

BiGRU-Transformer GPU Memory Requirement Prediction for Transformer Training Workloads on GPUMemNet

Jordan Smith

Computer Science, University of Maryland, College Park, MD, USA
jsmith_umd_v2@yahoo.com

ABSTRACT

Abstract—Accurate GPU memory estimation supports scheduling, admission control, and out-of-memory prevention for neural-network training. This study evaluates memory-requirement regression on the GPUMemNet MLP, CNN, and Transformer datasets, with the principal analysis centered on Transformer workloads. The target is peak GPU memory in MiB, and the predictors are restricted to architecture-level information available before execution. The evaluation includes 3,000 MLP records, 8,920 CNN records after removing 80 non-positive target rows, and 5,011 Transformer records. A BiGRU-Transformer feature-token regressor is compared with Ridge, K-nearest neighbors, decision tree, random forest, Extra Trees, and gradient boosting under a fixed 70/15/15 train/validation/test split with seed 42 and target-quantile stratification. On Transformer workloads, the BiGRU-Transformer achieves 1,191.3 MiB MAE, 1,827.7 MiB RMSE, $R^2 = 0.975$, and 86.6% accuracy for 8-GB memory bins. Extra Trees gives the lowest Transformer RMSE at 1,248.8 MiB and also leads on MLP and CNN, indicating that the aggregated GPUMemNet descriptors strongly favor randomized tree ensembles. The neural estimator remains competitive, offers a structured mechanism for modeling interactions among workload descriptors, and provides a practical foundation for future layer-sequence inputs.

1. Introduction

Modern neural-network training is constrained by accelerator memory as well as arithmetic throughput. Transformer architectures made self-attention a general sequence-modeling primitive [1], and later language models increased depth, hidden width, sequence length, and batch-level activation volume [5]-[7]. These choices raise the amount of memory required for parameters, gradients, optimizer states, activations, and temporary workspaces. A job admitted with an insufficient reservation can terminate with an out-of-memory error, whereas an overly conservative reservation leaves expensive accelerator capacity idle.

The scheduling decision often precedes the first complete forward and backward pass. At that point a resource manager may know the model family, batch size, sequence length, number of layers, embedding dimension, and parameter counts, but not the allocator peak that will occur during execution. The estimator must therefore operate under a pre-execution information constraint. Distributed training methods such as Megatron-LM and ZeRO redistribute model states across devices [8], [9], activation checkpointing trades memory for recomputation [10], and mixed-precision training changes storage requirements for weights and activations [11]. Hybrid-cloud deployment adds further variation in hardware, software stacks, and cost objectives [47]. These interactions make a single architecture-independent accounting formula difficult to calibrate.

GPUMemNet provides controlled MLP, CNN, and Transformer workload records with architecture descriptors and measured peak memory [25]. The three workload families support a useful progression: MLP configurations provide a comparatively direct relationship between model size and memory; CNN configurations introduce image dimensions, convolutional patterns, and many high-memory cases; and Transformer configurations expose the batch, sequence, embedding, attention, and depth variables that motivate a sequence-aware estimator. Using all three subsets establishes whether a method is broadly effective or merely fitted to one architecture family.

This study proposes a BiGRU-Transformer regressor for the Transformer subset. Each standardized descriptor is represented as a feature token. A bidirectional gated recurrent unit models contextual relationships in both directions [2], [3], and a Transformer encoder captures global interactions among tokens [1]. The design draws on gated sequence modeling while remaining smaller than an LSTM-based alternative [4]. It is evaluated

beside linear, local, single-tree, bagged-tree, randomized-tree, and boosted-tree regressors so that the neural architecture is judged against strong tabular baselines [14]–[19].

The study makes four contributions. First, it defines a leakage-controlled protocol in which only pre-execution architecture descriptors are used. Second, it provides a consistent comparison across the MLP, CNN, and Transformer datasets and a focused neural evaluation on Transformer workloads. Third, it combines continuous error measures with scheduler-oriented 8-GB bin accuracy and tolerance rates. Fourth, it analyzes memory-range errors, ablations, validation behavior, and feature importance to explain why the proposed model is competitive while Extra Trees remains the strongest predictor on the present tabular representation.

2. Literature Review

2.1 GPU Memory Estimation

GPU memory estimators can be organized into analytical, CPU-side, and data-driven approaches. DNNMem derives memory consumption from model structure and execution characteristics [26], while Horus combines prediction with interference-aware scheduling [27]. Such methods are fast and interpretable but depend on detailed architectural assumptions and may not fully reflect framework-level allocation, reuse, and kernel workspace behavior. LLMem specializes analytical estimation for fine-tuning pretrained language models [28], illustrating the value of domain-specific formulas but also the difficulty of transferring them across workload families.

CPU-side analysis seeks to reproduce relevant execution behavior without occupying a GPU. xMem performs dynamic analysis on the CPU to estimate peak training memory before accelerator execution [29]. Data-driven systems instead learn the mapping from workload descriptors to resource consumption. Graph-based runtime prediction has modeled deep-learning performance from computation structure [30], and GPUMemNet studies machine-learning estimators over controlled MLP, CNN, and Transformer configurations [25]. These approaches reduce reliance on hand-derived formulas, although they remain sensitive to feature coverage, new layer types, framework changes, and hardware distribution shift.

2.2 Scheduling, Collocation, and Capacity Planning

Memory prediction is operationally valuable because it informs placement and collocation. Gandiva uses introspective scheduling for deep-learning jobs [31], Gavel accounts for heterogeneous accelerators [32], and Orion supports fine-grained interference-aware GPU sharing [33]. Empirical work on collocated training shows that shared-device performance depends on resource interactions rather than memory capacity alone [34]. Large-scale cluster studies similarly document heterogeneous workload mixes, utilization gaps, and the importance of scheduling policy [35], while profiling frameworks provide the measurements required to characterize training behavior [36].

Recent capacity-planning research extends this perspective from a single job to fleet-level uncertainty. Time-series foundation models have been used for calibrated capacity forecasting in heterogeneous GPU clusters [37], and cross-cloud transfer learning addresses workload and topology shift [38]. Conformal demand envelopes support risk-bounded oversubscription [39], while instance-readiness and scheduling-delay models connect forecast outputs to salable GPU supply [40]. Multi-horizon demand forecasting with operational risk curves [41] and few-shot cold-start forecasting for new inference tenants [42] address temporal and tenant-level uncertainty. Profit-aware admission control [43] and power-aware inventory planning [44] further show that resource predictions become most useful when coupled to explicit operational objectives.

2.3 Operational Reliability and Human Oversight

A memory estimate ultimately enters a broader operational workflow. Noisy-neighbor modeling can identify degradation risk in shared virtualized environments [45]. Root-cause attribution over CPU and network faults [48], retrieval-grounded failure narratives for distributed logs [49], evidence-grounded DevOps question answering [50], and bilingual incident triage [51] illustrate how predictive signals can be linked to diagnosis. Self-supervised log anomaly detection [53] and conformal alert control with grounded incident tickets [54] provide complementary mechanisms for detecting and communicating abnormal behavior. Incident visualization cards [52] and capacity-dashboard brief cards [46] emphasize that uncertainty, evidence, and

recommended actions must be presented clearly to operators rather than reduced to an isolated point estimate.

