GPU Kernel Optimization Task Datacard

English edition 29 October 2026

Dataset introduction

Our dataset is purpose-built for automated GPU kernel optimization (Auto Research). Experts develop tasks from real engineering workloads, write reference implementations, and cover tensor shapes encountered in production. In our evaluations, frontier models achieve roughly 1.5× greater speedups than less capable models.

Our key differentiators are expert curation, realistic workload coverage, and strong customization capabilities. A broad network of contributing engineers with deep domain knowledge enables us to tailor tasks to customers' hardware, GPU architectures, and operator domains.

We see this combination of practical relevance and engineering support as a core advantage over general-purpose open-source datasets and standardized benchmark offerings.

Datacard contents

The overview covers task counts, domains and runtime specifications. The results section defines correctness and performance metrics. Each of T01 through T07 has a dedicated page covering inputs, outputs, computation, workload boundaries, optimization challenges and results.

Task set overview

This card describes seven GPU optimization tasks spanning quantized linear algebra, sequence modeling, sparse attention, training loss and mixture-of-experts computation. Evaluation considers both correctness of the complete pipeline and execution latency, including precision boundaries, irregular layouts and dependent operations.

Each task has 16 workloads, for 112 in total. Two evaluation groups ran twice per task, giving 28 attempts. Of these, 27 produced final submissions and passed every workload for their task; 26 outperformed the reference. One attempt lacked a final submission.

Task composition

ID	Task	Primary domain	Workloads
T01	INT4 linear projection and activation	Quantized linear algebra	16
T02	RMSNorm and FP8 linear projection	Quantized linear algebra	16
T03	Causal convolution and gated projection	Sequence modeling	16
T04	Paged sparse GQA decode	Sparse attention	16
T05	Vocabulary projection with loss and gradient	Training loss and gradients	16
T06	INT4 SwiGLU expert computation	Mixture of experts	16
T07	Variable route expert projection and combine	Mixture of experts	16

The primary-domain distribution is two quantized linear tasks, one sequence task, one sparse attention task, one loss-and-gradient task and two mixture-of-experts tasks. The expert tasks use fixed top-k routing and variable-degree routing respectively.

Data and runtime specifications

Inputs include floating tensors, integer indices and scalar parameters. Workloads are explicitly enumerated dimension/layout records, with fresh random values for each call. Reported bounds summarize coverage; they do not imply that every combination within those bounds was tested.

The target is one NVIDIA H200 GPU with two CPU threads. Multiple kernels and available matrix libraries are allowed. The entire input-dependent pipeline is timed; inputs are read-only. Floating outputs must meet error, shape, dtype and finiteness checks.

Results and metric definitions

A denotes GPT-6.1-SOL and B denotes HY4 Preview. Each model ran twice per task. Values are speedups over the reference, rounded to three decimals. Missing submissions are marked separately and have no numerical score.

ID	Passed / tried	A run 1	A run 2	B run 1	B run 2
T01	4/4	3.176×	3.393×	2.323×	2.958×
T02	3/4	3.116×	2.898×	Missing	0.980×
T03	4/4	2.918×	3.063×	1.311×	2.819×
T04	4/4	4.873×	4.603×	4.599×	3.898×
T05	4/4	4.104×	3.728×	2.866×	2.461×
T06	4/4	8.995×	10.244×	8.930×	8.057×
T07	4/4	6.369×	6.506×	5.836×	5.577×

The final-submission-and-pass rate is 27/28, or 96.4%; all 27 final submissions passed. Group A achieved 14/14 and B achieved 13/14. B run 1 on T02 lacked a submission; B run 2 passed but was slightly slower than the reference.

Correctness and performance

Each floating output uses global normalized root mean square error (NRMSE) with a 0.01 threshold: error RMS divided by reference RMS, with a denominator floor of 0.000000000001. Integer outputs require exact equality. Formal performance scores require complete correctness.

The default schedule is 10 correctness calls, 10 warmups and three rounds of 10 timed calls. Timed outputs are checked. For each workload, take the median of the three mean latencies, then sum across its task’s 16 workloads.

Speedup is total reference latency divided by total submission latency. The score is one minus submission latency divided by reference latency, floored at zero. A correct submission without a speedup remains correct but receives a performance score of zero.

Interpretation limits

Workloads are not weighted by production frequency. Timing includes process communication, synchronization and output cloning or transport, rather than pure GPU kernel time. Expert-task speedups may partly reflect removal of reference per-expert scheduling costs.

