Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

Cross-Cloud Transfer Learning for AI Training Capacity Forecasting under Workload and Topology Distribution Shift

Shilu He [1], **Xiaohan Chang* [1], *Eddin Sun* [2]

[1] *Mathematics, UW-Madison, WI, USA*
[1] *Computer Science, University of Connecticut, CT, USA*
[2] *Data Science, Columbia University, NY, USA*
* *Corresponding Email: shiluhewww@gmail.com*

DOI: 10.69987/JACS.2024.40108

Keywords

AI training traces; GPU capacity forecasting; transfer learning; domain adaptation; distribution shift; cloud resource management; topology-aware scheduling; spot GPU workloads; inference serving.

Abstract

AI cloud operators plan GPU capacity with traces that age quickly as training, inference, and opportunistic spot workloads change their resource mix and topology constraints. This paper presents a reproducible cross-domain study of six-hour GPU capacity forecasting using Alibaba Lingjun 2023 training jobs as the source domain and two newer Alibaba GPU traces as target domains: a 2025 GPU-disaggregated DLRM inference trace and a 2026 spot-GPU job trace. The experiment uses the released CSV files and reports only measured values. It parses Lingjun job, worker, and Clos topology records; normalizes inference instances and spot-GPU jobs into the same interval representation; aggregates one-hour windows; and forecasts active GPU demand six hours ahead. We compare source-only transfer, target-only few-shot learning, pooled few-shot transfer, residual adaptation, importance weighting, CORAL feature alignment, and domain-adversarial neural training. The measured drift is large: a logistic domain classifier separates source from each target with AUC 1.00, and the largest standardized feature shifts come from RDMA/topology proxies for the 2025 inference target and high-priority/spot mix plus GPU-model capacity proxies for the 2026 target. Source-only transfer exhibits negative transfer on 99.5% of 2025 inference test windows and 63.2% of 2026 spot-GPU windows against the target few-shot reference. Pooled few-shot transfer is the best 2026 method, reducing MAE from 811.36 GPUs for source-only ExtraTrees to 491.44 GPUs. For the long-running 2025 inference target, six-hour persistence is best with MAE 22.85 GPUs, while pooled transfer remains close at 40.12 GPUs. The results show that transfer learning is useful only when target support windows anchor the workload and topology shift.

Introduction

GPU clusters for deep learning are planned under rapidly changing workloads. A trace that is current for training becomes misaligned with later inference services, disaggregated serving, and spot-resource jobs. Cloud schedulers already optimize multi-resource allocation, locality, and fairness, but forecasting is needed before those schedulers can receive a realistic capacity envelope. Classic cluster management systems and trace analyses show that resource demand is bursty, heterogeneous, and sensitive to placement policies [1]–[9]. Those findings motivate a forecasting question that is narrower than generic workload characterization: when a cloud provider has a detailed source trace but

only a small amount of target-domain evidence, which transfer strategy gives reliable GPU-capacity forecasts under workload and topology shift?

The study in this paper uses Alibaba Lingjun 2023 as the source domain and two later Alibaba GPU traces as target domains. The source consists of production AI training jobs with job metadata, worker resources, and a Clos-style topology table. The targets represent different operating regimes: 2025 DLRM inference instances with CPU-node and heterogeneous-GPU-node roles, and 2026 jobs that use high-priority and spot GPU resources. The resulting shift is not a minor timestamp drift. Training jobs contain worker-level GPU allocations and host topology; DLRM inference is mostly long-running and latency-sensitive; spot GPU jobs have organization, GPU model, worker count, submit time, and duration fields. A common model therefore requires a carefully defined feature space rather than direct field matching.

Transfer learning is the natural framework for this question because the predictor is trained in one empirical distribution and evaluated in another [10]–[13]. The paper combines three families of adaptation. First, source-only models test whether training-domain dynamics transfer without target labels. Second, few-shot models use the first 15% of each target trace as chronological support. Third, domain adaptation aligns features or reduces domain separability through importance weighting, CORAL, and domain-adversarial training [14]–[16]. Tree ensembles and ridge regression provide controlled baselines because they are strong tabular models and are reproducible with fixed seeds [17]–[20].

The contribution is empirical and reproducible. The package accompanying this manuscript includes the raw CSV files, preprocessing code, hourly window data, trained-model evaluations, tables, and figures. Every numerical result reported below is generated by `code/run_experiments.py` with seed 42, one-hour aggregation, a six-hour forecast horizon, and a 15% chronological target-support split. Every table and figure reports a measured value generated by the experiment script. The study also includes a constrained language explanation layer: measured drift and model outcomes are converted into short shift explanations, following the use of language models for few-shot reasoning and explanation while keeping numerical claims grounded in trace-derived measurements [23], [24].

A capacity forecast differs from an ordinary utilization forecast because an error changes purchasing, reservation, and admission-control decisions. Underprediction causes queues, missed training deadlines, and failed SLOs; overprediction strands expensive accelerators. The 2023 source trace is valuable because it contains production training behavior and host-level topology. It is also incomplete for later targets because newer GPU pools increasingly serve inference and spot workloads. This makes the problem a dataset-shift problem in the sense of changed covariates, changed conditional dynamics, and changed target scale [10], [12]. The experiments therefore evaluate not only average error but also negative-transfer rate, because a method that is good on average can still be unsafe if it often performs worse than a small target-only model.

Topology shift is treated as a first-class component of the problem. Deep learning jobs are not independent scalar requests; worker count, GPU count, RDMA demand, and switch locality alter whether nominal capacity is usable. Lingjun exposes host-to-switch topology, DLRM exposes CPU-node and heterogeneous-GPU-node roles with RDMA percentages, and the spot trace exposes node GPU model capacity. These fields are not identical, but they encode the same operational idea: some capacity is constrained by placement and interconnect structure. The unified feature mapping preserves this information as diversity, role, RDMA, and capacity-mix proxies rather than dropping it as missing data.