2.4 Sequence Models and Tabular Baselines

The proposed architecture lies at the intersection of sequence representation and tabular regression. Recurrent neural networks learn distributed representations through backpropagation [24], GRUs provide a compact gating mechanism [2], [3], and bidirectional processing can contextualize each feature using information on both sides of an ordered token sequence. Self-attention then models long-range pairwise interactions without recurrence [1]. However, the GPUMemNet descriptors are also a low-dimensional tabular representation with threshold effects. Random forests [14], Extra Trees [15], and gradient boosting [16] are therefore essential comparators, alongside Ridge regression [17], nearest-neighbor regression [18], and a single decision tree [19].

3. Method

3.1 Data and Prediction Target

The prediction target is Max GPU Memory (MiB). Rows with non-positive targets are removed because they do not represent a usable measured memory requirement. This filtering affects only the CNN file, where 80 records have zero memory and empty activation information. Positive-memory rows marked as out of memory are retained as high-memory observations; their possible censoring near device capacity is considered in the limitations. Table 1 summarizes the three files and the predictor counts, and Table 2 describes the target distributions after filtering.

The workflow in Figure 1 begins with architecture-specific CSV ingestion, applies target filtering and leakage control, creates quantile-stratified train, validation, and test partitions, and then sends the transformed predictors to classical regressors or the BiGRU-Transformer. Predictions are returned to MiB before metric calculation, bin analysis, and feature diagnostics.

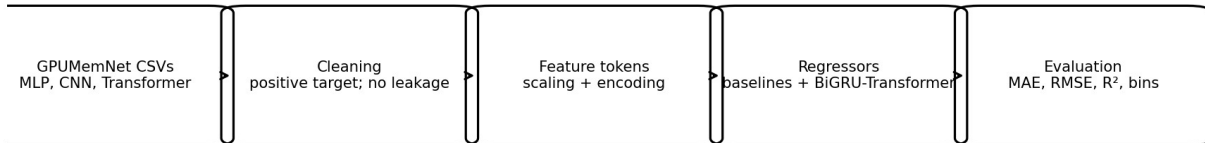


Figure 1. End-to-end evaluation workflow for the three GPUMemNet workload families.

Table 1. GPUMemNet files and rows used in the evaluation.

Dataset	CSV	Raw	Used	Excluded	Columns	Predictors	Numerical	Categorical	Size (MB)	SHA-256 prefix
MLP	mlp_data_step2.csv	3,000	3,000	0	15	7	6	1	2.070	c095068ad3925800
CNN	cnn_data_step1.csv	9,000	8,920	80	24	15	13	2	15.930	1856b809c115cd31
Transformer	transformer_data.csv	5,011	5,011	0	17	13	13	0	14.500	957857b58141220d

Table 2. Target-memory distribution after positive-target filtering.

Dataset	Rows	Min (MiB)	P25 (MiB)	Median (MiB)	Mean (MiB)	P75 (MiB)	Max (MiB)	SD (MiB)	OOM retained
MLP	3,000	1,443	1,491	1,576	1,743.3	1,741	4,925	469.892	0

Dataset	Rows	Min (MiB)	P25 (MiB)	Median (MiB)	Mean (MiB)	P75 (MiB)	Max (MiB)	SD (MiB)	OOM retained
CNN	8,920	1,451	5,472.5	18,419	21,238.6	38,857.5	40,369	15,375.9	2,797
Transformer	5,011	1,683	5,056	10,453	14,138.9	18,918	40,369	11,538.4	472

3.2 Leakage Control, Preprocessing, and Splitting

Predictors are restricted to architecture- and size-level fields available before or at job-submission time. Filename and Activations-Params are not included in the baseline table. Status is excluded because it is an outcome label, and Avg GPUTL, Avg GRACT, Avg SMACT, Avg SMOCC, and Avg FP32A are excluded because they are observed during execution. This separation prevents a scheduler from relying on information that becomes available only after the job has started.

The split is fixed at 70% training, 15% validation, and 15% test with random seed 42. Target-quantile stratification preserves low-, medium-, and high-memory cases in each partition. Table 3 shows the split sizes and similar target means across partitions. The held-out validation and test design follows established practice for model selection and unbiased evaluation [22]. Numeric inputs are standardized, categorical inputs are one-hot encoded, and classical regressors use $\log_{10}(y)$ before inverse transformation to MiB. Table 4 records these preprocessing decisions.

Table 3. Train, validation, and test split sizes and target means.

Dataset	Train	Validation	Test	Train mean	Validation mean	Test mean
MLP	2,100	450	450	1,741.3	1,751.3	1,744.6
CNN	6,244	1,338	1,338	21,228.5	21,322.2	21,201.8
Transformer	3,507	752	752	14,123.3	14,256.8	14,093.7

Table 4. Preprocessing and leakage-control decisions.

Pipeline item	Implemented choice	Reason
Target	Max GPU Memory (MiB)	Continuous regression target; rows with non-positive target removed.
Excluded text fields	Filename; Activations-Params	Filename encodes configuration strings and raw layer traces are variable-length; they were not used in baseline tabular features.
Excluded leakage fields	Status; Avg GPUTL; Avg GRACT; Avg SMACT; Avg SMOCC; Avg FP32A	Status and GPU-utilization counters are observed after launch and were not used as predictors.
Numeric preprocessing	StandardScaler	Applied to all numeric architecture predictors.
Categorical preprocessing	OneHotEncoder	Applied to activation-function and CNN architecture labels where present.
Target transform	$\log_{10}(y)$	Applied to all classical regressors and inverted for reporting in MiB.

3.3 Baseline Regressors

The comparison includes Ridge regression [17], K-nearest neighbors [18], a decision tree [19], random forest [14], Extra Trees [15], and gradient boosting [16], implemented with scikit-learn [13]. These models span smooth linear scaling, local similarity, threshold rules, bagging, randomized partitioning, and additive residual correction. Hyperparameters are fixed to provide a controlled comparison rather than an extensive search. Table 5 lists the settings and each model's role in the evaluation.

Tree ensembles are especially relevant because GPU memory can change abruptly when a descriptor crosses an allocation or architecture threshold. A single tree can represent these discontinuities but is sensitive to partition variance. Random forest reduces that variance through bagging, Extra Trees introduces additional split randomization, and gradient boosting fits successive residual corrections. Their inclusion tests whether a sequence model adds value beyond established tabular methods.

Table 5. Model hyperparameters used in the empirical comparison.

Model	Hyperparameters used	Role in evaluation
Ridge	alpha=1.0, log1p target	Linear baseline for smooth memory scaling.
KNN	k=7, distance weighting, log1p target	Local nonparametric baseline.
Decision Tree	max_depth=12, min_samples_leaf=3, log1p target	Interpretable nonlinear baseline.
Random Forest	40 trees, min_samples_leaf=2, log1p target	Bagged tree ensemble baseline.
Extra Trees	50 randomized trees, min_samples_leaf=1, log1p target	Strong tabular ensemble baseline.
GradientBoosting	50 trees, learning_rate=0.06, max_depth=3, log1p target	Boosted tree baseline.
BiGRU-Transformer	32-d feature embeddings, BiGRU(32), one Transformer encoder layer, AdamW, 35 epochs	Proposed sequence-aware neural regressor for Transformer workloads.