Two runs per group and a missing submission do not support statistical-significance or cross-hardware claims. Full-model throughput, production latency and training convergence require separate evaluation.

T01 Grouped INT4 Linear Projection and SiLU

Quantized linear algebra | 16 workloads

A dependent pipeline combines packed-weight decoding, grouped scaling, linear projection, bias and activation. It tests how a quantized representation is integrated with matrix computation.

Computation and interface

Five inputs: BF16 activations x[M,K], uint8 packed weights[N,ceil(K/2)], FP32 group scales, BF16 bias and group size G. The output is a contiguous BF16 tensor y[M,N].

The low nibble stores the even channel and the high nibble the odd channel, both as signed INT4. Decode and scale weights in FP32, then compute the matrix product, bias and SiLU in FP32 before the final BF16 cast. For odd K, the last high nibble is unused.

Workload coverage

The 16 workloads contain 15 distinct dimension combinations. M ranges from 1 to 2047, K from 1024 to 7169, and N from 257 to 2049; G is 64 or 128. Cases cover odd dimensions, partial groups, positive noncontiguous strides, varying group scales and cancellation.

Challenges and optimization analysis

Sign extension, nibble order and group indexing must agree. Unsigned decoding, swapped nibbles or a shared scale change the result. Rounding decoded weights to BF16 early also changes the specified arithmetic and must satisfy the final error check.

Possible optimizations include decoding and scaling inside tiled computation to reduce FP32 weight traffic, or retaining a matrix-library path while fusing preprocessing and epilogues. Small and large M may need different tiling and scheduling. Weight preprocessing is part of each call.

Results

A: 3.176×, 3.393×; B: 2.323×, 2.958×.

All four runs produced final submissions and passed every workload. Both groups outperformed the reference, demonstrating achievable optimization gains for this pipeline under the specified workloads and timing protocol.

T02 RMSNorm and Dynamic FP8 Linear Projection

Quantized linear algebra | 16 workloads

This pipeline combines normalization, dynamic scale reductions, FP8 quantization, matrix multiplication and scale restoration. It tests both numerical semantics and fusion decisions.

Computation and interface

Four BF16 inputs: x[M,K], normalization weights gamma[K], matrix weights[K,N] and bias[N]. The output is a contiguous BF16 tensor y[M,N].

Compute RMSNorm in FP32 with epsilon 0.000001 and multiply by gamma. Obtain separate maximum-absolute-value scales for each activation row and weight column, then quantize to E4M3FN with round-to-nearest-even. Accumulate the quantized matrix product in FP32, restore row and column scales, add bias, and cast to BF16.

Workload coverage

The 16 workloads contain 15 dimension combinations. M ranges from 1 to 2047, K from 1024 to 7169, and N from 257 to 2049. Cases include positive noncontiguous strides, values near the normalization epsilon transition, varied dynamic ranges and signed values. Zero rows and columns use a protected scale floor.

Challenges and optimization analysis

FP8 quantization and row/column scale granularity are part of the mathematical definition. Omitting gamma, skipping quantization or replacing row scales with one tensor-wide scale can fail validation. Normalized values and matrix outputs before scale restoration retain the prescribed FP32 precision.

Normalization, row maxima and activation quantization can be designed together, but reductions and matrix multiplication impose synchronization and layout constraints. Compare fused preprocessing, reusable scale tensors and FP8 matrix paths. Dynamic weight quantization remains input-dependent work within the call.

Results

A: 3.116×, 2.898×; B: missing submission, 0.980×.

Three of four runs produced final submissions; all three passed every workload. Both A runs accelerated the reference. B had one valid submission at 0.980×, slightly slower than the reference; its other run lacked a final submission, so numerical correctness cannot be inferred.

T03 Causal Convolution with Gated Projection and Residual

Sequence modeling | 16 workloads

A causal depthwise convolution is followed by SiLU, channel gating, output projection and residual addition, with variable sequence lengths and tensor layouts.

Computation and interface

Seven inputs: BF16 x[B,T,D], convolution weights[D,W], bias[D], gate[B,T,D], projection[D,O] and residual[B,T,O], plus int32 lengths[B]. The output is BF16[B,T,O].

Convolution reads only current and earlier positions; the final tap applies to the current token, and positions before the sequence start are zero. Accumulate in FP32, apply SiLU and the supplied gate, then round to BF16. Compute projection and residual addition in FP32, then output BF16. Positions beyond each valid length are defined as zero, including the residual contribution.