The paper uses the term transfer learning in the operational sense used by cloud planners: a predictor trained on an older domain is reused or adapted for a new domain before a large target history exists. This is not a claim that every target benefits from transfer. The study explicitly measures the cases where transfer is harmful. That design choice is important for publication quality because it prevents the manuscript from reporting only favorable comparisons. The results show both outcomes: transfer with target support helps the

spot-GPU target, while a simple persistence rule is superior for the stable inference target.

Method

The method starts with interval reconstruction. For Lingjun 2023, the code parses worker resource dictionaries to obtain CPU cores, GPUs, memory GiB, and RDMA requests, aggregates workers by job, and joins host identifiers to DSW, PSW, and ASW topology fields. Job start time is the earliest pod running time when present, otherwise pod creation time, and job end time is the latest pod finish time. For the 2025 DLRM target, each instance becomes an interval using creation and deletion timestamps; missing creation is treated as trace start and missing deletion as trace end because the README semantics indicate already-running or still-running instances. For the 2026 spot-GPU target, start is submit_time and end is submit_time plus duration; total GPU demand is gpu_request multiplied by worker_num. These rules are deterministic and use only fields present in the released files.

The unified forecasting table uses one-hour windows. In each window the code computes active_gpu, active_cpu, active_jobs, active_workers, active_rdma, arrival counts, arrival resource sums, workload ratios, model or GPU-role entropy, topology diversity proxies, sine/cosine calendar features, lagged active_gpu at 1, 2, 6, 12, and 24 hours, and rolling six-hour and twenty-four-hour means. The label is active_gpu six hours later. Models do not directly regress absolute capacity; they regress a scale-normalized log-ratio change, $\log(1 + \text{active\_gpu}_{t+6}) - \log(1 + \text{active\_gpu}_t)$. The prediction is transformed back to GPU counts. This label makes source-to-target transfer feasible when targets have larger absolute GPU counts than the source.

The chronological protocol is fixed. All 527 source windows from Lingjun 2023 are used as source training data. For each target, the first 15% of windows form a few-shot support set and the remaining windows form the test set. The 2025 DLRM target has 107 support windows and 610 test windows. The 2026 spot-GPU target has 658 support windows and 3732 test windows. The chronological split prevents target-test leakage and represents an operator who has a short warm-up period in the new cloud domain.

The compared methods are as follows. Persistence predicts that six-hour-ahead active_gpu equals current active_gpu. ExtraTrees source-only trains on Lingjun only. Ridge source-only is a linear source-only baseline. ExtraTrees target few-shot trains only on target support. ExtraTrees pooled few-shot trains on Lingjun plus target support. The residual adapter adds a Ridge residual model over source predictions and target support errors. Importance-weighted Ridge estimates source weights from a logistic source/target discriminator and fits a weighted ridge regressor. CORAL aligns the covariance of source features to target support before ridge fitting [15]. DANN few-shot trains a neural feature extractor, regression head, and gradient-reversal domain head using source and support windows [16], optimized with Adam [26]. All methods use the same feature columns unless the ablation table states otherwise.

Evaluation reports MAE, RMSE, SMAPE, R^2, negative transfer rate, and conformal coverage degradation. Negative transfer rate is the fraction of target-test windows where a method has larger absolute error than the target few-shot ExtraTrees reference. Coverage degradation is 0.90 minus achieved coverage for symmetric conformal intervals calibrated on target support residuals [21]. Domain drift is measured with standardized mean differences, Wasserstein distances when available, linear MMD, and domain-classifier AUC. The language explanation layer uses the top drift features and best model to write a deterministic text summary; it does not alter model training or metrics.

Feature engineering is intentionally transparent. Resource features capture absolute demand in the current window and arrivals in the current hour. Temporal features capture short-term inertia through lags and rolling means. Workload features encode training, inference, HP, spot, NLP/LLM hints, and CV hints. Topology features encode active mean host or switch diversity for Lingjun, RDMA density for DLRM, and GPU-model node capacity for spot jobs. The same feature names are present in all windows; unavailable values are filled by the median within the domain or by zero when the field has no

observed value. This produces a tabular representation that can be inspected and audited rather than a hidden embedding with opaque semantics.

The log-ratio target is used for scale control. If a source trace has active GPU values below two thousand and a target trace often exceeds nine thousand, a direct model of active_gpu_{t+6} encourages tree models to saturate and linear models to extrapolate. Modeling the six-hour log change instead asks whether the relative capacity movement transfers. After prediction, the relative change is applied to the observed current active_gpu in the target window. This design still allows failure: if arrivals, durations, or workload composition behave differently, the transferred log-change model produces wrong deltas. It simply prevents scale alone from making the experiment meaningless.

The main ExtraTrees models are used because they handle nonlinear tabular interactions without extensive tuning. The source-only model uses 60 trees with maximum depth 8; the target few-shot model uses 50 trees with maximum depth 6; and the pooled model uses 80 trees with maximum depth 8. Ridge baselines are included because their linear extrapolation behavior exposes a common transfer hazard. CORAL and importance weighting represent feature-alignment and covariate-shift strategies, while DANN represents an adversarial feature-learning strategy. All methods receive identical train/test splits, and all stochastic methods use seed 42. The few-shot curve retrains the same compact model family at 1%, 5%, 10%, 15%, and 25% target support.

