Exact Integer Execution of a Small Transformer: Specification, Conformance, and Number Formats

Flyxion
Independent Researcher

November 2026

Abstract

Most work on integer inference for transformers asks how much speed or memory can be recovered at an acceptable loss of accuracy. This essay asks a different question of the same arithmetic: whether the computation of a pretrained language model can be written as a specification so complete that independent implementations on different hardware must agree on every intermediate integer. The specification developed here covers HuggingFaceTB/SmolLM2-135M, a 30-layer decoder with 135 million parameters. Every stored value is an integer with a declared format, every rounding point and saturation is named, every nonlinear function is a table or an exact integer procedure, and every accumulation in a matrix product or reduction is carried out in 64-bit integers, with the single exception of the normalization total, which needs a wider integer, so that no reduction order can change a result. A numpy prototype defines the semantics, and a scalar C program and a CUDA program reproduce every recorded checkpoint exactly on the tested inputs: the embedding output, the residual stream after each of the thirty blocks, and every logit, for three test sets of 24, 128 and 160 tokens. The number formats were chosen from a measured activation profile, and that profile produced the most consequential finding: a single sink token per window carries a value near 20,500 in one residual channel for layers 12 through 28, against a median magnitude between 1.6 and 10.9, which explains why a single 16-bit residual scale is inadequate. Narrowing the residual stream and the flagged sites together to 16-bit per-tensor grids costs about 0.12 nats per token against the float baseline, nearly twenty times the entire cost of quantizing the weights to eight bits. A three-way ablation on the same eight windows attributes about 82 percent of that excess over the eight-bit-weight reference to the residual stream, 11 percent to the flagged sites (all of it in the gated product of the feed-forward layer), and the remainder to their interaction, and giving channel 507 alone its own exponent removes 99 percent of the residual's share, while one exponent per channel, evaluated as a storage grid, removes essentially all of it. With a 32-bit residual and 32-bit storage at flagged sites, the integer engine scores 3.1585 nats per token on eight validation windows against 3.1587 for the same eight-bit weights evaluated in floating point, so the integer arithmetic itself contributes almost nothing beyond the weight representation. The essay sets out the concepts, the evidence, and the limits of what agreement between implementations establishes.

1 Introduction

A trained transformer is usually described as a function on real numbers, and it is executed as a sequence of floating point operations whose results depend on the order in which sums are accumulated, on the width of fused multiply-add units, on the tiling chosen by a library for a given matrix shape, and on how many other requests happen to share a batch. Floating point addition is not associative [4], so two correct implementations of the same mathematical function routinely disagree in the low bits, and those low bits propagate through thirty layers and a softmax to a different probability distribution. Recent analysis of production serving argues that the dominant source of run-to-run variation is not concurrency as such but the absence of batch invariance in the kernels for normalization, matrix multiplication and attention, so that the output for one request depends on the unrelated requests it is batched with [5]. The usual remedy is to constrain a floating point implementation until it behaves, at some cost in throughput.

An alternative is to remove the source of disagreement and leave the floating point arithmetic behind. Integer addition and multiplication are exact so long as no intermediate value overflows, and exact addition is associative, which means that the order of a reduction cannot affect its result. If every place where information is discarded is named and given a definition, then the whole forward pass becomes a function from token identifiers to integers that any machine can compute and any observer can check. The program of this essay is to carry that out for a real pretrained model, to measure what it costs in model quality, and to record honestly what the resulting agreement does and does not demonstrate.

The distinction between conformance and quality organizes the whole study. Conformance is judged against the integer reference alone: an implementation conforms if it reproduces the reference integers exactly. Quality is judged against the original floating point model: the reference is good if the language model it computes is close to the one that was trained. The two questions are answered by different evidence and should not be confused. Exact agreement among three programs written from one definition provides evidence of implementability and checkpoint consistency on the tested inputs, and it shows nothing about whether the definition was a wise choice. The quality measurements bear on that second question, and the profiling work that precedes them supplies the reasons for the particular formats chosen.

The artifacts described here are published under a public domain dedication at `https://github.com/standardgalactic/exact-integer-inference`, together with the pinned inputs, the measured profile, the test vectors' hashes, and a dated record of each conformance run.

2 Related Work

Integer-only inference has a substantial history. Jacob and colleagues described a quantization scheme for convolutional networks in which inference is carried out with integer arithmetic alone, together with a training procedure that simulates quantization so that accuracy survives, and they demonstrated it on mobile CPUs [6]. Extending the idea to transformers is harder because softmax, the activation function and layer normalization are not piecewise linear. I-BERT addressed this for BERT and RoBERTa by approximating GELU and softmax with low-order

integer polynomials and by computing the normalization with an integer square root algorithm, reporting accuracy comparable to the full precision baseline on GLUE tasks and a speedup of 2.4 to 4.0 times for INT8 inference on a T4 GPU in a preliminary implementation [7]. These works pursue efficiency. Their approximations are tuned to preserve task accuracy, and they are not intended to pin down every intermediate value, so that two independent implementations of the same scheme would agree on accuracy without necessarily agreeing on integers. The present work uses tables for the exponential and the SiLU function and an exact integer square root for normalization, and it is concerned with the definition of each rounding step more than with the choice of approximation.

