$, and maximized by the opposite choices. Disjointness makes each pair of choices simultaneously feasible. Thus the endpoints are attained rather than merely interval enclosures. □

Locating the reference value used for selection. We choose among valid sigmoid lines by comparing their values at one reference value. The construction uses the affine difference evaluated at the score midpoint, $r_m^\star = \underline{d}_m(t)$. The next lemma shows that this particular value lies inside the true difference interval, so the candidate lines can be compared there.

Lemma A.3 (The reference value lies in the true interval)**.** *With the midpoint references in* (11)–(13)*, $r_m^\star = \underline{d}_m(t)$ satisfies $D_m^- \le r_m^\star \le D_m^+$.*

Proof. The high tangent is exact at its reference, so $h_{H_m}(t_{H_m}) = \mathrm{LSE}(t_{H_m}) \ge \mathrm{LSE}(\ell_{H_m})$. For a generic low group, $y_j \le e^{u_j - \xi}$ implies $r_{L_m}(t_{L_m}) = \xi + \log Y_{L_m} \le \mathrm{LSE}(u_{L_m})$. The singleton identity has the same property. Hence $\underline{d}_m(t) \ge D_m^-$. The other direction follows from $\underline{d}_m(t) \le d_m(t) \le D_m^+$. □

The reference value lies in the required interval, although $\underline{d}_m(s)$ can leave it elsewhere in the box. Clipping the reference value to the interval therefore has no effect in exact arithmetic. The next section explains why an affine approximation outside the interval still gives a valid composition.

A.3 THE EXACT INTERCEPT OF A SELECTED SIGMOID SLOPE

We now have an affine lower difference and an interval for the true difference. The remaining task is to place a line below the sigmoid on that interval. Once a nonnegative slope is chosen, endpoints and stationary points determine the highest valid line. A second lemma then justifies substituting the affine lower difference into that line.

Placing a line with a chosen slope. For a fixed slope, raising the intercept moves the whole line upward. The largest permitted intercept is the smallest vertical difference between the sigmoid and the zero-intercept line with that slope:

$$\kappa_m(\lambda_m) = \min_{z\in[D_m^-, D_m^+]} \{\sigma(z) - \lambda_m z\}. \tag{15}$$

Only the endpoints and interior stationary points need to be checked in this minimization. The following lemma identifies them by setting the derivative of the vertical difference to zero.

Lemma A.4 (Fixed-slope sigmoid minorant)**.** *Let $a \le b$ be finite and $0 \le \lambda \le 1/4$. The largest κ such that $\lambda z + \kappa \le \sigma(z)$ for all $z \in [a, b]$ is*

$$\kappa(\lambda) = \min_{z\in[a,b]} \{\sigma(z) - \lambda z\}.$$

It is obtained by evaluating the endpoints and the in-interval points

$$z = \log\frac{p}{1-p}, \qquad p \in \left\{\frac{1-\sqrt{1-4\lambda}}{2}, \frac{1+\sqrt{1-4\lambda}}{2}\right\} \cap (0,1). \tag{16}$$

Repeated points need be evaluated only once.

Proof. The inequality is equivalent to $\kappa \le \sigma(z) - \lambda z$ for every feasible z, so the displayed minimum is necessary and sufficient. A continuously differentiable function on a compact interval attains its minimum at an endpoint or an interior stationary point. For $p = \sigma(z)$, stationarity is $p(1-p) - \lambda = 0$, whose roots are exactly the probabilities in (16). A root in $(0, 1)$ has the unique finite logit shown there; roots outside the interval are discarded. □

If $\lambda = 0$, there is no finite stationary point and $\kappa = \sigma(a)$. If $\lambda = 1/4$, the two roots coincide at $z = 0$. If $a = b$, every maximal fixed-slope intercept gives a line that touches the sigmoid at that single point; the reported implementation uses the deterministic constant choice $\lambda = 0$. These cases do not require numerical root searches.

Substituting the affine difference. The line is valid on the true interval, but its affine input approximation may fall below that interval. Nonnegativity of the slope resolves this issue: first compare the two line values, then apply the sigmoid bound at the true argument. This is the step that connects the LSE construction to a valid group-probability bound.

Lemma A.5 (Composition uses the true argument interval)**.** *Suppose $d(s) \in [a, b]$, $\underline{d}(s) \le d(s)$, and $\lambda z + \kappa \le \sigma(z)$ for $z \in [a, b]$, with $\lambda \ge 0$. Then $\lambda\underline{d}(s) + \kappa \le \sigma(d(s))$ on the domain, whether or not $\underline{d}(s) \in [a, b]$.*

Proof. Multiplication by a nonnegative slope gives $\lambda\underline{d}(s) + \kappa \le \lambda d(s) + \kappa$. The latter is at most $\sigma(d(s))$ because the *true* argument belongs to the interval on which the sigmoid minorant is valid. □

We therefore have a valid affine lower bound for each group probability without clipping the affine function. Only the true difference must lie in the interval used to construct the sigmoid line.

A.4 PROOF OF LOCAL SOUNDNESS

The ingredients above now establish the bound for the complete weighted softmax expression. Each group has a valid affine lower bound, and the nonnegative coefficient gaps let us add those bounds without reversing an inequality.

Each group contributes a score coefficient vector and a constant. Their weighted sums are the two parts of the final affine bound:

$$A_c = \sum_{m=1}^{K-1} \Delta c_m \lambda_m v_m, \qquad B_c = c_{(1)} + \sum_{m=1}^{K-1} \Delta c_m (\lambda_m \rho_m + \kappa_m). \tag{17}$$

The proof first applies the preceding lemmas to one group, then uses the ordered-mass identity to combine the groups. The single-score and zero-gap cases require no separate approximation.

Proof of Theorem 3.2. For $K = 1$, softmax is one and the claimed return is exact. For $K \geq 2$, fix any $s \in [\ell, u]$. Lemma A.1 gives $\underline{d}_m(s) \leq d_m(s)$, Lemma A.2 gives $d_m(s) \in [D_m^-, D_m^+]$, and Lemma A.4 provides a sound sigmoid line. By Lemma A.5,

$$q_m(s) \geq \lambda_m (v_m^\top s + \rho_m) + \kappa_m.$$

Since $\Delta c_m \geq 0$, multiplying and summing preserves every inequality. Lemma 3.1 then gives

$$\begin{aligned} F_c(s) &= c_{(1)} + \sum_m \Delta c_m q_m(s) \\ &\geq c_{(1)} + \sum_m \Delta c_m \{\lambda_m (v_m^\top s + \rho_m) + \kappa_m\} = A_c^\top s + B_c. \end{aligned}$$

Zero-gap terms may be omitted. Applying the established lower-bound result to $-c$ and multiplying by -1 gives $F_c(s) \leq -A_{-c}^\top s - B_{-c}$. □

This completes the validity argument for the local affine bound. We next ask what retaining this affine dependence can achieve compared with returning a constant; Appendix B later shows how to evaluate the same bound efficiently.

A.5 Comparing the scalar bound with propagation to the input

Validity alone does not explain why an affine bound can outperform a constant after propagation. On the same independent score box, the exact scalar minimum is already the strongest constant bound. The opportunity appears when propagation uses relationships between scores that the box omits. The identity below separates this opportunity from the losses introduced by the affine relaxation and propagation toward the input.