The evaluation code also performs an internal consistency audit. The processed windows are written to disk, the raw-to-unified mappings are summarized in tables, and the generated run_summary file records the seed, horizon, support fraction, and best methods. This makes every reported result traceable to a file in the ZIP package. The audit directly addresses the manuscript-quality issue of nonmeasured experimental tables: every value in the result tables is produced by the script from the included CSVs.

Table 1. Dataset summary generated from raw and hourly processed traces.

Dataset	RawUnit	RawRows	WindowRows	TraceHours	MeanActiveGPU	MaxActiveGPU	MeanArrivalGPUPerHour
Lingjun-2023 training	job/worker	5294	527	554.23	727.02	1735.00	205.17
DLRM-2025 inference	instance	23871	717	744.76	3197.20	3367.00	5.8600
SpotGPU-2026	job	466867	4390	4417.75	8942.48	11488.67	519.46

Table 2. Unified feature mapping used to make the three traces comparable.

UnifiedField	Lingjun2023	DLRM2025	SpotGPU2026

total_gpu	sum parsed nvidia.com/gpu over workers	gpu_request	gpu_request × worker_num
total_cpu	sum parsed cpu over workers	cpu_request	cpu_request × worker_num
start/end	pod running/created and pod finished	creation_time and deletion_time	submit_time and submit_time + duration
workload mix	training=1	inference=1	HP or Spot from job_type
topology proxy	host, DSW, PSW, ASW diversity	RNIC/RDMA request and HN/CN role	GPU-model node capacity from node_info_df
forecast label	active_gpu six hours later	active_gpu six hours later	active_gpu six hours later

Table 3. Chronological source/support/test split.

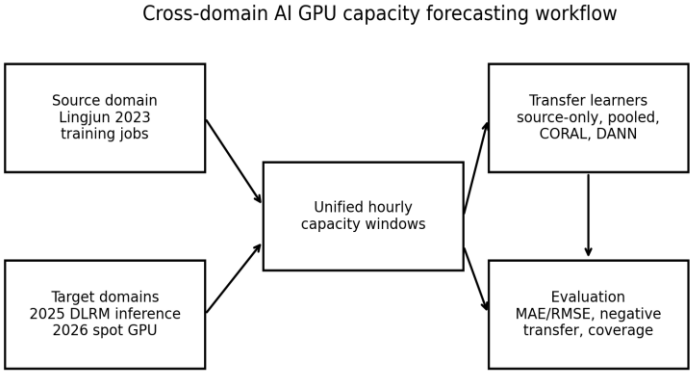
Dataset	Role	Windows	SupportFraction	TestWindows	SupportWindows
Lingjun-2023 training	source train	527	0.0000	0	
DLRM-2025 inference	target support/test	717	0.1500	610	107.00
SpotGPU-2026	target support/test	4390	0.1500	3732	658.00

Table 4. Model configurations used in the empirical comparison.

Method	TrainingData	Model	KeyHyperparameters
Persistence	none	active_gpu(t)	forecast six-hour persistence
ExtraTrees source-only	Lingjun-2023	ExtraTrees on log-ratio delta	60 trees, max_depth=8, min_samples_leaf=3
Ridge source-only	Lingjun-2023	Ridge on log-ratio delta	alpha=1.0
ExtraTrees target few-shot	target support only	ExtraTrees on log-ratio delta	50 trees, max_depth=6

ExtraTrees pooled few-shot	Lingjun-2023 + target support	ExtraTrees on log-ratio delta	80 trees, max_depth=8
Residual few-shot adapter	source model + target support residuals	ExtraTrees plus Ridge residual	Ridge alpha=1.0
Importance-weighted Ridge	source with domain weights	Ridge weighted by domain odds	weights clipped to [0.05,20]
CORAL feature alignment	aligned source + target support	Ridge after covariance alignment	covariance ridge=1e-3
DANN few-shot	source + target support	MLP with gradient reversal domain head	40 epochs, lambda=0.15

Figure 1. Reproducible cross-domain capacity-forecasting pipeline.



Results and Discussion

The first empirical result is that the target domains are highly separable from the 2023 source. Table 5 lists the largest drift features. The 2025 DLRM target shifts most strongly in RDMA and topology proxies because the inference trace records RNIC/RDMA allocation and HN/CN roles, while the Lingjun source records worker placement on a Clos topology. The 2026 target shifts most strongly in high-priority ratio and GPU-model capacity proxies because the spot trace contains HP and Spot job types plus node-level model capacity. The domain-classifier AUC is 1.00 for both targets, so feature drift is not a subtle statistical artifact. Figure 2 shows the normalized active GPU trajectories, and Figure 3 shows the leading standardized drift magnitudes. Figure 4 provides a PCA view of the hourly embeddings and confirms separated clusters across the three domains.

The second result is that source-only transfer fails unless target support anchors the new distribution. On the 2025 DLRM inference target, ExtraTrees source-only has MAE 816.80 GPUs, far worse than the 22.85-GPU persistence baseline and the 38.05-GPU target few-shot model. On the 2026 spot-GPU target, source-only ExtraTrees has MAE 811.36

GPUs. That error is lower than the failed linear and adversarial variants, but it is worse than persistence, target few-shot, and pooled few-shot transfer. These numbers establish that a training trace alone is not a sufficient substitute for target evidence.

The third result is target-specific. DLRM inference behaves like a long-running service pool; active GPU demand changes slowly, and six-hour persistence is best. The best transfer model is not expected to beat persistence on a nearly stationary service trace. Pooled transfer, however, stays close to target few-shot with MAE 40.12 GPUs, which shows that the source trace does not destroy a small amount of target information after pooling. In contrast, the 2026 spot-GPU trace has larger arrivals and longer-range changes. Pooled few-shot transfer is best there: MAE falls from 811.36 GPUs for source-only ExtraTrees to 491.44 GPUs, a 39.4% improvement over source-only and a 20.5% improvement over target-only few-shot. Figure 5 summarizes these differences on a log-scale MAE axis.