Quantization of large language models has been dominated by the difficulty of outliers. Dettmers and colleagues showed that systematic outlier feature dimensions emerge in transformer hidden states, that they dominate attention and predictive performance, and that standard eight-bit matrix multiplication fails without special treatment, and they proposed vector-wise quantization together with a mixed-precision decomposition that isolates the outlier dimensions into a 16-bit multiplication while more than 99.9 percent of values are multiplied in eight bits [3]. SmoothQuant takes the complementary route of migrating the quantization difficulty from activations to weights through a mathematically equivalent per-channel scaling, enabling eight-bit weights and activations for the matrix multiplications of large models [12]. A related line of work concerns the magnitude of particular activations rather than the existence of outlier channels. Sun and colleagues documented massive activations, a very small number of activations that can be 100,000 times larger than others, located in fixed feature dimensions and attached to the starting token and to the first delimiter token, which function as indispensable bias terms and concentrate attention probability on their tokens [11]. Earlier, Xiao and colleagues described the attention sink, the tendency of models to direct strong attention to initial tokens regardless of their semantic content [13]. The measurements in this essay show that these phenomena reach down to a model of 135 million parameters and that, for an integer specification, they are the principal obstacle to narrow storage of the residual stream.

3 The Model and the Inputs

The subject is the base checkpoint of SmolLM2-135M, a member of the family documented by Allal and colleagues [2], pinned at revision 93efa2f097d58c2a74874c7e644dbc9b0cee75a2. It is a decoder-only transformer with 30 layers, a hidden width of 576, nine query heads and three key-value heads sharing keys and values in groups of three [1], a head dimension of 64, a gated feed-forward layer of width 1,536 with a SiLU gate [9], a vocabulary of 49,152 tokens, and input and output embeddings that share a single matrix. Normalization is RMSNorm [14] with an epsilon of 10^{-5}, and positions are encoded by rotary embeddings [10] in the non-interleaved rotate-half arrangement with a base of 100,000, not the more common 10,000. Both of those configuration values are easy to assume wrongly and both matter for an integer specification, since the rotation tables and the normalization constant derive from them. The checkpoint stores its tensors in bf16, so widening to float32 is exact and the floating point baseline carries exactly the checkpointed values. The weight file is 269,060,552 bytes with SHA-256 digest

80521b40281d6ce74e35c9282c22539e75aa0ac8578892b2a59955ef78d55da1, and every tensor, the configuration and the tokenizer assets are hashed in a manifest.

The text used for calibration and validation is the test split of the raw WikiText-2 corpus [8], at a pinned dataset revision with a pinned file hash. Rows are joined with blank lines and tokenized once with no beginning-of-sequence token, giving 304,978 tokens, which are cut into 595 non-overlapping windows of 512 tokens, with a 338-token tail discarded. Windows 0 through 63 serve for calibration, which means that they choose scales, and windows 64 through 127 serve for validation, so that no clipping rate reported below was measured on data that chose a scale. Each window is processed as an independent sequence at batch size one. The held-out status is limited to this procedure: the text is held out from the calibration, and nothing guarantees that it is absent from the pretraining mixture.

The floating point reference is a forward pass written by hand so that every site can be observed. On three windows it matches the Hugging Face eager implementation to a maximum logit difference of 8.0×10^{-5}, with identical argmax at every position and a negative log-likelihood equal to about seven digits. Its perplexity on the scored tokens is 20.22 on the calibration windows and 20.89 on the validation windows, each over 32,704 tokens. The quality comparisons in Section 6 use the first eight validation windows, which contain 4,088 scored tokens, and the figures there are therefore not directly comparable to these perplexities, which cover all 64 windows.

4 What a Specification Must Fix

An executable specification of a transformer in integers has to settle, for every operation, what information is kept and how it is discarded. The semantics developed for this model fix the following. A stored tensor is a pair of an integer array q and an integer exponent f, the represented value being $q \cdot 2^{-f}$, where f may be negative. The rounding right shift is

$$\mathrm{RS}(x, n) = \begin{cases} \left\lfloor \dfrac{x + 2^{n-1}}{2^n} \right\rfloor, & n > 0, \\ x, & n = 0, \\ x\, 2^{-n}, & n < 0, \end{cases}$$

so that ties go toward positive infinity and a negative n denotes an exact left shift. This is the mathematical definition, and it should be kept apart from its implementation. The C and CUDA programs compute the positive case as an addition followed by a right shift of a signed 64-bit value, which is an arithmetic shift on the compilers used but is implementation-defined for negative operands in C before C23, and they compute the case $n < 0$ as a multiplication by a power of two, because a left shift of a negative signed value is not portably defined. Mathematical exactness does not by itself establish that the corresponding language operation is defined, and the agreement reported below is agreement on specific compilers. Saturation clips to the signed range of the destination width. All tables are rounded half up from 50-digit values computed with arbitrary-precision arithmetic, so that the table entries themselves are defined by mathematics and not by a library's floating point routines.