Write $V_{\mathcal{B}}(c) = \min_{s \in \mathcal{B}} F_c(s)$ for the exact scalar minimum. For any sound affine lower bound g, comparison on the same box gives

$$\min_{s \in \mathcal{B}} g(s) \leq V_{\mathcal{B}}(c). \tag{18}$$

To compare the completed bounds, we distinguish three quantities. The exact objective has infimum $V_{\mathcal{S}} = \inf_{s \in \mathcal{S}} F_c(s)$ over the reachable scores. Replacing the objective by its affine lower bound gives $G_{\mathcal{S}} = \inf_{s \in \mathcal{S}} g(s)$. Finally, let $P_{\mathcal{X}}(g)$ be the finite lower bound obtained through sound propagation toward the input, so $P_{\mathcal{X}}(g) \leq G_{\mathcal{S}}$. Adding and subtracting these quantities gives

$$\begin{aligned} P_{\mathcal{X}}(g) - V_{\mathcal{B}}(c) = \; & \underbrace{V_{\mathcal{S}} - V_{\mathcal{B}}(c)}_{\text{gain from the reachable set}} - \underbrace{V_{\mathcal{S}} - G_{\mathcal{S}}}_{\text{loss from the affine relaxation}} \\ & - \underbrace{G_{\mathcal{S}} - P_{\mathcal{X}}(g)}_{\text{loss from propagation to the input}}. \end{aligned} \tag{19}$$

The three gaps have a direct interpretation: restricting to reachable scores can help, while relaxing the objective and propagating it approximately can lose some of that benefit. The proposition verifies their signs.

Proposition A.6 (Comparison for an isolated weighted-softmax term)**.** *If $g(s) \leq F_c(s)$ throughout the compact box $\mathcal{B}$, then* (18) *holds. If $\varnothing \neq \mathcal{S} \subseteq \mathcal{B}$ and $P_{\mathcal{X}}(g) \leq \inf_{s \in \mathcal{S}} g(s)$ is finite, then* (19) *holds and its three subtracted or added gaps are nonnegative.*

Proof. Continuity of F_c on the compact box gives a minimizer s^*. Therefore

$$\min_{s\in\mathcal{B}} g(s) \le g(s^*) \le F_c(s^*) = V_{\mathcal{B}}(c).$$

Restriction of the feasible set gives $V_{\mathcal{B}}(c) \le V_{\mathcal{S}}$, and pointwise soundness gives $G_{\mathcal{S}} \le V_{\mathcal{S}}$. The remaining inequality, $P_{\mathcal{X}}(g) \le G_{\mathcal{S}}$, is the assumed soundness of propagation toward the input. All values are finite because F_c and g are bounded on $\mathcal{B}$. Adding and subtracting $V_{\mathcal{S}}$ and $G_{\mathcal{S}}$ proves (19). □

The affine bound improves this comparison for one weighted-softmax term when the reachable-set gain exceeds the two approximation losses. A full graph may also constrain the relationship between this term and other terms in the verification objective; the separation result below needs only the relationships among scores. Before proving it, we specify the exact scalar control used in the experiments.

A.6 THE EXACT SCALAR CONTROL

The strongest constant over the score box can be found by checking a finite set of coefficient-ordered endpoint assignments. We give the formula and justify why these assignments suffice. This defines the exact-scalar control used to compare constant and affine bounds.

The finite formula below restates the threshold result of Rezazadeh & Davoodi (2026). We include it to define the scalar ablation in Section 5; the scalar solver and its ordering principle are attributed results.

In the same coefficient order $c_{(1)} \le \cdots \le c_{(K)}$, let

$$\xi = \max_j u_j, \qquad E^-_{(j)} = \exp(\ell_{(j)} - \xi), \qquad E^+_{(j)} = \exp(u_{(j)} - \xi).$$

An endpoint assignment gives large exponential weights to the first m coefficients and small weights to the rest. Its objective value is the following ratio, for $m \in \{0, \dots, K\}$:

$$\tau_m = \frac{\sum_{j\le m} c_{(j)} E^+_{(j)} + \sum_{j>m} c_{(j)} E^-_{(j)}}{\sum_{j\le m} E^+_{(j)} + \sum_{j>m} E^-_{(j)}}. \tag{20}$$

Empty sums are zero. The denominator is strictly positive in real arithmetic, including at $m = 0$ and $m = K$.

Proposition A.7 (Exact independent-box scalar; attributed restatement)**.** *For every fixed $c \in \mathbb{R}^K$ and finite $\ell \le u$,*

$$V_{\mathcal{B}}(c) = \min_{s\in[\ell,u]} c^\top \operatorname{softmax}(s) = \min_{0\le m\le K} \tau_m. \tag{21}$$

Thus the local return $(a, b) = (0, V_{\mathcal{B}}(c))$ is sound and no valid constant can be larger. The formula handles coefficient ties, degenerate coordinate intervals and $K = 1$. It uses $O(K \log K)$ operations and $O(K)$ storage under the same elementary-function cost model as Theorem 3.4.

The proof rewrites the objective as a ratio over positive variables. At its minimum, a linear comparison reveals which variables should take their upper or lower endpoints; sorting makes those choices a single threshold.

Proof. The change of variables $y_j = e^{s_j - \xi}$ is a bijection from the score box to the positive box $[E^-, E^+]$. It transforms the objective into

$$R(y) = \frac{\sum_j c_j y_j}{\sum_j y_j}.$$

Let μ be its attained minimum. Because the denominator is positive,

$$R(y) \ge \mu \quad\Longleftrightarrow\quad \sum_j (c_j - \mu) y_j \ge 0.$$

The linear function on the right therefore has minimum zero on the positive box. Its coordinates minimize independently: choose $y_j = E_j^+$ if $c_j < \mu$, choose $y_j = E_j^-$ if $c_j > \mu$, and either endpoint if $c_j = \mu$. In sorted order, choosing the upper endpoint also for ties gives a threshold with $m = \#\{j : c_j \le \mu\}$. This point minimizes the linear function, so its value is zero and its ratio is μ. Every other threshold is feasible and therefore has ratio at least μ. This proves (21) without assuming distinct coefficients or nonzero interval widths.

The zero-slope return equals the minimum of the objective throughout the box, and any larger constant would fail at a minimizer. When $K = 1$, every ratio is c_1. Sorting takes $O(K \log K)$ operations; prefix sums of $c_{(j)}E^+_{(j)}$ and $E^+_{(j)}$, together with suffix sums of $c_{(j)}E^-_{(j)}$ and $E^-_{(j)}$, evaluate all $K + 1$ ratios in linear additional work and storage. □

This control changes only the local bound supplied to the verifier. With incoming c fixed, all score coefficients for this term are set to zero and its exact minimum is added to the constant term in backward propagation; other graph terms remain unchanged. The corresponding constant upper bound is $-V_{\mathcal{B}}(-c)$. The control uses the fixed-parameter selection procedure in Appendix D.5. It evaluates the local scalar substitution within the same host verifier, rather than the full verifier described in the cited preprint.

This joint minimum differs from minimizing every nested mass separately, as Proposition A.10 explains. The scalar evaluation with numerical safeguards and its independent check using logarithms are described with the numerical scope in Appendices C and D.5.

A.7 Proof of the separation theorem

The comparison above leaves open whether the gain from score dependence can survive the affine approximation. The two-score construction in Proposition 3.3 shows that it can: both scores share an input, and their difference is fixed. The tangent bound retains this equality and becomes exact after propagation.

We use the same construction in the theorem below. Its parameters correspond to the main text through $A = a$, $R = r$, and $\varepsilon = a - 2r > 0$. Introducing ε also lets us vary the family while keeping the largest difference $s_2 - s_1$ permitted by the box negative. The theorem calls the smallest box containing the reachable scores their coordinate hull. We first prove the strict improvement, then quantify its ratio, and finally verify that the six-candidate rule returns the required tangent.

Theorem A.8 (Separation after correlated composition)**.** *Fix $A > \varepsilon > 0$, put $R = (A - \varepsilon)/2$, and let*

$$\mathcal{X} = [-R, R], \qquad s(x) = (x, x - A), \qquad c = (0, 1).$$

Let $\mathcal{B} = [-R, R] \times [-R - A, R - A]$ be the independent coordinate hull. Its exact scalar minimum is $V_{\mathcal{B}}(c) = \sigma(-2A + \varepsilon)$. The affine function

$$g_A(s) = \sigma(-A) + \sigma'(-A)(s_2 - s_1 + A) \tag{22}$$

is a sound lower bound on $\mathcal{B}$ and satisfies $g_A(s(x)) = F_c(s(x)) = \sigma(-A)$ for every $x \in \mathcal{X}$. Consequently, for fixed ε,

$$\frac{\inf_{x\in\mathcal{X}} g_A(s(x))}{V_{\mathcal{B}}(c)} = \frac{1 + e^{2A-\varepsilon}}{1 + e^A} \longrightarrow \infty \quad \text{as } A \to \infty. \tag{23}$$

The positive additive difference is asymptotic to e^{-A} and tends to zero.