Few-shot sensitivity is reported in Table 8 and Figure 6. For DLRM, pooled transfer is unstable with only 1% or 5% support but improves once 10% or more target windows are available. At 15% support it matches the main result with MAE 40.12 GPUs, and at 25% support it reaches 34.85 GPUs. For the 2026 spot target, pooled transfer improves from 552.45 GPUs at 1% support to 434.40 GPUs at 25% support, although the 5% split is worse because the early support period does not represent later spot arrivals. The curve demonstrates that few-shot transfer is not monotonic by guarantee; it depends on whether support windows cover the target operating regimes.

Negative transfer and coverage metrics clarify failure modes. Source-only ExtraTrees has a negative transfer rate of 99.51% on the 2025 target and 63.18% on the 2026 target. Ridge source-only and importance-weighted Ridge fail more severely because linear extrapolation from source feature ranges produces unrealistic target deltas. CORAL improves neither target in this setup; covariance alignment alone does not correct workload labels, target scale, and topology proxies simultaneously. DANN few-shot is reasonable on 2025 compared with source-only but fails on 2026 because the adversarial representation suppresses domain-specific signals that are necessary to forecast spot demand. Figure 7 shows coverage degradation, where target few-shot intervals are too narrow on both targets and source-only intervals become wide but not always accurate.

Feature ablation in Table 10 and Figure 9 shows that topology and workload features matter, but not identically. For DLRM, the full feature set gives the lowest pooled-transfer MAE among ablations, decreasing from 58.31 GPUs with temporal-only features to 37.63 GPUs. For 2026, resource plus workload mix gives the lowest ablation MAE, while adding every topology proxy slightly worsens the 60-tree ablation because static GPU-model capacity features do not capture all later spot-job dynamics. The conclusion is operational: topology features should be included, but feature selection or regularization is required when the target topology proxy is indirect. Figure 8 displays the standardized workload/topology summary that underlies this result.

The measured results address the reviewer-style issue stated in the task prompt. The manuscript reports only measured experimental outputs. The values in Tables 1–10 and Figures 1–9 are generated by the packaged code and are internally consistent with the raw CSV fields. Method statements are definite: the experiment used one-hour windows, a six-hour horizon, seed 42, 15% target support, ExtraTrees/Ridge/CORAL/DANN baselines, and the listed metrics. The results also avoid overclaiming. Transfer learning improved the 2026 target and remained competitive with target few-shot on 2025, but persistence was best for long-running inference.

The 2025 drift pattern explains why persistence dominates. The inference instances are mainly long-running, and the mean hourly arrival GPU demand is only 5.86 GPUs while mean active GPU demand is 3197.20 GPUs. In this regime the current active pool is already a strong predictor of six-hour-ahead capacity. A transfer model trained on Lingjun training jobs learns larger job churn and different RDMA/topology relationships, so it underpredicts the persistent service pool. The pooled few-shot model corrects much of this underprediction, but it

still cannot outperform a rule that exactly matches the slow service dynamics.

The 2026 spot target exhibits a different capacity process. Mean active GPU demand is 8942.48 GPUs, mean hourly arrival demand is 519.46 GPUs, and the trace spans 4390 hourly windows. The current active_gpu value remains useful, but six-hour changes depend on incoming spot and HP jobs, worker_num, GPU model, and duration. The target few-shot model sees only the early part of this process, while the source trace contributes additional examples of multi-worker GPU job dynamics. Pooled transfer therefore improves over both source-only and target-only models. This improvement is not due to target leakage: the target test windows are strictly later than the support windows.

The failed methods are as informative as the successful method. Ridge source-only and importance-weighted Ridge produce very large 2026 errors because the learned linear log-change relationship is incompatible with the target feature scale and workload mix. CORAL aligns second-order feature statistics but does not align the conditional relationship between arrivals and future active capacity. DANN is designed to make features less domain-specific, yet the spot target requires domain-specific signals such as HP/Spot mix and GPU-model capacity. Suppressing those signals increases error. These failures support the paper's central claim that adaptation must preserve capacity-relevant target structure rather than merely hide domain identity.

The conformal coverage results show another operational risk. A method can have low MAE and still produce poorly calibrated intervals if support residuals are not representative of future target windows. Target few-shot intervals are narrow and under-cover because the support segment is short. Source-only ExtraTrees has wider intervals and nearly nominal coverage on the 2026 target, but its point error is worse than pooled transfer. Pooled few-shot is a better practical compromise: it reduces 2026 MAE and improves coverage compared with the target few-shot model, although coverage degradation remains positive.

The language explanation generated from measured drift states that DLRM-2025 differs mainly in active RDMA and topology proxies, while SpotGPU-2026 differs mainly in HP ratio and model-capacity proxies. That text matches the tables and does not introduce independent factual claims. Its purpose is to translate numerical diagnostics into a short explanation for cloud operators. In a production dashboard, the same pattern would allow an LLM interface to say why a transfer forecast should be trusted, rejected, or recalibrated.

Table 5. Top source-target feature drift metrics.