3.4 BiGRU-Transformer Regressor

For the Transformer subset, each standardized input feature becomes one token. For feature j with standardized scalar value x_j , the initial representation is $e_j + W_v x_j$, where e_j is a learned feature-identity embedding and W_v projects the scalar value into the token dimension. The ordered token sequence enters a bidirectional GRU, whose forward and backward states contextualize each descriptor [2], [3]. A linear projection maps the recurrent output to the Transformer dimension, and one encoder layer applies multi-head self-attention [1]. Mean pooling feeds a two-layer regression head that predicts the standardized log target.

Figure 2 shows the resulting architecture. The recurrent stage captures local order and directional context, while self-attention allows every feature token to interact with every other token. This is useful for relationships such as batch size with sequence length, embedding size with parameter count, and depth with repeated attention and feed-forward blocks. Model parameters are learned by backpropagation [24] with AdamW, which combines adaptive optimization [20] with decoupled weight decay [21]. The model uses 32-dimensional feature embeddings, a BiGRU hidden size of 32, one Transformer encoder layer, and 35 CPU training epochs in PyTorch [12].

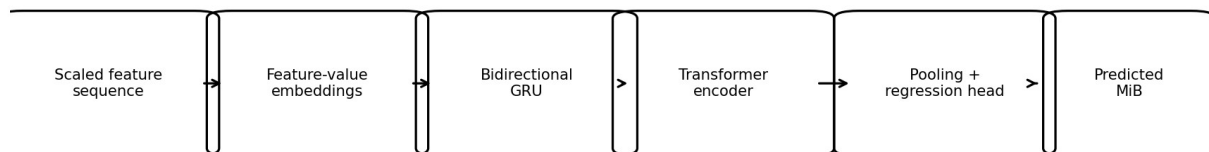


Figure 2. Feature-token BiGRU-Transformer architecture for Transformer memory regression.

3.5 Evaluation Measures

Performance is measured with mean absolute error (MAE), root mean squared error (RMSE), coefficient of determination (R^2), mean absolute percentage error (MAPE), 8-GB bin accuracy, and the fractions of predictions within 1 GB and 2 GB of the measured value. For observations y_i and predictions $\hat{y}_i$, MAE is the mean of $|y_i - \hat{y}_i|$, RMSE is the square root of the mean squared error, and R^2 is one minus the residual sum of squares divided by the total sum of squares.

The scheduler-oriented bin metric maps predictions into $[0, 8)$, $[8, 16)$, $[16, 24)$, $[24, 32)$, $[32, 40)$, and at least 40 GB. It complements continuous error because a resource manager often needs a device-capacity decision rather than only a numerically close estimate. Fit time is included for context but is not used as the principal ranking criterion.

4. Results and Discussion

4.1 Dataset Characteristics

The evaluation includes all positive-target records: 3,000 MLP, 8,920 CNN, and 5,011 Transformer configurations. Figure 3 shows a narrow low-memory MLP distribution and much broader CNN and Transformer distributions. The MLP median is 1,576 MiB. CNN has a median of 18,419 MiB, while Transformer has a median of 10,453 MiB; both reach 40,369 MiB. The broad upper tails make CNN and Transformer substantially more demanding regression problems and motivate the combined use of absolute error and capacity-bin accuracy.

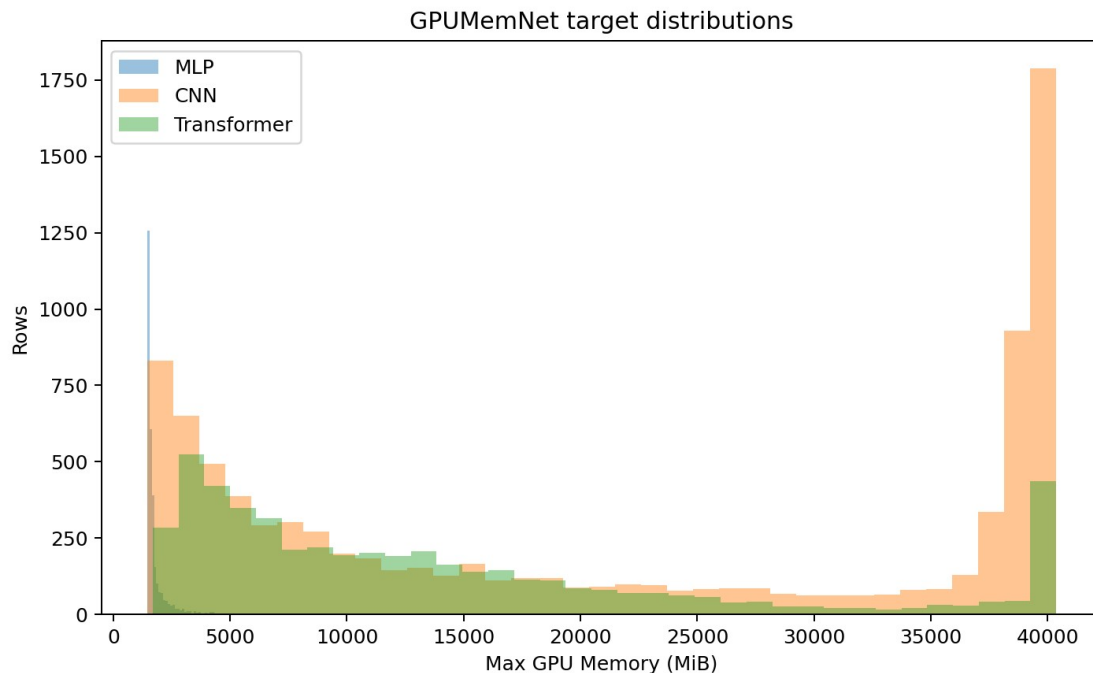


Figure 3. Target-memory distributions for MLP, CNN, and Transformer configurations after filtering.

4.2 Main Model Comparison

Table 6 presents the held-out test results. On MLP, Extra Trees gives the lowest RMSE at 28.0 MiB, followed by random forest at 30.6 MiB. On CNN, Extra Trees again leads with 875.9 MiB MAE, 1,672.2 MiB RMSE, $R^2 = 0.988$, and 93.1% 8-GB bin accuracy. The Transformer results show the same overall ranking: Extra Trees reaches 675.8 MiB MAE, 1,248.8 MiB RMSE, $R^2 = 0.988$, and 94.7% bin accuracy. The BiGRU-Transformer reaches 1,191.3 MiB MAE, 1,827.7 MiB RMSE, $R^2 = 0.975$, and 86.6% bin accuracy.

Table 6. Main regression results on the held-out test splits.