The proof compares the single score difference allowed by the graph with the interval allowed by the box. A tangent on the convex branch of the sigmoid retains the graph's equality. The final step evaluates the ratio and absolute difference separately.

Proof of Theorem A.8. For $c = (0, 1)$, $F_c(s) = \sigma(s_2 - s_1)$. On the reachable set, $s_2(x) - s_1(x) = -A$, so $F_c(s(x)) = \sigma(-A)$. The coordinate hull permits

$$s_2 - s_1 \in [-2R - A, 2R - A] = [-2A + \varepsilon, -\varepsilon].$$

Both endpoints are attained in the independent box. Since the sigmoid is strictly increasing, its exact box minimum is $\sigma(-2A + \varepsilon)$.

The entire difference interval is negative. The sigmoid satisfies

$$\sigma'(z) = \sigma(z)(1-\sigma(z)), \qquad \sigma''(z) = \sigma'(z)(1-2\sigma(z)),$$

so it is convex on $(-\infty, 0]$. Its tangent at the interior point $-A$ is therefore a minorant on the difference interval. Substituting $z = s_2 - s_1$ gives (22), and substituting $s(x)$ makes its linear term vanish. Because the score map is affine, exact affine propagation preserves this cancellation.

Using $\sigma(-z) = 1/(1+e^z)$,

$$\frac{\sigma(-A)}{\sigma(-2A+\varepsilon)} = \frac{1+e^{2A-\varepsilon}}{1+e^{A}} \sim e^{A-\varepsilon} \longrightarrow \infty.$$

Meanwhile $\sigma(-A) \sim e^{-A}$ and $\sigma(-2A+\varepsilon) \sim e^{-2A+\varepsilon}$, so their difference is positive and asymptotic to e^{-A}. □

The growing ratio concerns a probability with the fixed normalization $c = (0, 1)$; shifting the objective would change that ratio. The absolute improvement tends to zero in this family. The strict improvement, rather than a large additive margin, is what establishes the mechanism in the main text.

Why the implemented rule selects this bound. The existence of a suitable tangent is not enough: the stated method must actually select it. When $K = 2$, the high and low LSE groups each contain one token, so (5) is exact. The reference value derived from the midpoint for this family is $r_1^\star = -A$. Choosing $\lambda_1 = \sigma'(-A)$ makes (15) the tangent intercept, since the tangent is sound and touches at $-A$. It attains the largest possible line value at the reference, $\sigma(-A)$. Any selection rule that includes this tangent and maximizes the line value at the reference therefore returns the same tangent; differentiability at the interior contact point fixes its slope. The six-candidate family explicitly contains the reference derivative (Appendix B.1). The theorem does not depend on which of the remaining candidates are included.

A.8 Choosing the sigmoid lower line at a reference value

The default method chooses from six slopes. The reference-optimal ablation asks how much could be gained by choosing the best permitted line at the same reference value. The answer has a geometric form: a tangent on a negative interval, a chord on a positive interval, and a tangent that reaches the right endpoint when the interval crosses zero. The optimized quantity is the line value at that reference value; it is not the minimum of the affine score bound or the completed CROWN bound.

The theorem separates these cases, and the proof follows the same order. When the interval contains both negative and positive values, the main task is to locate the point where a tangent reaches the right endpoint.

Theorem A.9 (Reference-optimal minorant on a finite interval)**.** *Let $a \le r \le b$. Among affine sigmoid minorants on $[a, b]$ with slopes in $[0, 1/4]$, a line maximizing its value at r is given by the following cases.*

1. *If $a = b$, the constant $\sigma(a)$ is a deterministic optimum.*
2. *If $a < b \le 0$, the tangent at r is optimal and attains $\sigma(r)$.*
3. *If $0 \le a < b$, the chord joining $(a, \sigma(a))$ and $(b, \sigma(b))$ is optimal.*
4. *If $a < 0 < b$, there is a unique $t_0 < 0$ satisfying*

$$\sigma'(t_0) = \frac{\sigma(b) - \sigma(t_0)}{b - t_0}. \tag{24}$$

If $t_0 < a$, the chord joining the two interval endpoints is optimal for every r. If $t_0 \ge a$, the tangent at r is optimal for $r \le t_0$, and the line joining $(t_0, \sigma(t_0))$ and $(b, \sigma(b))$ is optimal for $r \ge t_0$.

All listed slopes belong to $[0, 1/4]$.

Proof. The singleton case is immediate. On a negative interval, convexity makes the tangent a valid minorant, and equality at r attains the largest possible value. On a positive interval, concavity makes the endpoint chord a minorant. Every other affine minorant is bounded by the sigmoid at both endpoints, and hence by the chord at every convex combination of those endpoints. This proves its optimality.

Locating the join in a mixed interval. The gap between the sigmoid at b and the tangent at t evaluated at b vanishes exactly when that tangent reaches the right endpoint. Define

$$G(t) = \sigma(b) - \sigma(t) - \sigma'(t)(b - t), \qquad t < 0.$$

Differentiation gives $G'(t) = -\sigma''(t)(b - t) < 0$. Also $G(t) \to \sigma(b) > 0$ as $t \to -\infty$, whereas

$$G(0) = \sigma(b) - \tfrac{1}{2} - \tfrac{b}{4} < 0,$$

because $\sigma'(z) < 1/4$ for $z > 0$. Continuity and strict monotonicity give a unique root $t_0 < 0$.

When the join lies inside the interval. Suppose first that $t_0 \geq a$, and let C join $(t_0, \sigma(t_0))$ and $(b, \sigma(b))$. Equation (24) makes C the tangent at t_0. Convexity gives $C \leq \sigma$ on $[a, 0]$. On $[0, b]$, the difference $\sigma - C$ is concave and nonnegative at both endpoints, so it remains nonnegative throughout. Thus C is a valid minorant on the full interval.

Now consider a reference $r \leq t_0$. Its tangent T_r lies below the sigmoid on $[a, t_0]$. At t_0, its value is at most $C(t_0)$. Its slope is also at most C', because σ' increases on the negative half-line. Hence $T_r \leq C$ for $z \geq t_0$, establishing validity on the full interval. Since $T_r(r) = \sigma(r)$, it is optimal at r.

For a reference $r \in [t_0, b]$, any affine minorant is bounded by the sigmoid at t_0 and b. It is therefore bounded by their joining line at r. This proves the optimality of C.

When the join lies to the left. Now suppose $t_0 < a$. Let $C_{a,b}$ be the endpoint chord and k its slope. Since G is strictly decreasing, $G(a) < 0$, which gives $\sigma'(a) > k$. The difference $H = \sigma - C_{a,b}$ satisfies $H(a) = H(b) = 0$, $H'(a) > 0$, and $H'' > 0$ on $[a, 0)$. Therefore $H \geq 0$ on $[a, 0]$. On $[0, b]$, H is concave with nonnegative endpoint values, so again $H \geq 0$. Thus the endpoint chord is a valid minorant. Endpoint interpolation bounds every other affine minorant by that chord, proving optimality. This argument checks the chord directly; restricting a convex envelope from a larger interval would not by itself establish this case.

Tangent slopes lie in $(0, 1/4]$. By the mean value theorem, every nondegenerate chord slope is also in $(0, 1/4]$; the singleton constant has slope zero. □

The theorem defines the reference-optimal variant used in the experiments. It also explains why a line that is best at the reference value need not be best after propagation: the two comparisons optimize different quantities.