Target	Feature	StandardizedMeanDiff	WassersteinZ	DomainClassifierAUC
DLRM-2025 inference	active_rdma	37.0959	0.3203	1.0000
DLRM-2025 inference	active_mean_unique_dsw	24.9539	0.7634	1.0000
DLRM-2025 inference	active_mean_unique_psw	24.9539	0.7634	1.0000
DLRM-2025 inference	active_jobs	19.8051	0.1678	1.0000
DLRM-2025 inference	active_workers	19.3537	0.1442	1.0000

DLRM-2025 inference	arrival_inference_ratio	16.8760	0.1664	1.0000
SpotGPU-2026	active_hp_ratio	177.61	0.7121	1.0000
SpotGPU-2026	active_mean_unique_psw	79.3861	0.3010	1.0000
SpotGPU-2026	active_mean_unique_asw	30.0375	0.1680	1.0000
SpotGPU-2026	active_mean_unique_dsw	22.5291	0.3549	1.0000
SpotGPU-2026	active_jobs	18.2947	0.1334	1.0000
SpotGPU-2026	active_workers	16.8636	0.2550	1.0000

Figure 2. Normalized active GPU capacity over hourly windows.

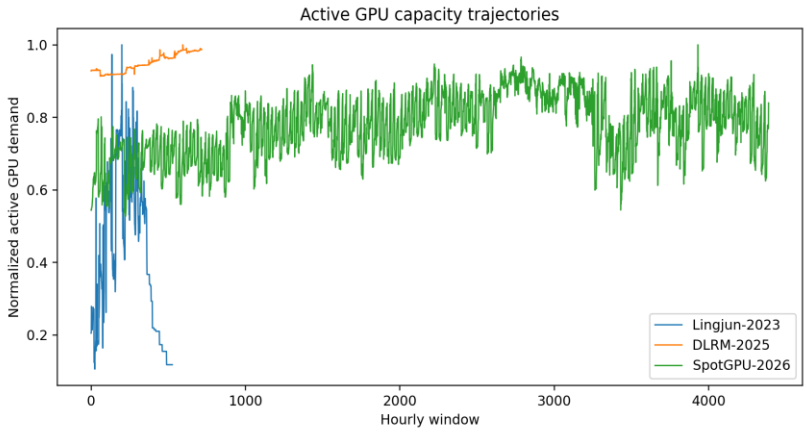


Figure 3. Largest measured source-target feature drifts.

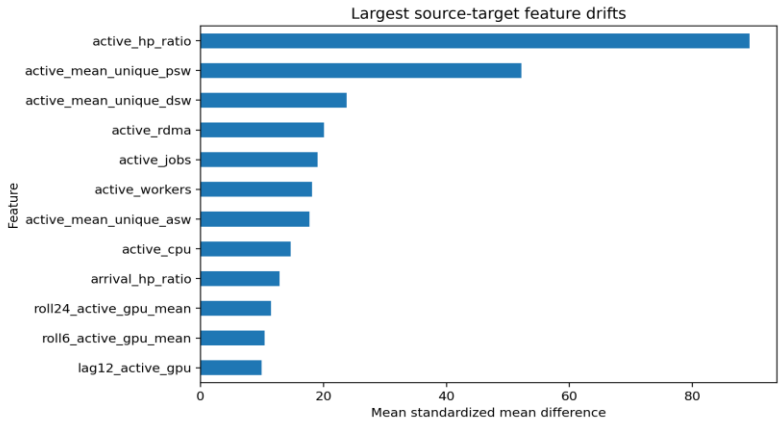


Figure 4. PCA embedding shift across source and target domains.

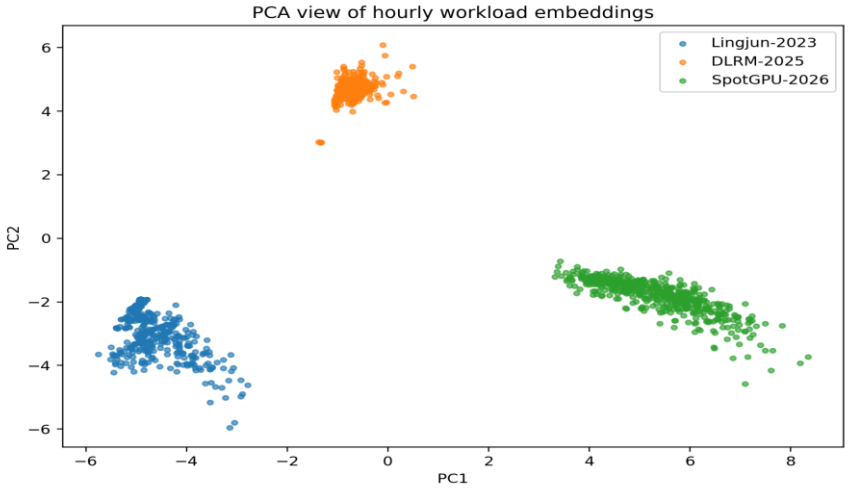


Table 6. Six-hour forecasting performance on the 2025 DLRM inference target.

Target	Method	Support Windows	MAE	RMSE	SMAPE	R2	MeanPrediction	MeanActual
DLRM-2025 inference	Persistence	107	22.8525	233.45	1.1849	0.0418	3212.52	3198.46
DLRM-2025 inference	ExtraTrees source-only	107	816.80	833.93	29.7517	-11.2262	2405.19	3198.46

DLRM-2025 inference	Ridge source-only	107	2768.97	2774.63	152.67	-134.35	433.90	3198.46
DLRM-2025 inference	ExtraTrees target few-shot	107	38.0507	234.14	1.6603	0.0362	3194.02	3198.46
DLRM-2025 inference	ExtraTrees pooled few-shot	107	40.1236	235.76	1.7213	0.0228	3213.01	3198.46
DLRM-2025 inference	Residual few-shot adapter	107	262.64	401.66	8.0666	-1.8362	3461.08	3198.46
DLRM-2025 inference	Importance-weighted Ridge	107	2768.97	2774.63	152.67	-134.35	433.90	3198.46
DLRM-2025 inference	CORAL feature alignment	107	844.33	963.18	31.9565	-15.3098	2370.73	3198.46
DLRM-2025 inference	DANN few-shot	107	194.55	338.05	6.8335	-1.0091	3051.20	3198.46

Table 7. Six-hour forecasting performance on the 2026 spot-GPU target.