Information is discarded only at named points: the quantization of weights, the requantization after each linear layer, the rounding of the rotary rotation, the requantization of attention scores, the division that produces attention probabilities, the SiLU table, the rounding of the gated product, and the rounding of the normalization output. Between these points the arithmetic is exact. Matrix products accumulate in 64-bit integers with operands widened before multiplication. The prototype computes these sums through a float64 matrix routine, which is exact under a sufficient condition on the operands and not merely on the finished sum, since cancellation can leave a small result after large partial sums. The assertion on each call checks that $K \cdot \max|a| \cdot \max|w| < 2^{53}$, where K is the inner dimension and the operand bound $\max|a|$ is supplied at the call site. That product bounds $\sum_i |a_i b_i|$, and therefore every product and every partial sum under any reduction order. The check is conservative and rests on the operand bound being correct. There is no fallback path: a violation halts the run, and none occurred in the reported runs, so every matrix product of the prototype in those runs went through the float64 routine, and the attention products are guarded by an assertion of the same kind. All of this is a device of the prototype, and the definition is the integer arithmetic. Because integer addition is associative, no ordering of a reduction, whether sequential, tiled, split across threads or batched with other sequences, can change a value that does not overflow, and this is the property on which agreement between a CPU loop and a GPU kernel rests. Overflow itself is excluded by bounds argued from the formats and, where cheap, asserted.

The residual stream is a 32-bit integer with 15 fractional bits at every layer, and the embedding lookup writes directly into it. Linear weights are eight-bit integers with one scale per output row, stored as a multiplier $m \in [2^{13}, 2^{14}]$ with 14 bits of precision and a shift n (the interval includes its upper endpoint, so an unsigned encoding of m would need 15 bits, and the reference holds it in a 64-bit integer), so that each requantization takes the form $\mathrm{RS}(\mathrm{acc} \cdot m,\ n + f_{\mathrm{in}} - f_{\mathrm{out}})$, and the product $\mathrm{acc} \cdot m$ stays below 2^{63} for every layer shape in this model. The tied embedding matrix is stored once as an eight-bit matrix with one such scale per vocabulary row, and it serves both for the lookup and for the output head. Activation formats follow the scaling proposal derived in Section 5: 16-bit with $f = 15 - k$ at ordinary sites and 32-bit with $f = 31 - k$ at the 118 flagged site-layers out of 876, where k is determined by a calibration maximum.

Normalization deserves separate attention because it is where a naive integer scheme overflows. For a residual vector x of 576 elements held as 32-bit integers, the sum of squares $S = \sum x_i^2$ can reach $576 \cdot 2^{62}$, which exceeds the 64-bit range even though measured values do not come near it. The specification therefore computes S exactly from three 64-bit partial sums over the high and low 16-bit halves of each element, adds a constant c equal to the epsilon expressed in squared-residual units, which is $\mathrm{round}(576 \cdot 10^{-5} \cdot 2^{30}) = 6{,}184{,}753$, and defines the reciprocal root as $U = \mathrm{isqrt}\left(\lfloor 2^{80}/(S+c) \rfloor\right)$. Since $|x_i| \le \sqrt{S} \le \sqrt{S+c}$, every product $x_i U$ is bounded in magnitude by 2^{40} for every possible input. That bound covers the product and its rounding to 16 fewer bits, which gives a rescaled value z with $|z| \le 2^{24}$. The next step, the multiplication of $3z$ by a 16-bit gain, is bounded separately by $3 \cdot 2^{24} \cdot 2^{15} < 2^{41}$, which also fits in 64 bits, and the final rounding and saturation into the output format are defined by the specification and cannot overflow. This is a bound on all inputs, not a measured range, and it is the one place where the specification supplies a proof in place of an observation.

The normalized value is then rounded to 24 fractional bits, multiplied by the 16-bit gain, and rounded into the output format.