Approximating the optimal slope. When the interval crosses zero, the tangent contact point may be approximated by repeatedly bisecting an interval that contains it. We therefore need to distinguish a slightly suboptimal slope from an invalid line. For any chosen slope in $[0, 1/4]$, recomputing (15) gives a valid line in real arithmetic. To bound the loss in reference value, write

$$\phi_r(\lambda) = \lambda r + \kappa(\lambda) = \min_{z \in [a,b]} \{\sigma(z) + \lambda(r - z)\},$$

Then

$$|\phi_r(\lambda) - \phi_r(\lambda')| \leq \max\{r - a, b - r\}|\lambda - \lambda'| \leq (b - a)|\lambda - \lambda'|. \tag{25}$$

To see this, each function inside the minimum is Lipschitz in λ with the displayed common constant, and taking a minimum preserves that inequality in both directions. The reference-value loss is therefore controlled by the slope error and interval width. Root-finding remains separate from the constant-size policy assumed by the scan theorem; floating-point safeguards are addressed in Appendix C.

A.9 Constant group bounds and common-score shifts

The remaining results explain what constant group bounds retain and how the affine construction behaves when the same constant is added to every score. Setting every sigmoid slope to zero can lose

information even before propagation, because the group minima may occur at different score points. Rebuilding the plane after a common shift preserves its represented values, although the coefficients of a fixed affine bound need not sum to zero. We establish these properties in turn.

Why constant group bounds differ from the exact scalar. Each group has an exact minimum, but attaining all those minima at one score vector is an additional requirement. The following proposition identifies it as precisely the condition for the combined constant group bounds to equal the exact minimum of the whole weighted expression.

Proposition A.10 (What setting all slopes to zero computes)**.** *The choice $\lambda_m = 0$ returns*

$$C = c_{(1)} + \sum_m \Delta c_m \sigma(D_m^-) \le V_{\mathcal{B}}(c).$$

Equality holds if and only if some score-box point minimizes every mass whose coefficient gap is positive.

Proof. Each $q_m(s) \ge \sigma(D_m^-)$ by its exact range and sigmoid monotonicity, which gives the inequality after summation. The continuous difference

$$F_c(s) - C = \sum_m \Delta c_m \{q_m(s) - \sigma(D_m^-)\}$$

is a sum of nonnegative terms and attains its minimum on the compact box. That minimum is zero precisely when all terms with positive weight vanish at one common point. □

For example, two consecutive thresholds can require a shared coordinate to be at opposite endpoints. The comparison using constant group bounds therefore does not generally compute the exact scalar minimum $V_{\mathcal{B}}(c)$, even though each group's minimum is exact.

What happens under a common score shift. Softmax probabilities do not change when the same constant is added to every score. For the affine construction, we must distinguish rebuilding the plane on a shifted box from evaluating one fixed plane at shifted scores. The next result shows that rebuilding behaves consistently while a fixed affine bound can still increase when the same constant is added to every score. The proof first tracks the translated construction, then sums its coefficients.

Proposition A.11 (Shift equivariance is not a zero-sum slope)**.** *Consider the midpoint construction with a candidate policy determined only by $(D_m^-, D_m^+, r_m^\star)$. Recomputing the plane after translating both score bounds by $\tau\mathbf{1}$ yields*

$$A_c^{\mathrm{new}} = A_c, \qquad B_c^{\mathrm{new}} = B_c - \tau A_c^\top \mathbf{1},$$

so $g_c^{\mathrm{new}}(s + \tau\mathbf{1}) = g_c(s)$. A fixed returned lower plane nevertheless satisfies $A_c^\top \mathbf{1} \ge 0$ and may have a strictly positive slope in the common-score direction.

Proof. Translation also sends t and ξ to $t + \tau\mathbf{1}$ and $\xi + \tau$. The shifted exponential statistics and chord slopes are unchanged. Each LSE lower or upper plane, evaluated at the translated score, equals its original value plus τ; the exact singleton rule has the same property. Their difference, its exact range and its scalar reference are unchanged under this paired translation. Thus the candidate policy and each λ_m, κ_m remain unchanged, giving the stated affine transformation law.

For the sign of the common-score slope, write each low-group interval as $[t_j - r_j, t_j + r_j]$, where $r_j \ge 0$. The generic low upper plane has slope sum

$$\theta_G = \frac{\sum_{j\in G} \alpha_j}{\sum_{j\in G} y_j} = \sum_{j\in G} \omega_j \frac{\tanh r_j}{r_j}, \qquad \omega_j = \frac{y_j}{Y_G} > 0, \quad \sum_j \omega_j = 1, \tag{26}$$

with the continuous value 1 when $r_j = 0$. Since $0 < \tanh r/r \le 1$, $0 < \theta_G \le 1$. The high tangent has slope sum one, and the exact low-singleton rule also has slope sum one. Thus every difference plane has nonnegative slope sum. Summing it with the nonnegative weights $\Delta c_m \lambda_m$ gives $A_c^\top \mathbf{1} \ge 0$. It is strictly positive whenever an active generic low group has positive interval width and receives positive weight. □

Rebuilding the plane therefore respects a common shift, even though evaluating one fixed plane need not.

B CUMULATIVE SUMS AND INTEGRATION WITH THE VERIFIER

Appendix A establishes the affine bound. Here we show how to compute that same bound efficiently and return it through the verifier. We first fix the slope choices so that the direct and scanned constructions refer to the same function. Shared cumulative sums then recover its coefficients with near-linear work and linear storage. Finally, we prove that substituting the affine bound and selecting between completed bounds preserve validity.

B.1 FIXING THE SIGMOID-LINE SELECTION RULE

Any valid slope gives a valid local bound. To show that the scan matches the direct construction, however, both must choose the same affine function. We must therefore fix which candidate sigmoid lines are considered and how ties are resolved. An ordered list with a constant number of slopes supplies this agreement and keeps selection work constant per threshold. The default rule uses six candidate sigmoid lines.

The mathematical algorithm takes an ordered list

$$\Lambda(D^-, D^+, r) = (\lambda^{(1)}, \dots, \lambda^{(J)}), \qquad \lambda^{(i)} \in [0, 1/4],$$

where J is independent of K. For each entry it computes the maximal valid intercept from Lemma A.4, then chooses the first entry attaining the largest value of $\lambda^{(i)} r + \kappa(\lambda^{(i)})$. This ordering makes tied choices reproducible; every tied line remains valid. When the interval has zero width, we use $\lambda = 0$, $\kappa = \sigma(D^-)$. A tangent with nonzero slope would also be valid at that single point, but would give a different affine bound.

The default list includes zero for a constant bound, the slope of the line joining the sigmoid endpoints, derivatives at the endpoints and reference value, and the maximum sigmoid derivative. Every candidate slope receives its own valid intercept. Write $a = D^-$, $b = D^+$ and clip the reference value r to $[a, b]$. For $a < b$, the ordered list is

$$\Lambda(a, b, r) = \left(0, \frac{\sigma(b) - \sigma(a)}{b - a}, \sigma'(a), \sigma'(r), \sigma'(b), \frac{1}{4}\right). \tag{27}$$

The implementation clamps each candidate slope to $[0, 1/4]$ and applies the intercept calculation, tie rule, and zero-width convention above. If a candidate evaluation produces NaN or infinity, the verifier uses the baseline bound.

The reference-derivative-only and reference-optimal policies are separate experimental variants; Appendix A.8 defines the latter. We can now derive the scan while holding the policy fixed. Numerical input checks and fallbacks are described in Appendix C.

B.2 SHARED SUMS AND PROOF THAT THE SCAN GIVES THE SAME BOUND

With the sigmoid-line selection rule fixed, the remaining work is to assemble the selected group bounds. Their groups overlap as prefixes and suffixes, so most terms can be shared. We first express the group quantities as cumulative sums, then collect each score's contributions, and finally prove that the result equals the direct construction.