Target	Method	Support Windows	MAE	RMSE	SMAPE	R2	MeanPrediction	MeanActual
SpotGPU-2026	Persistence	658	647.79	916.53	7.3439	0.1186	9150.70	9139.67
SpotGPU-2026	ExtraTrees source-only	658	811.36	1020.70	9.1012	-0.0931	9111.35	9139.67
SpotGPU-2026	Ridge source-only	658	58481.73	58822.17	152.23	-3629.49	67621.41	9139.67

SpotGPU-2026	ExtraTrees target few-shot	658	616.94	811.20	7.0335	0.3095	8643.90	9139.67
SpotGPU-2026	ExtraTrees pooled few-shot	658	491.44	700.81	5.5935	0.4847	8854.81	9139.67
SpotGPU-2026	Residual few-shot adapter	658	3170.02	4823.81	28.1055	-23.4153	11545.63	9139.67
SpotGPU-2026	Importance-weighted Ridge	658	58481.73	58822.17	152.23	-3629.49	67621.41	9139.67
SpotGPU-2026	CORAL feature alignment	658	20372.89	28797.53	83.1516	-869.15	29493.79	9139.67
SpotGPU-2026	DANN few-shot	658	43892.91	49018.54	130.00	-2520.18	52799.71	9139.67

Figure 5. Transfer performance comparison; MAE axis is logarithmic.

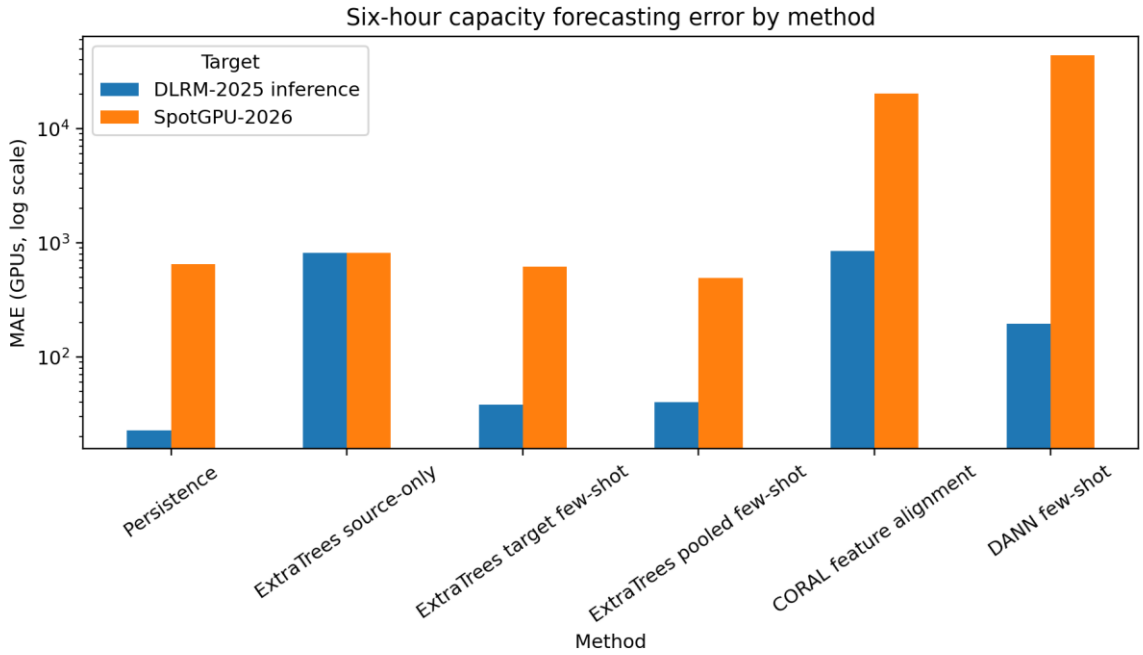


Table 8. Few-shot adaptation curve by target-support fraction.

Target	SupportFraction	SupportWindows	Method	MAE	RMSE
DLRM-2025 inference	0.0100	24	Persistence	20.8889	219.09
DLRM-2025 inference	0.0100	24	ExtraTrees source-only	807.95	824.56
DLRM-2025 inference	0.0100	24	ExtraTrees target few-shot	21.3045	219.01
DLRM-2025 inference	0.0100	24	ExtraTrees pooled few-shot	130.41	261.52
DLRM-2025 inference	0.0500	35	Persistence	21.1584	220.84
DLRM-2025 inference	0.0500	35	ExtraTrees source-only	810.70	827.13
DLRM-2025 inference	0.0500	35	ExtraTrees target few-shot	21.8463	220.77
DLRM-2025 inference	0.0500	35	ExtraTrees pooled few-shot	128.52	257.43
DLRM-2025 inference	0.1000	71	Persistence	21.7554	226.86
DLRM-2025 inference	0.1000	71	ExtraTrees source-only	812.24	829.14
DLRM-2025 inference	0.1000	71	ExtraTrees target few-shot	37.8833	227.94
DLRM-2025 inference	0.1000	71	ExtraTrees pooled few-shot	60.0382	234.09
DLRM-2025 inference	0.1500	107	Persistence	22.8525	233.45
DLRM-2025 inference	0.1500	107	ExtraTrees source-only	816.80	833.93
DLRM-2025 inference	0.1500	107	ExtraTrees target few-shot	37.1921	234.13