Dataset	Model	MAE (MiB)	RMSE (MiB)	R^2	MAPE (%)	8-GB bin acc.	Within 2 GB	Fit time (s)
MLP	Ridge	44.732	97.462	0.959	2.160	1.000	1.000	0.010
MLP	KNN	69.462	97.920	0.958	3.850	1.000	1.000	0.010
MLP	Decision Tree	16.513	35.027	0.995	0.810	1.000	1.000	0.020
MLP	Random Forest	13.729	30.589	0.996	0.670	1.000	1.000	0.340
MLP	Extra Trees	13.159	28.039	0.997	0.660	1.000	1.000	0.420
MLP	GradientBoosting	23.950	50.895	0.989	1.170	1.000	1.000	0.180
CNN	Ridge	8,387.1	13,157.0	0.265	52.240	0.427	0.270	0.020
CNN	KNN	4,064.9	6,431.2	0.824	35.770	0.679	0.516	0.010
CNN	Decision Tree	1,123.0	2,173.7	0.980	6.590	0.908	0.848	0.080
CNN	Random Forest	907.975	1,775.6	0.987	5.080	0.928	0.886	2.130

Dataset	Model	MAE (MiB)	RMSE (MiB)	R2	MAPE (%)	8-GB bin acc.	Within 2 GB	Fit time (s)
CNN	Extra Trees	875.907	1,672.2	0.988	4.740	0.931	0.892	1.740
CNN	GradientBoosting	1,727.6	2,580.4	0.972	9.800	0.902	0.651	0.980
Transformer	Ridge	5,747.8	13,508.4	-0.385	36.930	0.626	0.448	0.010
Transformer	KNN	1,699.9	3,192.2	0.923	14.620	0.868	0.781	0.010
Transformer	Decision Tree	1,237.6	2,350.7	0.958	9.410	0.888	0.815	0.030
Transformer	Random Forest	927.906	1,661.2	0.979	6.790	0.934	0.894	0.690
Transformer	Extra Trees	675.819	1,248.8	0.988	4.970	0.947	0.931	0.600
Transformer	GradientBoosting	1,811.4	3,246.1	0.920	12.630	0.826	0.775	0.380
Transformer	BiGRU-Transformer	1,191.3	1,827.7	0.975	10.300	0.866	0.838	21.580

Table 7 condenses the best-RMSE model for each dataset. Extra Trees is the only model that leads on all three workload families, with RMSE values of 28.0 MiB on MLP, 1,672.2 MiB on CNN, and 1,248.8 MiB on Transformer. The result indicates that aggregated parameter and activation totals, coupled with threshold-like memory behavior, are highly favorable to randomized tree partitions.

Table 7. Best test-RMSE model for each dataset.

Dataset	Model	MAE (MiB)	RMSE (MiB)	R2	MAPE (%)	8-GB bin acc.	Within 2 GB
CNN	Extra Trees	875.907	1,672.2	0.988	4.745	0.931	0.892
MLP	Extra Trees	13.159	28.039	0.997	0.661	1.000	1.000
Transformer	Extra Trees	675.819	1,248.8	0.988	4.974	0.947	0.931

Figure 4 compares RMSE across datasets and models. Ridge is adequate only for the narrow MLP range and yields negative R2 on Transformer because a log-linear trend cannot represent the interaction structure. K-nearest neighbors improves on the two broader datasets but remains sensitive to local sparsity. A single decision tree captures nonlinear thresholds, while random forest and Extra Trees provide more stable estimates through ensembling. Gradient boosting performs strongly on MLP but is less competitive on the high-variance CNN and Transformer distributions under the fixed settings.

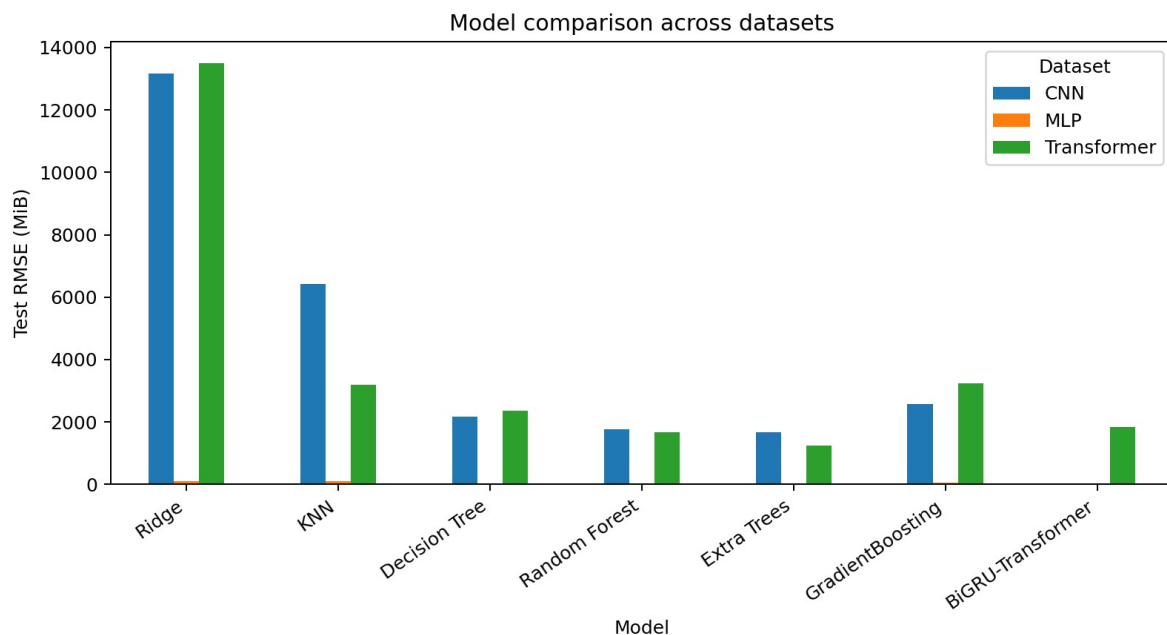


Figure 4. Held-out test RMSE across datasets and regressors.

4.3 Transformer-Focused Analysis

Figure 5 shows the BiGRU-Transformer test predictions, whereas Figure 6 shows the Extra Trees predictions. Both follow the identity line over most of the range, but Extra Trees is visibly tighter in the middle and upper regions. The neural model shows a modest conservative bias in several low- and mid-memory cases, which can reduce packing efficiency but is generally less damaging than severe underestimation when out-of-memory failures are costly.

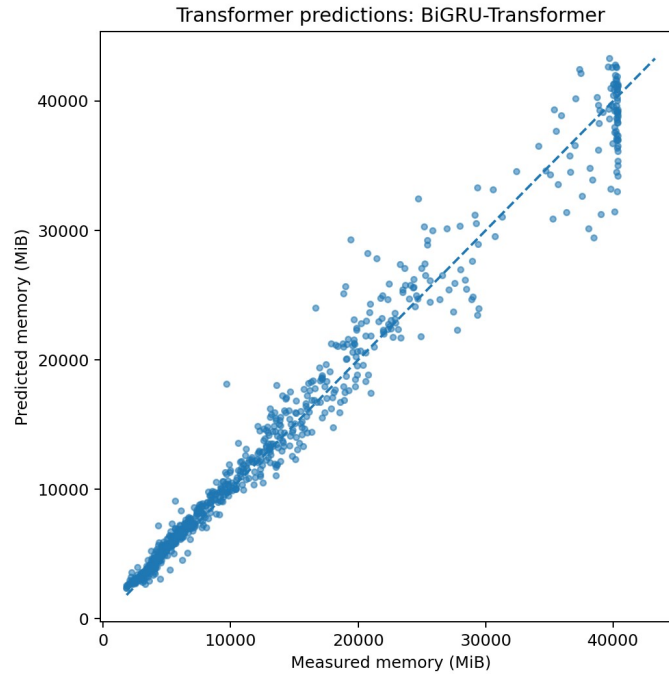


Figure 5. BiGRU-Transformer predictions on the Transformer test set.