Sharing the group calculations. For $K = 1$, the affine coefficients and constant are exactly $(0, c_1)$; the formulas below concern $K \geq 2$. Work in coefficient-sorted order, suppressing the permutation in coordinate subscripts. The endpoint and midpoint exponentials are the reusable factors:

$$t_j = (\ell_j + u_j)/2, \quad \xi = \max_j u_j, \quad \delta_j = u_j - \ell_j, \qquad E_j^- = e^{\ell_j - \xi}, \quad E_j^+ = e^{u_j - \xi}, \quad E_j^0 = e^{t_j - \xi}.$$

The chord from Appendix A.2 takes the following form in these factors:

$$e^{s_j - \xi} \leq y_j + \alpha_j (s_j - t_j), \qquad y_j = (E_j^- + E_j^+)/2, \quad \alpha_j = \begin{cases} (E_j^+ - E_j^-)/\delta_j, & \delta_j > 0, \\ E_j^-, & \delta_j = 0. \end{cases} \tag{28}$$

Table 5 groups the quantities needed at each threshold. The first two rows determine the true difference range; the last two supply the affine bounds and their reference values. Every entry is available from a prefix or suffix sum.

Table 5: Sufficient statistics for each threshold $1 \le m < K$. All eight arrays are computed by prefix or suffix scans.

Low prefix	High suffix
$L_m^- = \sum_{j \le m} E_j^-$	$H_m^- = \sum_{j > m} E_j^-$
$L_m^+ = \sum_{j \le m} E_j^+$	$H_m^+ = \sum_{j > m} E_j^+$
$Y_m = \sum_{j \le m} y_j$	$M_m = \sum_{j > m} E_j^0$
$T_m = \sum_{j \le m} \alpha_j t_j$	$N_m = \sum_{j > m} E_j^0 t_j$

Recovering the difference and reference value. We now express the two affine group bounds using these shared sums. The high-group tangent has coefficient E_j^0/M_m on $j > m$ and bias $\xi + \log M_m - N_m/M_m$. For $m \ge 2$, the low-group upper plane has coefficient α_j/Y_m on $j \le m$ and bias $\xi + \log Y_m - T_m/Y_m$. Thus the constant terms of the affine differences are

$$\rho_1 = \xi + \log M_1 - N_1/M_1, \tag{29}$$

$$\rho_m = \log M_m - \log Y_m - N_m/M_m + T_m/Y_m, \quad m \ge 2. \tag{30}$$

The first expression uses the exact identity $r_{L_1}(s) = s_1$; applying the generic low formula to this case would produce a different plane. The range of the true group difference and the reference value also follow from the cumulative sums:

$$D_m^- = \log H_m^- - \log L_m^+, \qquad D_m^+ = \log H_m^+ - \log L_m^-, \tag{31}$$

$$r_1^\star = \xi + \log M_1 - t_1, \qquad r_m^\star = \log M_m - \log Y_m \quad (m \ge 2). \tag{32}$$

The reference value belongs to the range of the true group difference by Lemma A.3; clipping it to that interval does nothing in exact arithmetic.

Collecting each score's contributions. A coordinate belongs to the high group for thresholds before it and to the low group for thresholds at or after it. This gives two cumulative sums for its coefficient. After selecting the sigmoid line for each group, write $\gamma_m = \Delta c_m \lambda_m$ and define

$$P_j = \sum_{m<j} \frac{\gamma_m}{M_m}, \qquad Q_j = \sum_{m=\max\{j,2\}}^{K-1} \frac{\gamma_m}{Y_m}. \tag{33}$$

Empty sums are zero. The high-group sum adds to the coefficient, the low-group sum subtracts from it, and the exact formula for a one-token low group contributes one separate correction. The resulting coefficient vector and constant are

$$(A_c)_{(j)} = E_j^0 P_j - \alpha_j Q_j - \gamma_1 \mathbf{1}_{\{j=1\}}, \quad 1 \le j \le K, \tag{34}$$

$$B_c = c_{(1)} + \sum_{m=1}^{K-1} \Delta c_m (\lambda_m \rho_m + \kappa_m). \tag{35}$$

The final vector is put back into the original coordinate order. To prove equivalence, we first check that the group calculations give the same selected lines, then match each coordinate's contributions. Counting the scans gives the complexity bound.

Proof of Theorem 3.4. Equations (29)–(32) expand the direct construction using the same midpoint, chord and singleton choices. Hence both implementations give the same difference interval, affine reference and difference bias at every threshold. Applying the same ordered candidate policy, exact fixed-slope intercept and deterministic tie rule gives identical (λ_m, κ_m).

Fix a coordinate j. Every threshold $m < j$ places it in the high group and contributes $\gamma_m E_j^0/M_m$ to its coefficient. These terms sum to $E_j^0 P_j$. Every generic low-group threshold $m \ge \max\{j, 2\}$ contributes $-\gamma_m \alpha_j / Y_m$, summing to $-\alpha_j Q_j$. Only coordinate 1 also receives the exact-singleton term $-\gamma_1$. This proves (34) coordinate by coordinate. Equation (35) is the direct scalar reduction, so the biases agree as well.

Sorting takes $O(K \log K)$ comparisons. The per-coordinate factors, eight group statistics, constant-size slope/intercept evaluations, two final scans, scalar bias reduction and inverse permutation take $O(K)$ operations and $O(K)$ storage. No K-by-$(K-1)$ intermediate array is required. The count charges elementary real functions unit cost; it is not a bit-complexity bound for exact transcendental arithmetic. □

The proof shows that the saving comes from sharing repeated sums, without changing the selected affine bound. Table 6 collects the formulas into an implementation specification, including the cases with constant bounds and restoration of the original score order. The direct quadratic construction remains a reference for checking the scan.

Table 6: Algebraic specification for the scan implementation. Algorithm 1 uses its six-candidate instance; this specification allows any ordered constant-size candidate policy. Floating-point safeguards are treated separately.

Input	Finite c, ℓ, u with $\ell \le u$; an ordered constant-size candidate policy Λ.
1	Stably sort by (c_j, j). For $K = 1$, return $(0, c_1)$. An all-equal direction likewise has an exact constant return.
2	Compute the shifted factors and the eight arrays in Table 5.
3	For each nonzero coefficient gap, compute $\rho_m, D_m^-, D_m^+, r_m^\star$. On a point interval use the constant convention; otherwise evaluate Λ, its exact intercepts, and the first reference maximizer. Zero-gap thresholds contribute zero.
4	Form γ_m, scan P_j, Q_j, and evaluate (34)–(35). Undo the coefficient permutation.
Output	(A_c, B_c) satisfying Theorem 3.2. For an upper bound, call the lower routine on $-c$ and negate both outputs.

B.3 Preserving validity when using the bound in a verifier

An efficient local bound is useful only if the verifier can substitute it while preserving validity. The final step is therefore to follow the affine bound through the surrounding computation graph. The proposition proves two facts: sound local substitution preserves the final lower bound, and selecting the larger completed bound gives a result at least as large as the baseline for the same constraint and input subdomain. We then specify which outputs are selected together.

Proposition B.1 (Matched-call soundness and lower-bound non-regression)**.** *Assume the score boxes and all other local backward rules are sound for the current input subdomain. Replacing each supported softmax term by Theorem 3.2 preserves the soundness of the final lower bound. For lower-bound vectors on the same subdomain and specifications, the elementwise selection in* (8) *is sound and no smaller than the CROWN-LSE vector. The statement applies independently to every sound branch-and-bound subdomain.*

The proof maintains a lower bound at every backward step, then checks each output constraint separately. Minimizing the propagated affine bound over the input domain gives the completed bound; the proof calls this step input concretization.