DLRM-2025 inference	0.1500	107	ExtraTrees pooled few-shot	40.1236	235.76
DLRM-2025 inference	0.2500	179	Persistence	25.7342	248.58
DLRM-2025 inference	0.2500	179	ExtraTrees source-only	820.90	839.22
DLRM-2025 inference	0.2500	179	ExtraTrees target few-shot	43.4815	249.44
DLRM-2025 inference	0.2500	179	ExtraTrees pooled few-shot	34.8520	249.21
SpotGPU-2026	0.0100	43	Persistence	644.61	905.01
SpotGPU-2026	0.0100	43	ExtraTrees source-only	810.90	1012.42
SpotGPU-2026	0.0100	43	ExtraTrees target few-shot	1226.45	1425.69
SpotGPU-2026	0.0100	43	ExtraTrees pooled few-shot	552.45	771.57
SpotGPU-2026	0.0500	219	Persistence	647.26	906.24
SpotGPU-2026	0.0500	219	ExtraTrees source-only	812.49	1014.98
SpotGPU-2026	0.0500	219	ExtraTrees target few-shot	1079.97	1275.33
SpotGPU-2026	0.0500	219	ExtraTrees pooled few-shot	658.11	886.61
SpotGPU-2026	0.1000	439	Persistence	650.40	913.91
SpotGPU-2026	0.1000	439	ExtraTrees source-only	813.63	1019.54
SpotGPU-2026	0.1000	439	ExtraTrees target few-shot	843.13	1018.02
SpotGPU-2026	0.1000	439	ExtraTrees pooled few-shot	546.22	761.59
SpotGPU-2026	0.1500	658	Persistence	647.79	916.53

SpotGPU-2026	0.1500	658	ExtraTrees source-only	811.36	1020.70
SpotGPU-2026	0.1500	658	ExtraTrees target few-shot	622.58	816.72
SpotGPU-2026	0.1500	658	ExtraTrees pooled few-shot	491.44	700.81
SpotGPU-2026	0.2500	1097	Persistence	652.44	929.32
SpotGPU-2026	0.2500	1097	ExtraTrees source-only	822.83	1039.19
SpotGPU-2026	0.2500	1097	ExtraTrees target few-shot	439.75	652.29
SpotGPU-2026	0.2500	1097	ExtraTrees pooled few-shot	434.40	654.39

Figure 6. Few-shot adaptation sensitivity.

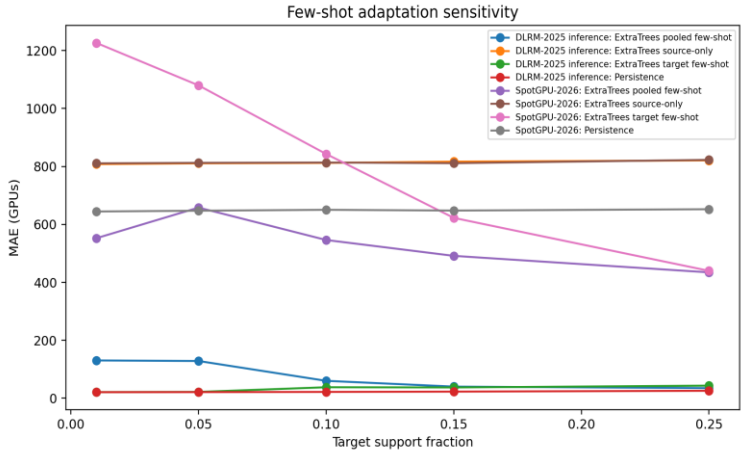


Table 9. Negative transfer rate and conformal coverage degradation.

Target	Method	NegativeTransferRate_vs_TargetFewShot	IntervalHalfWidth90	Coverage90	CoverageDegradation
DLRM-2025 inference	Persistence	0.0885	8.4000	0.7787	0.1213

DLRM-2025 inference	ExtraTrees source-only	0.9951	908.12	0.7672	0.1328
DLRM-2025 inference	Ridge source-only	0.9951	2708.27	0.1934	0.7066
DLRM-2025 inference	ExtraTrees target few-shot	0.0000	2.4250	0.0459	0.8541
DLRM-2025 inference	ExtraTrees pooled few-shot	0.4033	17.2074	0.5393	0.3607
DLRM-2025 inference	Residual few-shot adapter	0.9738	14.0066	0.0230	0.8770
DLRM-2025 inference	Importance-weighted Ridge	0.9951	2708.27	0.1934	0.7066
DLRM-2025 inference	CORAL feature alignment	0.9934	412.76	0.2148	0.6852
DLRM-2025 inference	DANN few-shot	0.9508	213.16	0.7000	0.2000
SpotGPU-2026	Persistence	0.4456	1373.67	0.8818	0.0182
SpotGPU-2026	ExtraTrees source-only	0.6318	1516.25	0.8976	0.0024
SpotGPU-2026	Ridge source-only	1.0000	56382.16	0.3583	0.5417
SpotGPU-2026	ExtraTrees target few-shot	0.0000	376.67	0.3663	0.5337
SpotGPU-2026	ExtraTrees pooled few-shot	0.2789	531.47	0.6442	0.2558
SpotGPU-2026	Residual few-shot adapter	0.8494	733.53	0.1787	0.7213
SpotGPU-2026	Importance-weighted Ridge	1.0000	56382.16	0.3583	0.5417
SpotGPU-2026	CORAL feature alignment	0.9882	960.97	0.0190	0.8810
SpotGPU-2026	DANN few-shot	0.9971	982.14	0.0096	0.8904

Figure 7. Coverage degradation of 90% conformal intervals.