Attention is specified to avoid every ambiguity of the usual softmax. Scores are 16-bit with seven fractional bits, obtained from the query-key product by a pure power-of-two shift with the factor 1/8 included. The exponential is a table indexed by the distance from the row maximum in steps of 1/128, holding $\exp(-d/128)$ as a 30-bit fraction, with entries beyond a distance of 12 defined to contribute exactly zero. Probabilities are integers with 16 fractional bits formed by a rounded integer division of each exponential by the row sum, with 65,536 meaning exactly one. Because 65,536 does not fit in an unsigned 16-bit integer, the reference programs hold probabilities in 64-bit integers and do not saturate, so exact one is representable and the denominator is 2^{16}. A narrower deployed format would have to choose among a wider type, a denominator of 65,535, or saturation at 65,535, and those choices yield different integers. Masked positions are skipped structurally, so that no sentinel value such as negative infinity belongs to the semantics. Rotary tables are Q30 and the rotation is rounded once. The SiLU function is a table of 65,536 entries per distinct pair of input and output formats, indexed by the 16-bit gate value.

5 Number Formats from a Measured Profile

Formats were chosen from an activation profile gathered by the floating point reference on the 64 calibration windows and checked on the 64 validation windows. At every layer the profiler records the residual stream at the block input, after attention and after the feed-forward layer, both normalization stages, the query, key and value projections before and after rotation, attention scores and their shifted forms, row maxima, exponential sums and probabilities, the attention context and output projection, and the gate, up, SiLU, gated-product and down-projection sites of the feed-forward layer, together with the final normalization and the logits. Each site keeps exact signed extrema, per-channel absolute maxima and sums of squares, a per-token absolute-maximum histogram, and a histogram of absolute values with 32 linear sub-bins per octave. Because octave boundaries are exact powers of two, exceedance counts at power-of-two thresholds are exact, and percentiles are interpolated within bins at most about three percent wide. Masked attention entries, which make up 49.9 percent of every score matrix, are excluded from every statistic. The profile ran on one thread with deterministic algorithms, and a two-window repeat reproduced its content digest bit for bit, although the full 64-window profiles were each run once.

The residual stream is the one site where a 16-bit format with a single scale fails, and the reason is a single token. From layer 12 through layer 28, one token per window carries a value near 20,500 in residual channel 507, while the median absolute residual value lies between 1.6 and 10.9 across those layers and the next largest channels reach only about 1,400 and 670. In all 64 calibration windows the first position is such a token, and three further cases occur at positions 1 and 2 of windows that begin with repeated digit tokens. The value is written by the feed-forward layer of layer 11, where the gated product reaches 2,519 in channel 1229 and the down projection writes 20,517 into channel 507, and it is mostly removed by

the feed-forward layer of layer 28, where the down projection contributes 15,975 to the same channel, leaving 4,348 at the input of layer 29. This is the massive activation described by Sun and colleagues [11], observed here in a model much smaller than most of those they studied, and its attention-side signature is also present: the probability placed on key position zero has a median of 0.37 at layer 12 and 0.67 at layer 20, consistent with the sink behavior described by Xiao and colleagues [13]. Reports that systematic outliers emerge only beyond several billion parameters [12] describe outlier channels in the bulk of the tokens, and the present observation is of a different kind, a single-token effect, but for the purpose of choosing integer formats the model's small size does not exempt it.

The consequence for a 16-bit residual is arithmetic. With one spare bit above the calibration maximum, a per-tensor scale needs a step of 2.0 at every layer from 12 to 28, so the median residual value spans about 0.8 steps at layer 12, 1.6 at layer 20 and 5.4 at layer 28. Removing channel 507 from the range calculation only improves the median to 13 steps at layer 12. Two alternatives were measured on the same data. A 32-bit residual with 15 fractional bits covers every layer with a spare bit, the largest magnitude being 20,558 against a range of 65,536, and spans at least 2,258 steps at the median everywhere. A 16-bit residual with one power-of-two exponent per channel keeps every channel's root-mean-square value at 346 steps or more. The first was adopted for the reference, and the second was not built as an engine, and its effect on quality is evaluated as a storage grid in Section 6, since the step counts show only that it keeps resolution. The 32-bit choice resembles in spirit the isolation of outliers into higher precision proposed by Dettmers and colleagues [3], applied here to a stored stream and not to a matrix product, and it differs from the rescaling route of SmoothQuant [12], which modifies weights and activations by an equivalent transformation and was not attempted.

Elsewhere the profile is benign. Under the default rule, which stores each site as a 16-bit integer with a static per-tensor exponent $f = 15 - k$ and $k = \lfloor \log_2(\text{calibration maximum}) \rfloor + 2$, the validation windows saturate zero values across all 876 site-layers. The query, key, value, attention context, gate, up, SiLU and normalization outputs keep channel root-mean-square values at 36 steps or more at nearly every layer. Exceptions are narrow: layers 11, 28 and 29 write massive values through the gated product, which starves 1,534 of 1,536 channels at layer 11, and through the down projection at layers 11 and 28. In total 118 of the 876 site-layers are flagged for wider storage, 545 stay at 16 bits with a per-tensor scale, the 30 probability sites are assigned an unsigned 16-bit format in the proposal (held wide in the reference, as noted above), and the normalization and exponential-sum scalars take wide fixed point. The 64-step flag threshold is a screening rule and not a derived bound.