Proof. At the softmax node, the current expression contains the complete term $c^\top \mathrm{softmax}(s)$ for a fixed incoming c. Theorem 3.2 replaces it by a pointwise lower affine function on the certified score box. Adding the unchanged terms in the running expression preserves the lower-bound invariant. Subsequent sound local substitutions preserve it by induction over the remaining backward graph. Sound input concretization then yields L_{TM}.

For any scalar specification $f_i(x)$ on that subdomain, both $L_{\text{CROWN-LSE},i} \le f_i(x)$ and $L_{\mathrm{TM},i} \le f_i(x)$ hold for every feasible x. Their maximum therefore also lower-bounds $f_i(x)$ and is at least $L_{\text{CROWN-LSE},i}$. This argument applies independently to each specification. Repeating it on a child domain with sound intermediate bounds proves the nodewise claim. □

Returning a consistent result. The integration optimizes CROWN-LSE, holds its relaxation parameters α fixed, and runs the additional Threshold-Mass pass without further gradient optimization. The proposition applies to the completed lower-bound vectors. The same argument does not justify mixing coefficients, upper bounds, and lower bounds from different methods. If the host also requests upper bounds or linear coefficients, the implementation returns the whole candidate result only when every lower bound matches or exceeds its baseline counterpart; otherwise it returns the whole baseline. This preserves the lower-bound guarantee; the upper bounds need not improve.

The implementation supports softmax calls with dense coefficient tensors of compatible dimensions. Missing intermediate bounds, incompatible dimensions, or intermediate quantities that are NaN or infinite cause the verifier to use the baseline rule. A fallback is sound only under the same assumption that the baseline is sound; it is not an independent numerical certificate.

From a bound to a search outcome. The result preserves validity on each input subdomain. Whether the search can eventually settle the full property also depends on its domain coverage, branching, and termination or convergence conditions (Bunel et al., 2020; Shi et al., 2025). Under a timeout, the extra pass also costs time and can change later search decisions. This is why Appendix D measures certificate counts and resource use alongside bound improvements.

C Numerical implementation and validation

The preceding proofs describe real arithmetic. Storing the affine coefficients and constant introduces a further question: how can coefficient rounding change the bound? A correction to the constant term, computed from the score ranges, compensates for that change under the lemma's assumptions. We derive the correction, describe the numerical implementation, and then examine it in 360 local tests, all of which pass the reported checks.

Compensating for rounded coefficients. Rounding a coefficient changes the affine bound by an amount that depends on the score. To keep a lower bound valid throughout the box, the constant term must account for the worst such change over all feasible scores. Because the error is affine and the domain is a box, that worst case is found by choosing an endpoint of each score interval.

Lemma C.1 (Box-aware correction after coefficient rounding)**.** *Suppose $a^\top s + b \le F(s)$ on $[\ell, u]$. Let $\widehat{a}$ be any stored approximation to a, set $\delta = a - \widehat{a}$, and define*

$$\chi(\delta; \ell, u) = \sum_{j=1}^{K} \begin{cases} \delta_j \ell_j, & \delta_j \ge 0, \\ \delta_j u_j, & \delta_j < 0. \end{cases} \tag{36}$$

If $\widehat{b} \le b + \chi(\delta; \ell, u)$, then $\widehat{a}^\top s + \widehat{b} \le F(s)$ for all $s \in [\ell, u]$.

Proof. Separability of the box gives $\chi(\delta; \ell, u) = \min_{s \in [\ell, u]} \delta^\top s$. Therefore, at any feasible score,

$$\widehat{a}^\top s + \widehat{b} \le \widehat{a}^\top s + b + \chi(\delta; \ell, u) \le \widehat{a}^\top s + b + \delta^\top s = a^\top s + b \le F(s).$$

□

The lemma turns coefficient-storage error into an adjustment to the constant term that is valid across the whole box. It assumes a valid original affine bound and a correction meeting the displayed inequality.

Evaluating and storing the affine bound. The implementation constructs the affine bound in 64-bit floating-point arithmetic (FP64). For the chord in (28), it evaluates the slope for an interval of positive width as $E_j^+[-\operatorname{expm1}(-\delta_j)]/\delta_j$, avoiding subtraction of nearly equal exponentials.

To store the bound, it first converts the score coefficients to the verifier's numerical format and computes the box correction. It then adjusts the constant, subtracts the configured 10^{-9} margin, converts it to the storage format, and uses `nextafter` to move to the next representable number toward $-\infty$. The upper-bound path reverses signs and moves its final constant toward $+\infty$.

Inputs are checked for finite values and ordered interval endpoints before handling special cases. If intermediate quantities are NaN or infinite, or a required positive sum rounds to zero, the verifier uses the baseline bound.

Rounding can also occur in the log-sum-exp calculations, elementary functions, sigmoid intercepts, and sum used for the correction. FP64 arithmetic and a fixed safety margin do not bound all these errors. The following checks test the stored result against an independent high-precision reconstruction.

Checking the implemented bound. We check whether the stored affine bound stays below an independent high-precision reconstruction throughout the score box. All 360 cases pass. Table 7 also reports agreement between the direct and scanned constructions and stability of the reference calculation.

The reference uses the direct quadratic construction at 120 and 160 decimal digits, independently of the production candidate-selection routine. The test set covers nine row lengths, $K \in \{1, 2, 3, 5, 8, 17, 32, 64, 128\}$, and both 32-bit (FP32) and 64-bit (FP64) storage. Each combination has four cases in each of five regimes: ordinary score boxes, zero and sub-10^{-12} interval widths, tied coefficients, a common constant added to all scores, and scores spanning a wide range.

For each case, we minimize the difference between the reference and stored affine bounds over the whole score box. As in the rounding lemma, choosing an endpoint for each score gives this minimum. The comparison therefore covers the entire box rather than only sampled score values.

Table 7: Local numerical validation over 360 test cases using 120- and 160-digit reference reconstructions.

Check	Cases	Failures	Recorded quantity or status
Reference-versus-stored box dominance	360	0	Minimum margin 9.9998×10^{-10}
Direct–scan agreement	360	0	Maximum FP64 difference 3.55×10^{-15}
120/160-digit reconstruction	360	0	Maximum difference 3.34×10^{-108}
Candidate-selection stability	360	0	No selected-candidate changes

What the numerical checks establish. These checks provide numerical evidence for the local construction; they do not establish validity for every floating-point execution of the complete verifier. Soundness benchmarks and formal verification address this broader implementation question (Szász et al., 2025; Zhou et al., 2025; George et al., 2026).

D Experimental protocols and additional results

The additional results connect the method's local construction to its use in a verifier. The one-block gain appears across training seeds; comparisons before branch-and-bound search show stronger completed bounds; and selected-property controls and repeats examine the effect of using those bounds. The exact-scalar study tests retained score dependence. We then examine construction cost, language-model attention rows, and the limits encountered on more demanding architectures. Each study retains its own population and comparison protocol. All experiments reported in this paper use NVIDIA RTX 5090 GPUs.

D.1 Matched one-block study

The one-block comparison gives 54 additional Threshold-Mass certificates with no CROWN-LSE-only certificates, and the gain appears at every training seed. Searching for the reference-optimal sigmoid slope gives no additional certificates. We describe the shared training and verification setup before reporting the seed-level evidence and the slope comparison.

The model contains one attention–residual–ReLU–MLP block without LayerNorm, with embedding dimension 32, four heads, MLP width 64, and 7×7 patches. Five training seeds each use 12,000 Fashion-MNIST training examples and 2,000 evaluation examples for eight epochs. Verification

uses the first 100 evaluation examples per seed at ℓ_∞ radius 0.02. Of the 500 attempts, 358 are clean-correct and 142 are clean-incorrect. The 142 incorrectly classified examples remain in the denominator and are not counted as certificates.

Within a seed, the methods use identical in-memory model weights. The order in which methods are evaluated is rotated across runs. All five seeds use RTX 5090 GPUs; runtime ratios are paired within seed. The CROWN-LSE reference uses the coordinate LSE relaxation (Wei et al., 2023). No fallback is reported for the evaluated Threshold-Mass calls in this study.