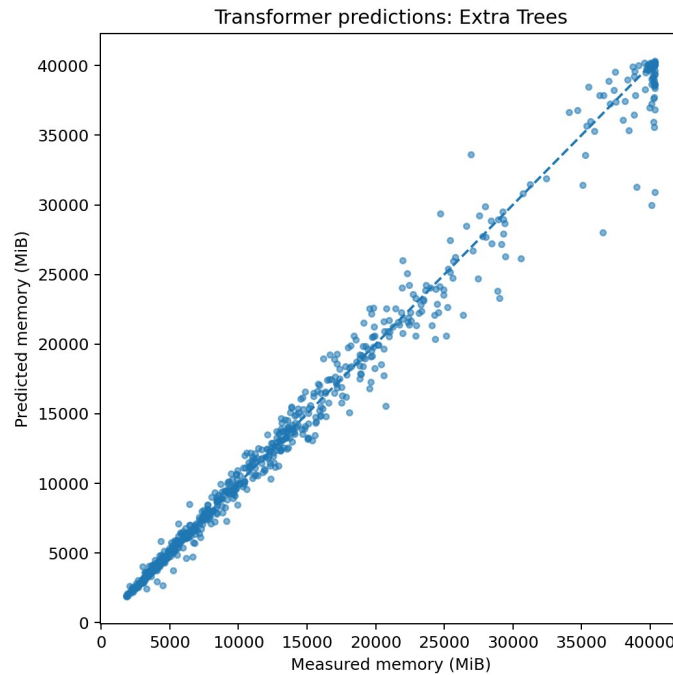


Figure 6. Extra Trees predictions on the Transformer test set.

Table 8 isolates the Transformer comparison. Random forest is second by RMSE at 1,661.2 MiB, and the BiGRU-Transformer follows at 1,827.7 MiB. The neural model substantially outperforms Ridge, K-nearest neighbors, decision tree, and gradient boosting in RMSE, while placing 83.8% of predictions within 2 GB. Its fit

time is higher than the classical models but remains small relative to the duration and cost of launching a training workload.

Table 8. Transformer-only model comparison.

Dataset	Model	MAE (MiB)	RMSE (MiB)	R2	MAPE (%)	8-GB bin acc.	Within 2 GB	Fit time (s)
Transformer	Ridge	5,747.8	13,508.4	-0.385	36.930	0.626	0.448	0.010
Transformer	KNN	1,699.9	3,192.2	0.923	14.620	0.868	0.781	0.010
Transformer	Decision Tree	1,237.6	2,350.7	0.958	9.410	0.888	0.815	0.030
Transformer	Random Forest	927.906	1,661.2	0.979	6.790	0.934	0.894	0.690
Transformer	Extra Trees	675.819	1,248.8	0.988	4.970	0.947	0.931	0.600
Transformer	GradientBoosting	1,811.4	3,246.1	0.920	12.630	0.826	0.775	0.380
Transformer	BiGRU-Transformer	1,191.3	1,827.7	0.975	10.300	0.866	0.838	21.580

4.4 Error Distribution and Ablation

Table 9 and Figure 7 examine the BiGRU-Transformer by true memory range. MAE is 407.1 MiB for 0-4 GB and 699.6 MiB for 4-8 GB. It rises to 1,828.9 MiB for 16-24 GB and 2,668.3 MiB for 24-32 GB, then reaches 2,483.4 MiB for 32-40 GB. The upper ranges contain fewer cases and many observations near the device ceiling, so small shifts in high-memory behavior have a larger effect on RMSE. The mean prediction in the 32-40 GB range is 38,164.8 MiB versus a mean target of 39,277.1 MiB, revealing a moderate underestimation tendency at the extreme end.

Table 9. BiGRU-Transformer error by true Transformer memory range.

True memory bin	Rows	MAE (MiB)	RMSE (MiB)	Mean true (MiB)	Mean predicted (MiB)
0-4GB	137	407.095	486.906	3,146.9	3,490.1
4-8GB	178	699.602	854.062	5,875.4	6,452.3
8-16GB	207	994.084	1,363.4	12,061.0	12,510.6
16-24GB	104	1,828.9	2,498.1	20,159.7	21,245.7
24-32GB	36	2,668.3	3,219.0	27,593.1	27,603.4
32-40GB	90	2,483.4	3,236.6	39,277.1	38,164.8

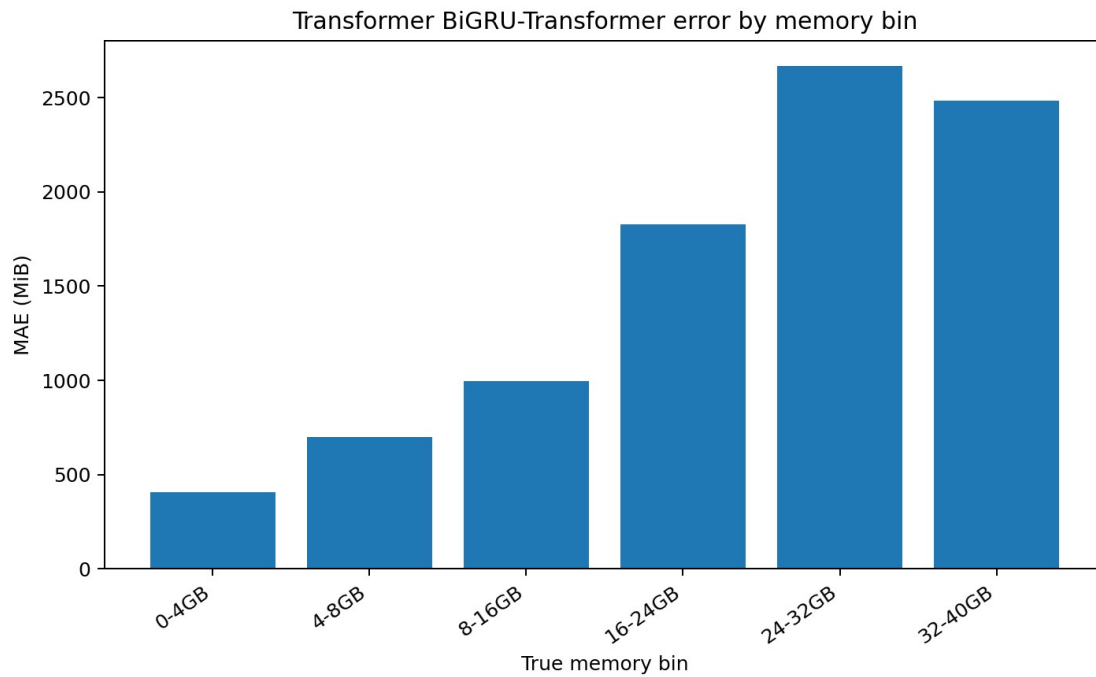


Figure 7. BiGRU-Transformer MAE by true Transformer memory bin.