Attention and normalization carry their own findings. Shifted scores reach a magnitude of 86, and between 32 and 79 percent of valid entries beyond layer 0 lie at 8 or less below the row maximum, so the exponential table needs a defined floor, which is why entries beyond a distance of 12 are set to zero. The consequence of that floor was not measured. Each omitted entry carries a weight below $e^{-12} \approx 6.1 \times 10^{-6}$ relative to the largest entry of its row, whose own weight is one, so even if all 511 other positions of a 512-token window were omitted, the discarded share of the row would be at most about 3.1×10^{-3}. That is an analytic worst case and not a measurement of the realized loss. Probabilities of exactly one occur on the first row, so an unsigned 16-bit format would need either saturation or one extra bit, and the reference

avoids the question by holding probabilities in 64-bit integers. At 16 fractional bits up to 53 percent of valid probability entries at some layers round to zero, yet the mean probability mass discarded per row is at most 2.0×10^{-4}. The mean square entering normalization spans 27 octaves, from 0.0055 to 741,000, and its reciprocal root spans 13.5 octaves, which is why a table directly over the input would be inadequate and why an exact procedure is used.

The tied embedding matrix plays two roles with different sensitivities, and the profile showed that a single scale for the whole matrix serves them unequally. Quantizing the matrix to eight bits with all else held in floating point, over 16 validation windows of 8,176 tokens, one per-tensor scale gives a Kullback–Leibler divergence from the floating point model of 8.0×10^{-4} with 98.8 percent top-1 agreement in the lookup role, but 1.6×10^{-2} with 90.8 percent in the output head role. One scale per vocabulary row reduces these to 6.4×10^{-5} with 99.6 percent and 8.5×10^{-4} with 98.0 percent, and using the same per-row matrix in both roles gives 9.1×10^{-4} with 98.0 percent. A per-row scale is a single vector of 49,152 values attached to one stored matrix, so the tied weight remains one representation. For the final normalized hidden state held in 16 bits, between 5 and 9 fractional bits are all benign, 10 begins to clip, and 11 or more breaks the model, with top-1 agreement falling to 82 percent at 11 bits and 31 percent at 13, which matches a final maximum of 57.1 and the eight fractional bits chosen by the one-spare-bit rule. Differences in negative log-likelihood in this study were small and several changed sign, so divergence and top-1 agreement are the dependable indicators at this sample size.

6 Quality

The integer engine was run in several configurations on eight validation windows, 4,088 scored tokens. In the wide mode the flagged site-layers use 32-bit storage as specified above. In the narrow mode the residual at block boundaries and the flagged sites are forced onto 16-bit per-tensor grids. Table 1 gives the two principal modes, with each divergence tagged by the reference it was measured against, and Table 2 gives the ablation of the narrow mode, in which every variant is compared with the same reference.

Variant	Nats/token	KL (reference)	Top-1 (reference)
Float32 baseline	3.1523	n/a	n/a
INT8 weights, float arithmetic	3.1587	5.9×10^{-3} (F)	97.0% (F)
Wide integer engine	3.1585	2.5×10^{-5} (W)	99.7% (W)
Narrow, all 16-bit	3.2731	0.107 (F)	82% (W)

Table 1: Quality on eight validation windows (4,088 scored tokens). Each divergence is measured against the reference in parentheses: F is the float32 baseline and W is the INT8-weight float model. The two columns of the narrow row use different references, as recorded in the repository, and entries compared against different references should not be read against one another.

The central comparison is between the second and third rows. The model with eight-bit weights evaluated in floating point scores 3.1587, and the wide integer engine, which performs every other operation in integers with the rounding points of Section 4, scores 3.1585. Its

divergence from the weight-only model is 2.5×10^{-5}, which is two orders of magnitude below the divergence the weights alone introduce. The arithmetic discipline of the specification, in other words, costs almost nothing in quality once the formats are chosen to hold the residual stream. What it costs in storage and bandwidth has not been measured, since the implementations are unoptimized.

The narrow variant shows the price of narrow storage at the residual stream and the flagged sites taken together. Forcing both onto 16-bit per-tensor grids raises the score to 3.2731, a loss of about 0.12 nats per token against the float baseline and nearly twenty times the entire loss attributable to weight quantization. This variant is not catastrophic, since it retains most of the model's behavior, but the loss is large compared with every other effect measured. To find where it comes from, the narrow mode was split into its two parts and each part was run alone, with the weights, inputs and all other arithmetic unchanged, and the wide and joint configurations were rerun in the same script as controls, reproducing the earlier negative log-likelihoods exactly.