All 358 clean-correct lower bounds improve. The five seed-level rate differences are 10, 2, 9, 11, and 22 percentage points. Their mean is 10.8 points; the reported seed-level t interval is $[1.9, 19.7]$ points, and the two-sided sign-test value is 0.0625. These summarize five independently trained models. The examples nested within each model are not separate training replications.

The reference-optimal slope variant gives the same certificate decisions as the six-candidate rule, with a slightly weaker mean propagated bound and higher runtime (Table 1). Optimizing the scalar reference value in Appendix A.8 therefore gives no certification advantage in this study.

D.2 Benchmark protocol and comparison scope

Threshold-Mass certifies 143 of 200 properties, compared with 127 for its shadow, retaining every shadow certificate and adding eight on each benchmark model. It also uses less budget-capped time and explores fewer input subdomains. The timing measure, called restricted time, is defined below. We describe the shared protocol and rules for choosing the bound used by the verifier, then report certification outcomes, their pairing, and the search measurements.

The VNN-COMP vision-transformer study uses an ℓ_∞ perturbation radius of $1/255$ and a 100-second budget per property. Table 8 expands the comparison in Table 2, retaining timeout and out-of-memory outcomes in every denominator.

The benchmark contains two models, each contributing 100 properties: a two-block, five-token PGD-trained model and a three-block, 17-token IBP-trained model. The native CROWN-LSE baseline performs no additional pass. The second-LSE and Threshold-Mass shadows compute an additional bound but return the original CROWN-LSE bound. Threshold-Mass-only returns its candidate, while selected Threshold-Mass returns the larger of the candidate and baseline bounds. All additional passes freeze the optimized CROWN-LSE relaxation parameters.

Property–method pairs use fresh processes, fixed property and branching settings, and method orders rotated across ten batches of 20 properties. The shared execution environment uses PyTorch 2.4.0, CUDA 12.1, and RTX 5090 GPUs.

Certification outcomes. Table 8 gives the full outcomes. Threshold-Mass gains eight certificates over its shadow on each model: 69 versus 61 for the PGD-trained model and 74 versus 66 for the IBP-trained model. Equal aggregate counts across methods do not establish identical per-property outcomes.

Table 8: Complete-verifier outcomes on 200 properties with a 100-second budget per property. All timeouts and out-of-memory outcomes remain certification failures.

Arm	Certified / 200	Rate	Timeout	OOM
CROWN-LSE	113	56.5%	55	32
Second-LSE shadow	127	63.5%	71	2
Threshold-Mass shadow	127	63.5%	73	0
Threshold-Mass only	143	71.5%	57	0
Selected Threshold-Mass	143	71.5%	57	0

The native baseline's 32 out-of-memory failures all occur on the IBP-trained model. Extra-pass controls also change memory outcomes, so these differences do not isolate an intrinsic memory advantage of the Threshold-Mass rule. The diagnostics below compare bounds before branching and test the effect of using the candidate Threshold-Mass bound.

Which properties are gained? Selected Threshold-Mass retains every certificate of its shadow and adds 16. It also retains all native CROWN-LSE certificates and adds 30. Both comparisons use 200 properties, with 57 certified by neither method. Appendix D.5 reports the separate exact-scalar comparison.

Time and search effort. Selected Threshold-Mass lowers mean restricted time to certificate from 51.8 to 47.6 seconds relative to its shadow. For each property, restricted time is the measured certification time capped at 100 seconds if certified, and 100 seconds otherwise. Averaging over all 200 properties therefore retains unsuccessful attempts at the full budget rather than rewarding fast failures.

The selected arm explores 355,042 domains versus 402,230 for its shadow, consistent with stronger bounds reducing the search needed. These measurements describe the complete integration under its time budget; Appendix D.6 measures local bound-construction cost separately.

D.3 Completed bounds and the effect of using them

Threshold-Mass strengthens every minimum margin bound in the comparison before branching. A separate selected-property diagnostic then finds additional certificates when the candidate bound is used, while computing and discarding the extra bound leaves certificate outcomes unchanged on this diagnostic set. We first measure bound strength at a matched optimization stage, then compare bound-selection rules on a smaller diagnostic set.

Bounds before branching. The comparison before branching covers the 161 benchmark properties that reach the complete-bound calculation; the other 39 are certified earlier. For each covered property, the Threshold-Mass candidate is compared with optimized CROWN-LSE from the immediately preceding pass at the same root. The decisive margin is the smallest completed bound across that property's output constraints. Comparing these minima after both passes avoids mixing an initial, unoptimized baseline with an optimized candidate.

The gain is positive on all 161 decisive margins, with a mean of 0.1347 and a 95% interval of $[0.1216, 0.1505]$. The interval is obtained by bootstrap resampling of properties separately within each model. These are comparisons of completed bounds before selection, not certificate counts. Improving the minimum margin does not require every individual constraint bound to improve.

Computing an additional bound versus using it. On 12 PGD-model properties fixed before the comparison, using the candidate Threshold-Mass bound increases the certificate count from seven to nine. Both the six-candidate rule and its reference-optimal variant give this result; a second LSE pass adds no certificates. Each shadow/selected pair produces identical vectors of baseline and candidate bounds before branching and differs only in which bound is returned. Every shadow reproduces the native CROWN-LSE outcomes on this set. Runs use fresh processes, RTX 5090 GPUs, and a 100-second budget per property. This selected-set diagnostic tests the effect of using the bound; it does not explain memory differences on the IBP model.

D.4 Repeatability on selected properties

Threshold-Mass retains a certificate advantage across three repeats on each selected comparison set. We describe how these sets are chosen before reporting the differences in Table 9.

The repeat study selects properties that one method certifies and the other does not, or whose runtimes are within ten seconds of the timeout, then repeats each property three times. The CROWN-LSE and Threshold-Mass-shadow comparisons use 90 properties; the exact-scalar comparison uses a separate 92-property set. These selected-subset measurements are separate from the 200-property totals in Tables 2 and 3; they measure repeatability on their own sets and are not pooled with the full benchmark.

The advantage is unchanged across repeats against the Threshold-Mass shadow and exact scalar, and varies by one certificate against matched CROWN-LSE. These results establish repeatability on the selected sets.

Table 9: Additional Threshold-Mass certificates on selected repeat sets. Differences are reported separately for each repeat.

Reference	Properties	Repeat 1	Repeat 2	Repeat 3
Threshold-Mass shadow	90	+8	+8	+8
Exact scalar	92	+30	+30	+30
Matched CROWN-LSE	90	+27	+28	+27

D.5 Exact scalar comparison

Using the affine bound certifies 33 more properties than the exact-scalar control and strengthens the completed root bound on 155 of the 161 compared properties. We first describe how the comparison replaces the local affine bound with a constant, then report certification outcomes and completed root bounds.

For a fixed score box and backward direction, Appendix A.6 computes the exact scalar minimum $V_{\mathcal{B}}(c)$. The control returns zero score coefficients and this minimum as the constant. Other graph terms continue through the host verifier. Both arms first optimize CROWN-LSE, hold its relaxation parameters fixed for an additional pass without further gradient optimization, and select the larger completed bound against their own baseline.

This post-hoc comparison uses the same 200 benchmark properties, 100-second budget, solver seeds, branching settings, fresh-process rule, and rotated method order across batches of properties. It is separate from the shadow comparison in Table 2.

The scalar's first pass producing completed bounds is checked by an independent FP64 calculation using logarithms; later calls omit that diagnostic. This check does not ensure that every rounding operation preserves the required bound inequality. Both methods use the same schedule for choosing bounds, with different local calculations.

Certification outcomes. Threshold-Mass retains every exact-scalar certificate and adds 33, as reported in Table 3.

Completed root bounds. On the 161 properties reaching the complete-bound calculation, Threshold-Mass improves 155 decisive root bounds relative to the exact-scalar control, ties six, and worsens none. The remaining 39 properties are certified earlier and remain in the 200-property certification denominator.