Table 10 reports the focused ablations. Removing accumulated parameters, total activations, and total parameters from gradient boosting increases RMSE from 3,246.1 to 5,441.1 MiB, confirming that aggregate size descriptors carry much of the signal. Using the raw target improves gradient-boosting RMSE to 2,656.6 MiB in this setting, showing that the preferred target transform depends on the learner. Neither change closes the gap to random forest or Extra Trees. The comparison also clarifies the role of the neural architecture: it supplies a coherent feature-interaction model, but the current aggregate table does not fully exploit its capacity to process a natural layer sequence.

Table 10. Transformer ablations and baseline comparisons.

Variant	Change	MAE (MiB)	RMSE (MiB)	R2	8-GB bin acc.
Full BiGRU-Transformer	None	1,191.3	1,827.7	0.975	0.866
Extra Trees baseline	Sequence encoder not used	675.819	1,248.8	0.988	0.947
Random Forest baseline	Sequence encoder not used	927.906	1,661.2	0.979	0.934
GradientBoosting baseline	Sequence encoder not used	1,811.4	3,246.1	0.920	0.826
KNN baseline	Sequence encoder not used	1,699.9	3,192.2	0.923	0.868
Ridge baseline	Sequence encoder not used	5,747.8	13,508.4	-0.385	0.626
GradientBoosting without size totals	Removed Accumulated Params, Total Activations, Total Parameters	3,827.4	5,441.1	0.775	0.610
GradientBoosting raw target	No log1p target transform	1,719.7	2,656.6	0.946	0.852

4.5 Learning Behavior and Feature Diagnostics

Figure 8 shows that Transformer validation loss decreases steadily during the 35 training epochs and then stabilizes, indicating a well-behaved optimization trajectory. The curve supports the use of the final neural predictions in the model comparison and does not show the abrupt divergence that would suggest unstable training.

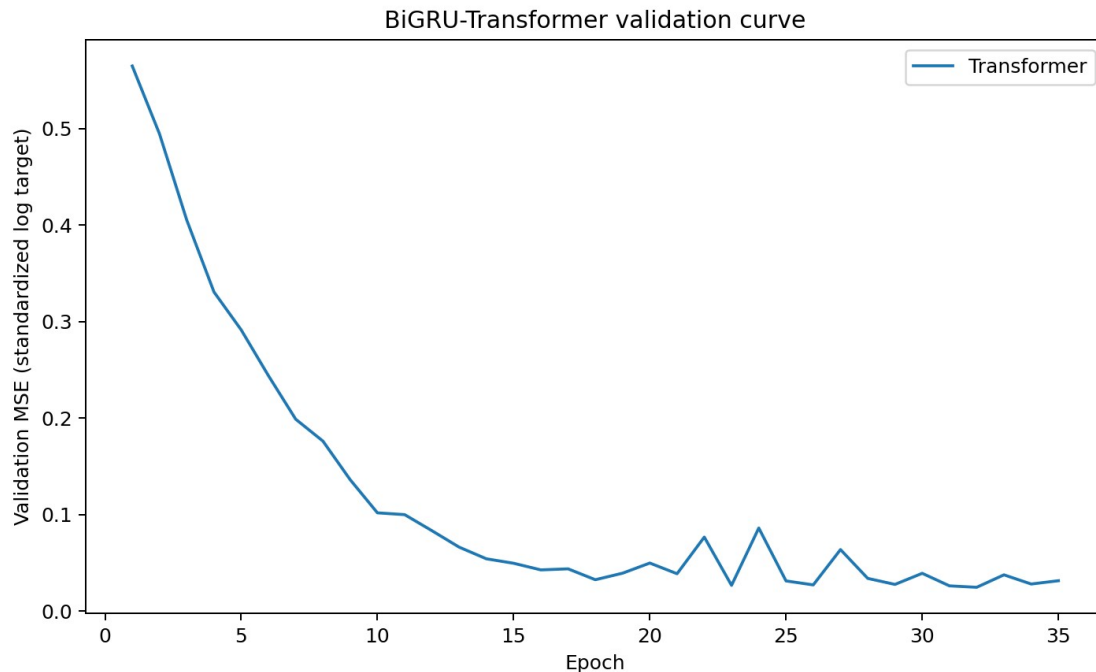


Figure 8. Transformer validation loss across BiGRU-Transformer training epochs.

Table 11 lists the leading Extra Trees importances, and Figure 9 visualizes the same ranking. Accumulated Params, Total Activations, and Embedding Size contribute 0.246, 0.225, and 0.224, respectively; Total Parameters and Batch Size follow at 0.125 and 0.115. Sequence length, depth, and layer count have smaller marginal importance after the aggregate totals are included. This pattern is consistent with Transformer memory structure: embedding width expands projection and feed-forward matrices, while parameter and activation totals already summarize many effects of depth, width, and sequence processing.

Table 11. Top Transformer feature importances from the Extra Trees baseline.

Feature	Extra Trees importance
Accumulated Params	0.246
Total Activations	0.225
Embedding Size	0.224
Total Parameters	0.125
Batch Size	0.115
Seq Length	0.014
Depth	0.009
Num Layers	0.009

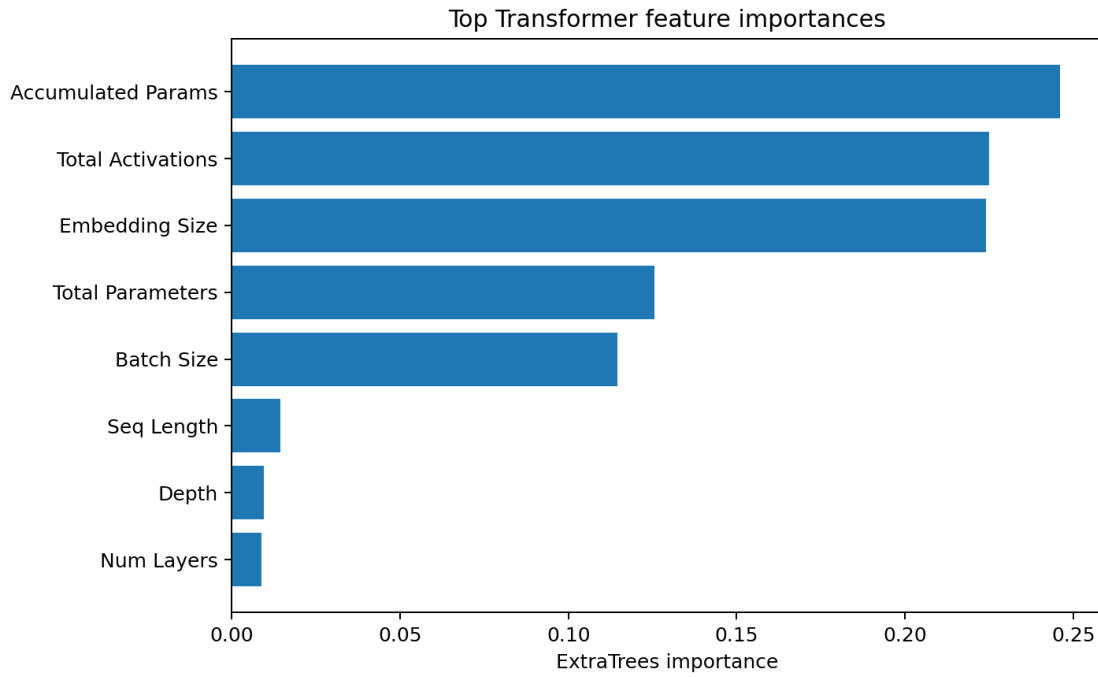


Figure 9. Extra Trees feature importance for Transformer workload descriptors.