Workload coverage

The 16 workloads contain 12 dimension combinations. B ranges from 1 to 7, T from 7 to 1023, D from 32 to 512, and O from 64 to 512; W is 3 or 4. Cases cover ragged lengths, channel strides, padded time layouts, zero gate channels and bias-dominated values.

Challenges and optimization analysis

Convolution direction and sequence boundaries matter. The gate is a supplied signed multiplier, with no extra sigmoid. The post-gating BF16 cast is a required semantic boundary. Ignoring lengths or retaining residuals in padded positions changes the result.

Fuse convolution, activation and gating to reduce intermediate traffic. Select a matrix-library or specialized tiled projection according to dimensions. Carry the length mask throughout computation to avoid invalid work, while preserving the required post-gating rounding when fusing the projection.

Results

A: 2.918×, 3.063×; B: 1.311×, 2.819×.

All four runs passed. The A results were close, while the B speedups varied substantially. Different runs can reach different performance levels on the same task; one result is insufficient to establish stable performance.

T04 Paged Sparse Grouped Query Attention Decode

Sparse attention | 16 workloads

Logical-index mapping, paged KV access, FP8 decoding and sparse grouped query attention are combined to test irregular memory access and masking semantics.

Computation and interface

Twelve inputs include BF16 queries; int32 selected indices, request mapping, page table, validity flags and sequence lengths; uint8 K/V codes; FP32 K/V scales; block size; and attention scale. Outputs are BF16 attention[M,HQ,D] and int32 valid counts[M].

Map local indices through the request page table to physical pages and offsets. Decode E4M3FN K/V and apply the physical-page and KV-head scales. Map query heads to KV groups and compute scores, softmax and weighted sums in FP32. Mask negative indices, out-of-length positions and invalid queries. Duplicates remain separate softmax terms; all-invalid rows are defined as zero.

Workload coverage

The 16 workloads contain 12 dimension combinations. Query count M ranges from 1 to 65 and selected length K from 31 to 256. Head dimension D is 64 or 128; query heads range from 2 to 12, KV heads from 1 to 4, and block sizes from 16 to 64. Cases include hotspots, local and duplicate indices, negative sentinels and mixed scales.

Challenges and optimization analysis

Page mapping, head grouping and validity jointly determine attention participants. Do not deduplicate indices or omit K/V scales. FP8 specifies KV storage here; attention operates on decoded values using FP32 arithmetic.

Fuse index mapping, KV reads and dequantization into attention tiles, and use online softmax to reduce intermediates. Balance page locality against reuse across queries, and handle all-invalid rows without producing nonfinite values.

Results

A: 4.873×, 4.603×; B: 4.599×, 3.898×.

All four runs passed, with speedups from 3.898× to 4.873×. Both groups reduced protocol-level latency. The measured gains include memory access, scheduling and fixed costs within the timing protocol.

T05 Vocabulary Projection with Smoothed Loss and Hidden Gradient

Training loss and gradients | 16 workloads

Starting from hidden states, compute vocabulary projection, label-smoothed cross entropy and the hidden-state gradient, covering forward and local backward dependencies.

Computation and interface

Eight inputs: BF16 hidden states[rows,hidden], weights[vocab,hidden] and bias[vocab]; int64 targets; and scalar smoothing, logit scale, z-loss scale and ignore index. Explicit outputs are an FP32 scalar loss and BF16 hidden gradient[rows,hidden].

Compute projection, bias and logit scaling in FP32, then form a numerically stable loss using logsumexp. Include label smoothing and z-loss, averaging over valid target rows only. The hidden gradient includes the logit-scale chain factor, smoothing and z-loss derivative before multiplication by the weights. Ignored rows have zero gradient by definition.

Workload coverage

The 16 workloads contain 13 dimension combinations. Rows range from 1 to 257, hidden width from 64 to 512, and vocabulary width from 4096 to 32768. Cases cover smoothing from 0 to 0.2, logit scale from 0.5 to 1, z-loss scale from 0 to 0.01, and varied ignore fractions, target sentinels and noncontiguous layouts.

Challenges and optimization analysis

Forward loss alone does not validate this task. Missing the chain factor, smoothing term or z-loss derivative can invalidate the gradient. Projection and reductions retain FP32 semantics; early BF16 rounding of logits is not the prescribed arithmetic.