Variant	Nats/token	Excess over W	KL	Top-1
Wide control	3.1585	−0.0001	2.5×10^{-5}	99.7%
Residual only, per-tensor 16-bit	3.2521	+0.0934	9.2×10^{-2}	84.0%
Flagged sites only (30 site-layers)	3.1715	+0.0129	8.2×10^{-3}	94.8%
gated product only (26)	3.1707	+0.0121	8.2×10^{-3}	95.2%
attention context and SiLU only (4)	3.1585	−0.0001	2.6×10^{-5}	99.8%
Joint (residual and flagged sites)	3.2731	+0.1145	1.0×10^{-1}	82.2%
layers 12 through 28 only	3.2538	+0.0952	9.0×10^{-2}	83.8%
layers 0 through 11 and 29 only	3.1700	+0.0113	9.9×10^{-3}	94.4%
channel 507 on its own exponent	3.1595	+0.0009	4.1×10^{-4}	98.9%
Residual only, per-channel 16-bit	3.1590	+0.0003	7.7×10^{-5}	99.5%
Per-channel residual, flagged sites narrow	3.1708	+0.0121	8.2×10^{-3}	94.8%

Table 2: Ablation of the narrow mode on eight validation windows (4,088 scored tokens). Every KL divergence and top-1 agreement is measured against the same reference, the float model carrying the dequantized INT8 weights (W, 3.1587 nats per token), and the excess is the variant's negative log-likelihood minus that of W. Across windows the KL of the per-tensor residual-only variant ranged from 8.6×10^{-2} to 1.0×10^{-1} and that of the per-channel residual-only variant from 7.2×10^{-5} to 8.2×10^{-5}.

The residual stream accounts for most of the penalty. Narrowing only the residual at block inputs and outputs, with every flagged site wide, raises the negative log-likelihood by 0.0934 nats per token over the weight-only reference, which is 82 percent of the 0.1145 excess of the joint variant. Narrowing only the flagged sites raises it by 0.0129, or 11 percent, and the two parts together exceed the sum of their separate effects by 0.0082 nats, which is the interaction. Within the flagged sites the effect is carried entirely by the gated product of the feed-forward layer, which accounts for 0.0121 of the 0.0129, while narrowing the attention context and SiLU sites changes nothing measurable. A count qualifies what the flagged-sites-only variant narrows. Of the 118 flagged site-layers, 81 are residual-stream sites, which act in this engine only through the block-boundary grids of the residual variant, and 7 are down-projection sites, whose output is written directly into the 32-bit residual and which have no stored width of their own, so the

variant narrows the remaining 30: 26 gated products, 2 attention contexts and 2 SiLU outputs.

Two further variants, indented under the residual-only row of Table 2, locate the loss within the residual stream. Applying the per-tensor grid only at layers 12 through 28, where the sink value is present, gives an excess of 0.0952, as large as the 0.0934 of the all-layer variant, while applying it only at the other 13 layers gives 0.0113. The two figures need not add, because rounding errors from different layers interact, and the all-layer figure is slightly smaller than the sink-layer figure. The sink layers therefore carry the loss, and the remaining layers contribute a smaller penalty of their own, consistent with the earlier finding that a per-tensor 16-bit residual resolves the median value poorly at several early layers. A narrower test keeps the per-tensor grid at all 30 layers but gives channel 507 its own exponent, with the other 575 channels sharing one exponent set by the calibration maximum of the remaining channels. The excess falls from 0.0934 to 0.0009, which removes 99 percent of it, with a KL of 4.1×10^{-4} and 98.9 percent top-1 agreement. A single channel therefore accounts for nearly all of the residual's penalty. The test isolates channel 507 and not the sink token: it uses the calibration maximum of that channel over all positions and does not separate the sink position from the others, though the earlier profile found the large value at one position. The second observation, with a separate exponent for every channel, points the same way: when each channel receives its own power-of-two exponent, chosen by the same one-spare-bit rule from the calibration maximum of that channel, the residual-only penalty falls from 0.0934 to 0.0003 nats per token, which removes 99.7 percent of it, with a KL of 7.7×10^{-5} and 99.5 percent top-1 agreement. Under a per-tensor scale a single channel with a calibration maximum of 20,558 sets the step for the other 575 channels, and a per-channel scale removes exactly that coupling. With the flagged sites also narrow, the per-channel residual leaves only the 0.0121 attributable to the gated product.

The per-channel and channel-507 results are measurements of storage precision and not specifications. The residual was rounded onto the per-channel (or two-group) grid at the same sites and by the same simulation as the per-tensor narrow mode, while the engine kept its 32-bit residual arithmetic and its exact normalization, so the experiment does not define how a residual addition or a normalization would operate across channels that carry different exponents, and it does not count the cost of storing the exponents. The narrow mode also rounds the residual only at block boundaries and not in mid-block, so it understates the penalty of a truly 16-bit residual. The comparison rests on eight windows of one text source without a formal significance test, although the window-to-window ranges in the table caption are small beside the differences between variants.