4.6 Operational Interpretation

The distinction between exact MiB regression and capacity-band decisions is important in production. Extra Trees is strongest on both objectives, while the BiGRU-Transformer still assigns 86.6% of Transformer cases to the correct 8-GB band and places 83.8% within 2 GB. Such estimates can feed risk-bounded oversubscription [39], profit-aware admission control [43], and power-aware inventory planning [44]. In a human-in-the-loop setting, the predicted band, uncertainty margin, and dominant workload factors can be displayed through a capacity brief or incident card [46], [52].

The results also indicate where a recurrent-attention architecture is most likely to add value. The present CSV represents a workload with aggregate columns rather than a full ordered layer trace. Extra Trees can exploit this representation efficiently through threshold splits. A future input that tokenizes layer type, activation volume, parameter count, precision, and ordering would align more directly with the BiGRU-Transformer design and could improve transfer to architectures not captured by aggregate totals.

5. Limitations

The evaluation is limited to the three GPUMemNet CSV files and does not establish universal performance across GPU generations, framework versions, precision modes, optimizers, or distributed-training configurations. Framework caching, kernel workspaces, fused operators, activation checkpointing, and optimizer-state layouts can change memory use even when high-level descriptors are similar [8]-[12]. Hardware and topology distribution shift is especially important for cross-cloud deployment [38], [47].

Positive-target out-of-memory rows are retained, but their recorded peaks may be censored near the device limit rather than equal to the memory required for successful completion. This can compress the upper tail and makes exact regression in the highest memory range harder. Capacity-bin accuracy partly reduces sensitivity to this issue, but a deployment-oriented model should distinguish completed runs from censored observations and use asymmetric costs when underestimation is more harmful than overestimation.

The BiGRU-Transformer receives ordered tokens derived from aggregate predictors rather than a complete layer-by-layer sequence. This provides a consistent comparison with the classical models but underuses the recurrent and attention components. Future work should parse the raw layer representation and include layer type, tensor dimensions, precision, checkpointing state, optimizer choice, and device characteristics. It should

also evaluate calibration and distribution-free prediction intervals so that the scheduler can choose a reservation according to an explicit risk tolerance [37], [39].

Hyperparameters are fixed rather than selected through a broad search. Random search or Bayesian optimization could improve several models [23], and repeated cross-validation would provide a fuller estimate of split variability. The current fixed protocol is appropriate for direct comparison, but model-selection conclusions should be revisited when richer features, new hardware, or additional workload families become available.

6. Conclusion

This study evaluated GPU memory requirement regression on the GPUMemNet MLP, CNN, and Transformer datasets using pre-execution architecture descriptors. The protocol removes non-positive targets, excludes post-launch status and utilization fields, applies a fixed quantile-stratified 70/15/15 split with seed 42, and compares six classical regressors with a BiGRU-Transformer on Transformer workloads.

The BiGRU-Transformer achieves 1,191.3 MiB MAE, 1,827.7 MiB RMSE, $R^2 = 0.975$, and 86.6% 8-GB bin accuracy. Extra Trees is the strongest model on the current tabular representation, with RMSE values of 28.0 MiB on MLP, 1,672.2 MiB on CNN, and 1,248.8 MiB on Transformer. Feature diagnostics show that accumulated parameters, total activations, embedding size, total parameters, and batch size explain most of the Transformer signal.

The findings support two practical conclusions. First, strong tree ensembles should be a standard baseline whenever architecture descriptors already contain aggregate size information. Second, the BiGRU-Transformer provides a viable sequence-aware framework whose main opportunity lies in richer layer-level inputs. Coupling such predictions with calibrated admission policies and clear operator-facing explanations can improve accelerator utilization while controlling out-of-memory risk.

References

- [1] A. Vaswani et al., "Attention is all you need," in Proc. Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.
- [2] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in Proc. Conf. Empirical Methods in Natural Language Processing, 2014, pp. 1724-1734.
- [3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv:1412.3555, 2014.
- [4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
- [5] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019, pp. 4171-4186.
- [6] T. B. Brown et al., "Language models are few-shot learners," in Proc. Advances in Neural Information Processing Systems, 2020, pp. 1877-1901.
- [7] J. Kaplan et al., "Scaling laws for neural language models," arXiv:2001.08361, 2020.
- [8] D. Narayanan et al., "Efficient large-scale language model training on GPU clusters using Megatron-LM," in Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis, 2021, pp. 1-15.
- [9] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He, "ZeRO: Memory optimizations toward training trillion parameter models," in Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis, 2020, pp. 1-16.
- [10] T. Chen, B. Xu, C. Zhang, and C. Guestrin, "Training deep nets with sublinear memory cost," arXiv:1604.06174, 2016.
- [11] P. Micikevicius et al., "Mixed precision training," in Proc. Int. Conf. Learning Representations, 2018.
- [12] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library," in Proc. Advances in Neural Information Processing Systems, 2019, pp. 8024-8035.
- [13] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
- [14] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5-32, 2001.
- [15] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," Machine Learning, vol. 63, no. 1, pp. 3-42, 2006.
- [16] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Annals of Statistics, vol. 29, no. 5, pp. 1189-1232, 2001.