Writing and rereading full logits can add traffic. Compare vocabulary tiling, online reductions and recomputation. Joint forward/backward reductions can save memory traffic, but recomputation adds matrix work. With many ignored rows, evaluate whether compacting valid rows offsets the extra scheduling cost.

Results

A: 4.104×, 3.728×; B: 2.866×, 2.461×.

All four runs passed, with speedups from 2.461× to 4.104×. Joint loss and hidden-gradient computation offers optimization opportunities, although the runs reached different performance levels.

T06 Fixed Route INT4 SwiGLU Expert Computation

Mixture of experts | 16 workloads

A quantized expert feed-forward network follows fixed top-k routing and returns exactly verifiable routing metadata, linking sparse dispatch with multiple matrix stages.

Computation and interface

Nine inputs: BF16 tokens[M,D], int32 expert IDs[M,K], FP32 router weights[M,K], and uint8 packed gate, up and down weights with their FP32 scales. Outputs are BF16[M,D], int32 expert offsets[E+1] and stable edge order[M×K].

Stable-group edges by expert ID, preserving input order within ties. Decode signed INT4, apply scales in groups of 32 input channels, and cast weights to BF16. Compute gate and up projections, SwiGLU and the down projection with the specified BF16 boundaries. Combine using FP32 router weights and output BF16. Repeated expert edges are weighted separately.

Workload coverage

Eight geometries paired with two load scenarios give 16 workloads. M ranges from 7 to 2047, D from 64 to 1024, and intermediate width I from 96 to 1024. Expert count E is 4, 8 or 16; K is 2, 3 or 4 per token. Cases include balanced, hotspot and few-active-expert routing, plus positive noncontiguous strides.

Challenges and optimization analysis

Routing order and offsets require exact integer equality. The floating path also depends on INT4 signs, grouped scales and several BF16 rounding points. Router weights cannot be dropped, and duplicate edges must retain their contributions.

Grouped or batched GEMMs can reduce per-expert dispatch. Fuse decoding, activation and matrix epilogues where useful. A hotspot can waste padded capacity, motivating hot/cold expert paths and adaptive capacity. Large speedups may partly reflect reduced per-expert scheduling in the reference.

Results

A: 8.995×, 10.244×; B: 8.930×, 8.057×.

All four runs passed, with speedups from 8.057× to 10.244×, the largest gains in this sample. These ratios apply to the current reference and fixed workloads; they are not full-model end-to-end speedups.

T07 Variable Route Expert Projection and Weighted Combine

Mixture of experts | 16 workloads

An unordered sparse edge list, including duplicates, describes token-to-expert relations. The task jointly computes routing metadata, expert projections and weighted per-token combination.

Computation and interface

Six inputs: BF16 tokens[T,D], int32 token IDs[L], int32 expert IDs[L], FP32 edge weights[L], BF16 expert weights[E,D,O] and FP32 bias[E,O]. Seven outputs comprise six int32 routing metadata tensors and a BF16 result[T,O].

Stable-sort by expert ID and produce frequencies, expert prefix offsets, gathered token IDs, edge permutation, inverse permutation and token-degree prefix counts. Project each edge and round to BF16; add bias, apply edge weights and sum per token in FP32. Apply SiLU only after the full combination, then cast to BF16.

Workload coverage

Eight geometries paired with two load scenarios give 16 workloads. T ranges from 7 to 2047, D from 64 to 512, O from 96 to 1024, and edge count L from 19 to 8191; E is 4, 8 or 16. Cases cover unordered and duplicate edges, empty tokens and experts, hotspots and noncontiguous layouts. Token degree is variable.

Challenges and optimization analysis

The edge permutation and its inverse are different. Token-degree prefix counts do not imply token-sorted input edges. Duplicates contribute independently, and empty tokens are defined as zero. Applying SiLU to each edge before combination changes the operator.

Reuse one stable grouping for metadata and expert dispatch to reduce index processing. Batched projection can lower scheduling costs but may waste padding under skew; compare dynamic capacity and separate expert paths. Fusing edge weighting with scatter combination can avoid materializing all edge contributions.

Results

A: 6.369×, 6.506×; B: 5.836×, 5.577×.

All four runs passed, with speedups from 5.577× to 6.506×. Its single-stage BF16 projection and variable routing complement T06, which uses fixed top-k routing and a multistage INT4 expert network.