Single blocks at layers 0, 11, 12 and 28, fed the quantized floating point residual and compared with a float64 block carrying the same dequantized weights, agree to between 10^{-4} and 4×10^{-3} relative root-mean-square error at every checkpoint, including layers 11 and 28 where the sink token is written and removed. The largest per-checkpoint errors are attributed, without a controlled test, to the one-over-128 resolution of attention scores. Rounding a score to a multiple of 1/128 bounds its error by 1/256, which bounds the relative error of an individual exponential factor by $e^{1/256} - 1$, about 0.4 percent, and because probabilities are normalized by a row sum, the relative error of a probability can approach $e^{1/128} - 1$, about 0.78 percent, before the other approximation effects. Eight fractional bits for scores, with a range of plus or

minus 128 against an observed maximum of 78, is a candidate refinement.

7 Conformance

The numpy prototype defines the semantics and produces the expected states. A scalar C program reimplements the whole model with plain integer arithmetic, 64-bit accumulators, a 128-bit integer type for the normalization total and the division by 2^{80}, and an exact digit-by-digit integer square root. A CUDA program reimplements it again with kernels for the embedding, normalization, linear layers, rotation, attention, residual addition and the gated product, in which the division by 2^{80} in the normalization is carried out by a shift-and-subtract loop. Both programs read the same model blob and the same expected-state files and compare, element by element, the embedding output, the residual stream after each of the 30 layers, and the final logits, reporting the first differing index when one exists. These are recorded checkpoints and not every intermediate integer. The specification is written to fix every intermediate state, but the comparison observes the checkpoints, and a discrepancy inside a block that cancelled or was rounded away before the next checkpoint would not be seen.

For the three test sets, window 64 with 24 tokens, window 64 with 128 tokens, and window 100 with 160 tokens, which is the 128-token prompt plus 32 continuation positions recomputed without a cache, the three implementations reproduce every recorded checkpoint exactly, at every layer and in the logits. The C program takes seven to ten seconds per sequence at ordinary optimization. The programs were built and run on Ubuntu 24.04 under WSL2 (kernel 6.18.33.1), with gcc 13.3.0 as the C compiler and nvcc from CUDA toolkit 13.3 (build cuda_13.3.r13.3/compiler.38244171_0) for the GPU program, at driver version 610.88. The CUDA program ran on a GeForce RTX 5060, and the 160-token set completed in 0.31 seconds on the device including the comparisons. The check is shown to be capable of failing: a copy of the 24-token set with one expected residual element changed by one unit is reported as a mismatch at the right layer and index by both programs. The CUDA program was additionally rebuilt without optimization and with a fixed architecture target instead of the native one, and each build agreed exactly on all three sets, which removes the optimization level and the architecture flag as explanations for agreement. Hashes of the model blob and of every expected-state file are recorded, and a dated record of each run is kept with the compiler versions, driver and commit.

What this establishes is narrower than it may sound, and the limits should be stated as plainly as the result. All three programs were written by one hand from one set of definitions, so their agreement is evidence of implementability and of consistency on these inputs, at the recorded checkpoints. It does not establish that the written specification is complete, since an ambiguity that these inputs never exercise, or an assumption that the author carried unwittingly into all three programs, would not show. It does not show that the definitions are the best available, and it does not show agreement on inputs outside the three test sets, which draw on a single text source with sequences of at most 160 tokens here and positions below 512 in the profile. The exponential, SiLU and rotary tables are generated by the prototype with arbitrary-precision arithmetic and read by the other programs from the model blob, so those

programs do not regenerate them, and an independent regeneration with a pinned digest is still owed before a specification is frozen. Accumulator safety is argued from the formats, with the sum of squares given a proof for all inputs, but the argument has not been checked by exhaustive or adversarial search. The check was run at batch size one without cached decoding.

8 Relevance

The first use of such a specification is verification. A reference that fixes every intermediate state allows a kernel author to test an optimized implementation against a target that has no tolerance in it, and to localize a defect to the layer, and within a checkpoint to the element, where it first appears, since a mismatch is reported with both values. The usual practice of comparing logits against a floating point implementation within a tolerance cannot do this, because a tolerance hides the small errors that are the early signature of an indexing or rounding mistake. The second use is independent recomputation. An exact execution specification supplies a stable target, allowing a verifier to compare a claimed output against a uniquely defined result. This does not establish that the claimed execution actually took place, and bitwise reproducibility is not a universal prerequisite for verifying computation, but it makes the comparison a matter of equality and not of judgment about tolerance.

The third relevance is to the analysis of nondeterminism. The explanation advanced for variation in served inference locates the problem in kernels whose reduction order depends on batch size [5]. In an integer specification with exact accumulation, reduction order cannot affect a result, so batch invariance of the matrix products follows from the definition and does not have to be engineered kernel by kernel. This is a structural consequence of the design and it is untested at any batch size above one, since only batch size one was run, and the remaining batch-dependent behavior would be confined to places where a rounding point is placed before a full accumulation, which the specification does not contain. Whether the discipline survives the optimizations that practical kernels demand, such as split reductions and fused operations, is a question for a faster implementation than the present ones.