- [17] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970.
- [18] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175-185, 1992.
- [19] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Mateo, CA, USA: Morgan Kaufmann, 1993.
- [20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations*, 2015.
- [21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proc. Int. Conf. Learning Representations*, 2019.
- [22] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proc. Int. Joint Conf. Artificial Intelligence*, 1995, pp. 1137-1143.
- [23] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281-305, 2012.
- [24] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533-536, 1986.
- [25] E. Yousefzadeh-Asl-Miandoab, R. Karimzadeh, D. Yorulmaz, B. Ibragimov, and P. Tozun, "GPU memory and utilization estimation for training-aware resource management: Opportunities and limitations," in *Proc. EuroMLSys*, 2026, pp. 127-138, doi: 10.1145/3805621.3807621.
- [26] Y. Gao, Y. Liu, H. Zhang, Z. Li, Y. Zhu, H. Lin, and M. Yang, "Estimating GPU memory consumption of deep learning models," in *Proc. ESEC/FSE*, 2020, pp. 1342-1352.
- [27] G. Yeung, D. Borowiec, R. Yang, A. Friday, R. Harper, and P. Garraghan, "Horus: Interference-aware and prediction-based scheduling in deep learning systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 1, pp. 88-100, 2022, doi: 10.1109/TPDS.2021.3079202.
- [28] T. Kim, Y. Wang, V. Chaturvedi, L. Gupta, S. Kim, Y. Kwon, and S. Ha, "LLMem: Estimating GPU memory usage for fine-tuning pre-trained LLMs," *arXiv:2404.10933*, 2024.
- [29] C. Wang, Z. Wen, R. Zhang, P. Xu, and Y. Jiang, "GPU Memory Requirement Prediction for Deep Learning Task Based on Bidirectional Gated Recurrent Unit Optimization Transformer," in *2025 5th International Conference on Artificial Intelligence, Virtual Reality and Visualization (AIVRV)*, Chengdu, China, 2025, doi: 10.1109/AIVRV67401.2025.11350369.
- [30] Y. Gao, X. Gu, H. Zhang, H. Lin, and M. Yang, "Runtime performance prediction for deep learning models with graph neural network," in *Proc. ICSE-SEIP*, 2023, pp. 368-380.
- [31] W. Xiao et al., "Gandiva: Introspective cluster scheduling for deep learning," in *Proc. 13th USENIX OSDI*, 2018, pp. 595-610.
- [32] D. Narayanan, K. Santhanam, F. Kazhamiaka, A. Phanishayee, and M. Zaharia, "Heterogeneity-aware cluster scheduling policies for deep learning workloads," in *Proc. 14th USENIX OSDI*, 2020, pp. 481-498.
- [33] F. Strati, X. Ma, and A. Klimovic, "Orion: Interference-aware, fine-grained GPU sharing for ML applications," in *Proc. EuroSys*, 2024, pp. 1075-1092.
- [34] T. Robroek et al., "An analysis of collocation on GPUs for deep learning training," in *Proc. Workshop on Machine Learning and Systems*, 2024, pp. 81-90.
- [35] Q. Weng et al., "MLaaS in the wild: Workload analysis and scheduling in large-scale heterogeneous GPU clusters," in *Proc. 19th USENIX NSDI*, 2022, pp. 945-960.
- [36] E. Yousefzadeh-Asl-Miandoab, T. Robroek, and P. Tozun, "Profiling and monitoring deep learning training tasks," in *Proc. EuroMLSys*, 2023, pp. 18-25, doi: 10.1145/3578356.3592589.
- [37] S. He, H. Tu, and I. Liu, "Safe PD capacity forecasting with time-series foundation models and calibrated uncertainty for heterogeneous GPU clusters," *Journal of Advanced Computing Systems*, vol. 3, no. 4, pp. 48-66, 2023, doi: 10.69987/JACS.2023.30404.
- [38] S. He, X. Chang, and E. Sun, "Cross-cloud transfer learning for AI training capacity forecasting under workload and topology distribution shift," *Journal of Advanced Computing Systems*, vol. 4, no. 1, pp. 100-120, 2024, doi: 10.69987/JACS.2024.40108.
- [39] S. Chen, S. He, and E. Sun, "Risk-bounded GPU resource oversubscription via conformal demand envelopes in production AI clusters," *Journal of Advanced Computing Systems*, vol. 4, no. 5, pp. 119-134, 2024, doi: 10.69987/JACS.2024.40509.
- [40] H. Tu, S. Zhao, and S. He, "LLM-augmented salable GPU supply forecasting for disaggregated recommendation serving: Predicting instance readiness, scheduling delay, and capacity risk," *Journal of Advanced Computing Systems*, vol. 4, no. 9, pp. 85-102, 2024, doi: 10.69987/JACS.2024.40908.

- [41] S. Zhao, J. Bai, and D. Roberson, "Multi-horizon GPU demand forecasting with workload semantics and operational risk curves: An empirical study on Alibaba Clusterdata GPU Trace," *Journal of Technology Informatics and Engineering*, vol. 4, no. 3, pp. 544-571, 2025, doi: 10.51903/jtie.v4i3.498.
- [42] S. He and A. Qian, "Few-shot cold-start workload forecasting for new AI inference tenants with time-series foundation models," *Journal of Technology Informatics and Engineering*, vol. 4, no. 1, pp. 306-324, 2025, doi: 10.51903/jtie.v4i1.546.
- [43] S. Zhao, Y. Ren, and X. Chang, "Profit-aware spot GPU admission control with cost-sensitive loss and evidence-grounded policy memos for AI workload supply-demand matching," *Journal of Technology Informatics and Engineering*, vol. 5, no. 2, pp. 45-59, 2026, doi: 10.51903/jtie.v5i2.545.
- [44] S. He, "Power-aware inventory planning for AI infrastructure using job-level forecasting and LLM workload explanations," *Journal of Technology Informatics and Engineering*, vol. 5, no. 1, pp. 341-359, 2026, doi: 10.51903/jtie.v5i1.548.
- [45] J. Nie and D. Zheng, "Noisy-neighbor-aware VM degradation risk modeling with unsupervised residual fusion," *Journal of Advanced Computing Systems*, vol. 4, no. 4, pp. 112-123, 2024, doi: 10.69987/JACS.2024.40409.
- [46] G. Liu, S. He, and H. Wong, "LLM-compatible visual brief cards for AI infrastructure capacity dashboards: A UI/UX framework for turning forecast risk into graphic design decisions," *International Journal of Graphic Design*, vol. 3, no. 1, pp. 196-213, 2025, doi: 10.51903/ijgd.v3i1.3723.
- [47] Q. Xin, "Hybrid cloud architecture for efficient and cost-effective large language model deployment," *Journal of Information Systems and Informatics*, vol. 7, no. 3, pp. 2182-2195, 2025, doi: 10.51519/journalisi.v7i3.1170.
- [48] G. Liu, S. He, and I. Liu, "LLM-augmented multi-source root cause attribution for CPU and network faults in microservices," *Journal of Advanced Computing Systems*, vol. 3, no. 6, pp. 39-57, 2023, doi: 10.69987/JACS.2023.30604.
- [49] J. Nie and D. Zheng, "Ambiguity-aware HDFS log anomaly detection with retrieval-augmented failure narratives and selective refusal," *Journal of Advanced Computing Systems*, vol. 3, no. 1, pp. 66-80, 2023, doi: 10.69987/JACS.2023.30105.
- [50] B. Zhang, H. Rao, and D. Zhao, "Evidence-grounded RAG for cloud-native DevOps: Hallucination-resistant AIOps question answering over private operations documents," *Journal of Advanced Computing Systems*, vol. 4, no. 3, pp. 109-125, 2024, doi: 10.69987/JACS.2024.40308.
- [51] G. Liu, C. Li, and E. Zhang, "OpsLLM for cloud incident triage: Bilingual RAG-based root cause analysis and alert summarization for AI infrastructure operations," *Journal of Advanced Computing Systems*, vol. 4, no. 4, pp. 97-111, 2024, doi: 10.69987/JACS.2024.40408.
- [52] J. Nie, G. Liu, C. Li, and T. Zou, "Evidence-constrained incident visualization cards for distributed cloud logs: A UI/UX framework for turning Hadoop, OpenStack, and ZooKeeper logs into actionable SRE design interfaces," *International Journal of Graphic Design*, vol. 4, no. 1, pp. 179-185, 2026, doi: 10.51903/ijgd.v4i1.3703.
- [53] Q. Xin, "Self-supervised log anomaly detection with LogBERT-style transformers: Full empirical evaluation on a reproducible SynHDFS benchmark," *Journal of Electrical Engineering and Computer Science*, vol. 11, no. 1, pp. 23-35, 2026, doi: 10.54732/jeeecs.v11i1.3.
- [54] Q. Xin, "Log anomaly detection with conformal alert control and evidence-grounded incident ticket generation," *AVITEC*, vol. 8, no. 2, p. 247, 2026, doi: 10.28989/avitec.v8i2.3974.