A stronger root bound need not certify the property: it may remain below the required threshold. The results support retaining score dependence in this integration, without implying that every affine bound outperforms replacing the objective by a constant. The scalar is the strongest constant on its score box; the observed advantage concerns subsequent propagation and selection.

D.6 Direct construction and scan timing

The cumulative-sum scan is faster than the direct construction at the tested row lengths above two scores, and its coefficients and bias closely agree with the direct reference. The smallest row length shows a slight slowdown. We give the timing conditions and representative measurements below, then relate the observed runtime to the scan's arithmetic cost.

The GPU timing experiment uses one RTX 5090, 32 rows processed together, FP32 stored inputs, FP64 intermediate calculations, one host thread, five warm-up calls, and nine synchronized repetitions. Times are medians per complete 32-row call. The direct implementation explicitly forms the threshold contributions and serves as a mathematical reference. It is not an optimized competing verifier.

Across the widths where both implementations run, the maximum absolute slope and bias differences are 2.50×10^{-16} and 3.78×10^{-15}. The scan's nearly flat time over this measured range reflects the overhead of launching GPU work and processing rows together; its arithmetic cost still grows with row length. The $K = 2048$ result establishes scan execution and memory use at that width, with no direct-path comparison.

Table 10: Representative widths from the local scaling study. Peak memory is the scan's incremental peak allocation. The direct path was not run at $K = 2048$.

Scores K	Direct (ms)	Scan (ms)	Direct / scan	Peak (MiB)
2	3.496	3.628	0.96×	0.02
17	41.731	3.683	11.33×	0.21
64	161.158	3.693	43.64×	0.82
128	324.274	3.695	87.76×	1.63
512	1302.240	3.726	349.47×	6.54
1024	2606.764	3.721	700.59×	13.09
2048	Not run	3.749	N/A	27.19

D.7 Local Qwen attention trace

In this fixed, recorded attention trace, Threshold-Mass improves bounds in 409 of the 448 local cases with nonzero coefficient vectors at tolerance 10^{-6}. The variant comparisons are mixed: improving more rows does not necessarily give the largest mean gain. We first define the trace, objective, and score boxes, then report the comparisons and their scope.

Trace, objective, and score boxes. The trace uses Qwen2.5-3B-Instruct with a fixed 128-token prompt and BF16 eager attention. It samples layers 0, 11, 23, and 35, all 16 heads, and causal query positions 63 and 127. This gives 128 attention rows, with row lengths of 64 and 128. Each row is evaluated at four perturbation radii, giving 512 cases.

At the final token position, we form the difference between the highest and second-highest logits. We differentiate this margin with respect to the probabilities in the sampled attention row. The resulting gradient supplies the local coefficient vector, which is held fixed for each comparison.

The local perturbation boxes are constructed around normalized query and key hidden vectors at radii 10^{-5}, 3×10^{-5}, 10^{-4}, and 3×10^{-4}. Bounds through the query and key projections provide radii r_q, r_k. For the stored centers q_0, k_{0j} after the positional rotation, the score uncertainty is bounded using

$$\left|(q_0 + \delta q)^\top (k_{0j} + \delta k_j) - q_0^\top k_{0j}\right| \le \|q_0\|_2 \|r_k\|_2 + \|k_{0j}\|_2 \|r_q\|_2 + \|r_q\|_2 \|r_k\|_2. \tag{37}$$

The bound follows by expanding the inner product and applying Cauchy–Schwarz to each varying term. Division by the square root of the head dimension and the numerical safety margin give the score radius. Each score center is recomputed as the FP64 dot product of the retained query and key vectors, so the center and perturbation bound use the same vectors.

Comparison with CROWN-LSE. For each case, we minimize each method's affine bound over the same independent score box and compare the resulting scalar lower bounds. The gain is the Threshold-Mass lower bound minus the CROWN-LSE lower bound.

There are 512 cases, of which 64 have exactly zero coefficient vectors because of the causal graph. On the 448 cases with nonzero coefficient vectors, an absolute comparison tolerance of 10^{-6} gives 409 improvements, 38 ties, and one regression relative to CROWN-LSE. At 10^{-9} the corresponding counts are 445 improvements, no ties, and three regressions. These two summaries use different tolerances on the same trace.

Slope-selection variants. We compare the default six-candidate rule with two simpler choices. The derivative-only rule uses the sigmoid derivative at the reference value as its single slope. The constant-group rule sets every sigmoid slope to zero; bounding each group separately this way does not generally give the exact minimum of the full weighted expression.

Table 11 compares these rules with CROWN-LSE at tolerance 10^{-9}. All rules retain every threshold. The denominator is all 512 cases, including the 64 zero coefficient vectors. The default improves the largest fraction of cases and has the largest median gain.

Table 11: Slope-selection variants on all 512 trace cases, including zero coefficient vectors. "Better" means a lower-bound gain over CROWN-LSE greater than 10^{-9}. Median gains use the same cases. All rules retain every threshold.

Policy	Better (%)	Median gain
Six candidates	86.9	6.52×10^{-5}
Derivative only	81.4	2.97×10^{-5}
Constant group bounds	34.6	-1.17×10^{-5}

Selecting the larger of each candidate bound and CROWN-LSE changes the comparison: the constant-group rule has a slightly larger mean selected gain than the default because of a few wide-box cases. Thus improvement frequency and mean gain after selection favor different rules on this trace.

Interpretation. Choosing the larger bound preserves or improves the CROWN-LSE result. Minimizing either affine bound over the independent score box does not exceed the exact scalar minimum. Together, these comparisons assess local bound quality on the frozen trace. Uncertain values, other layers, and complete input perturbation sets are not propagated through the language model, so the results do not provide certificates for the complete model.

D.8 Additional blocks, LayerNorm, and longer rows

Stronger finite margins yield only two Threshold-Mass certificates with two blocks and none with pre-LayerNorm. At 49 tokens, memory failures prevent bounds on clean-correct examples. We describe the shared evaluation setup, then distinguish bound quality from certification and memory outcomes.

The five-seed study compares Threshold-Mass with native CROWN-LSE and independent implementations of Wei ER, LSE, and LSE*. It retains the Fashion-MNIST data sizes, eight training epochs, embedding dimension 32, four heads, and MLP width 64 from Appendix D.1. Seeds 0–4 each verify the first 100 evaluation examples at ℓ_∞ radius 0.02, giving 500 attempts per method and setting, with failures retained in the denominator. Methods share the trained weights within each seed and setting.

Each setting changes one architectural feature of the one-block model. The two-block setting uses two blocks without LayerNorm. The pre-LayerNorm setting keeps one block and adds pre-LayerNorm. Both retain 7×7 patches. The long-row setting keeps one block without LayerNorm and uses 4×4 patches, giving 49 tokens.

Table 12: Certificate counts for more demanding architectures, out of 500 attempts per setting and method. The 49-token setting is dominated by memory exhaustion.

Setting	Threshold-Mass	Native LSE	Wei ER	Wei LSE	Wei LSE*
Two blocks	2	0	0	0	0
Pre-LayerNorm	0	0	0	0	0
49 tokens	0	0	6	9	10

Bound quality and certification. The bound comparison includes clean-correct examples with finite bounds from Threshold-Mass and all three Wei references: 371 two-block examples and 386 pre-LayerNorm examples. Threshold-Mass improves the completed minimum-margin bound over the strongest Wei reference on every included example. These improvements rarely cross the threshold needed for a certificate. Native LSE records six nonfinite-bound errors in the two-block setting and 49 assertion failures in one LayerNorm seed.

Memory failures and execution conditions. The full study uses shared processes. Threshold-Mass and native CROWN-LSE each exhaust memory on all 335 clean-correct long-row examples; the three Wei implementations also have memory failures. These outcomes depend partly on execution conditions and do not isolate the mathematical quality of the bounds.