The fourth relevance concerns quantization practice. The measured finding that the arithmetic contributes almost nothing beyond the weights, while narrow storage of the residual stream and the flagged sites together cost nearly twenty times as much as all of the weights together, with about four fifths of that excess arising in the residual stream, redirects attention from the matrix multiplications, where most quantization work concentrates, to the residual stream and to the few sites that carry massive values. A profile that reports a per-tensor maximum and a percentile is enough to detect the problem, and the cheapest remedies in this setting were wider storage or a per-channel exponent for the residual stream, together with wider storage for the gated product, and not a transformation of the weights. How this scales to larger models, where outlier channels are known to be more pervasive [3], is not answered by a single 135-million-parameter case.

9 Limitations and Open Problems

The ablation leaves two questions open. It does not show that a per-channel residual can be specified end to end, since the residual addition and the normalization across channels with different exponents have not been defined or implemented. And it does not say whether the gated-product sites, which carry the remaining 0.012 nats, can use a format cheaper than 32 bits. The evidence is a single text source, windows of 512 tokens at most, batch size one, no cached decoding, and eight windows for the quality figures, so the percentile and clipping figures describe this distribution and not a worst case, and rare tokens may produce maxima that grow beyond the calibration values. The largest ratio of a validation maximum to its calibration maximum was 1.77, at the mean square of the first normalization, and 1.69 at the gated product, so one spare bit is adequate here but not generous at the heavy-tailed sites. The score resolution has not been varied, and a table-plus-Newton method for the reciprocal root is acceptable only if it reproduces the same integers, otherwise the definition should change to that method. The conformance programs are unoptimized, and nothing here measures the speed or memory that an integer specification could recover. The specification is not frozen. The most useful next steps are an independent regeneration and hashing of the generated tables, and a second implementation written by a different hand, since only that can test the definitions and not merely their transcription. A cache is also needed, because no program implements one yet and the continuation positions are recomputed over the full prefix; with a cache added to at least one program, cached and uncached execution over the same fixed 32-token continuation would test arithmetic equivalence, and a separate greedy generation from the 128-token prompt would test the resulting token trajectory.

10 Conclusion

A pretrained transformer can be executed as a function on integers whose every intermediate state is fixed by a written specification, and three separately written programs, on a CPU and a GPU, reproduce that function exactly on the inputs tested. The cost in model quality of the arithmetic itself is negligible beside the cost of eight-bit weights, provided that the residual stream is stored with 32 bits or with a per-channel exponent and the gated-product sites stay wide, a requirement that the sink-token measurements explain without the ablation isolating the sink token as its sole cause. That finding concerns number formats, and the more general result is methodological: a specification that names every place where information is discarded turns questions about reproducibility, about the origin of numerical error, and about the safety of each format from matters of opinion into matters that can be tested one site at a time.

Availability

The pinned manifest, profile, scaling proposal, test vector hashes, source code for all three implementations, and the dated conformance records are in the repository at `https://github.com/standardgalactic/exact-integer-inference`, released under CC0 1.0 Universal. The implementation evaluated for the conformance results is commit

`c798574d20700ed5eeda25318442a370c3a12467`. The ablations of Section 6 were produced by `python/ablate_narrow.py`, `python/ablate_perchannel.py` and `python/ablate_layers.py`, which were added to the repository after that commit.

References

[1] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[2] Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When Smol Goes Big, Data-Centric Training of a Small Language Model. arXiv preprint arXiv:2502.02737, 2025.

[3] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. LLM.int8(): 8-Bit Matrix Multiplication for Transformers at Scale. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022.

[4] David Goldberg. What Every Computer Scientist Should Know About Floating-Point Arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991.

[5] Horace He, in collaboration with others at Thinking Machines Lab. Defeating Nondeterminism in LLM Inference. *Thinking Machines Lab: Connectionism*, September 2025.

[6] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2704–2713, 2018.

[7] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W. Mahoney, and Kurt Keutzer. I-BERT: Integer-Only BERT Quantization. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of Proceedings of Machine Learning Research, pages 5506–5518, 2021.

[8] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. In *International Conference on Learning Representations*, 2017.

[9] Noam Shazeer. GLU Variants Improve Transformer. arXiv preprint arXiv:2002.05202, 2020.

[10] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced Transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024.

[11] Mingjie Sun, Xinlei Chen, J. Zico Kolter, and Zhuang Liu. Massive Activations in Large Language Models. In *First Conference on Language Modeling (COLM)*, 2024.

[12] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of Proceedings of Machine Learning Research, pages 38087–38099, 2023.

[13] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient Streaming Language Models with Attention Sinks. In *International Conference on Learning Representations*, 2024.

[14] Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, 2019.