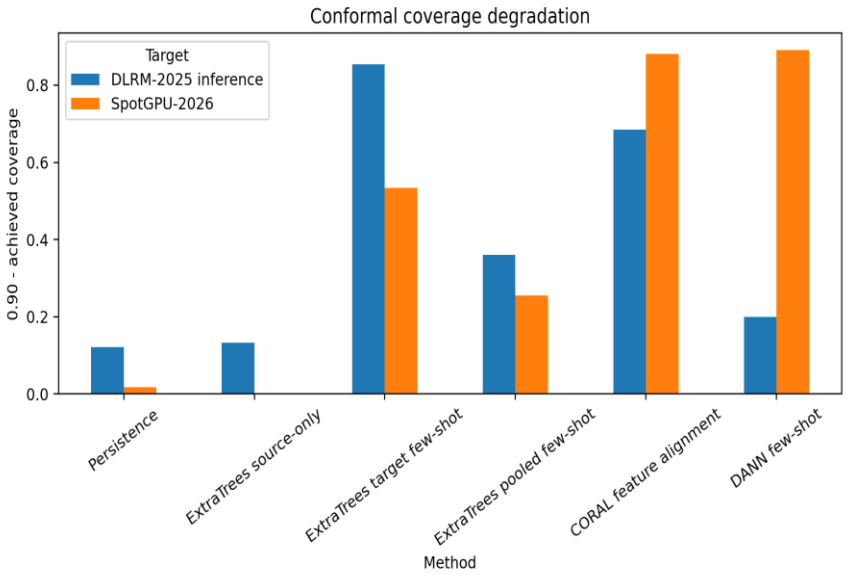


Figure 8. Standardized workload and topology summary heatmap.

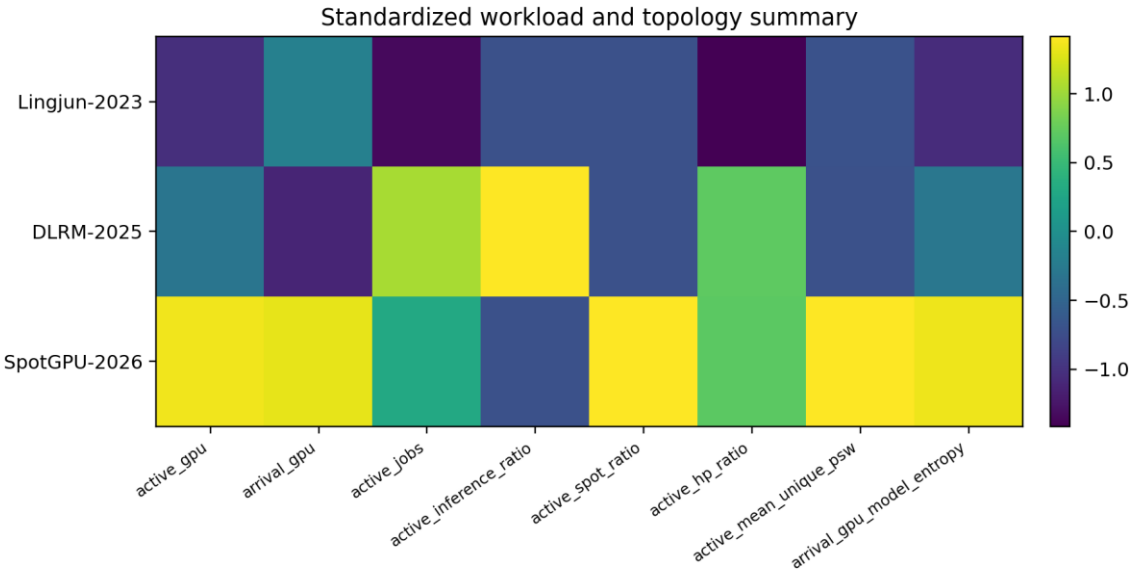
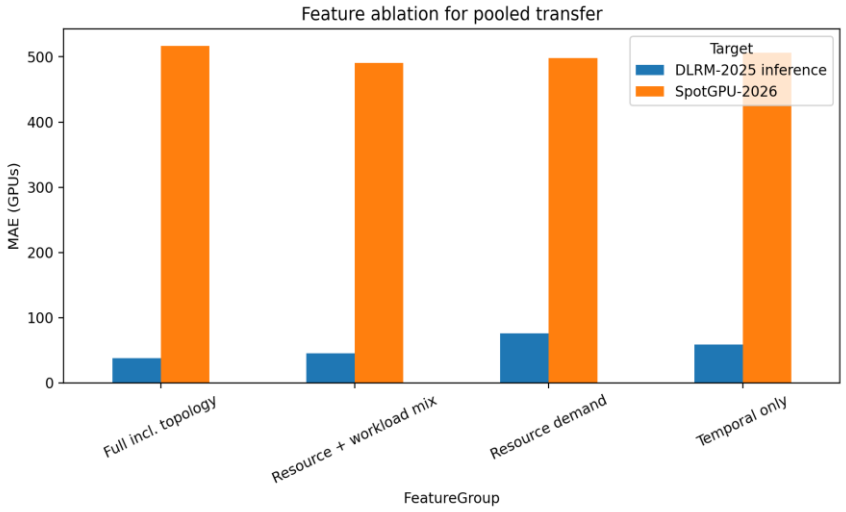


Table 10. Feature ablation for pooled few-shot transfer.

Target	FeatureGroup	NumFeatures	MAE	RMSE	SMAPE	R2

DLRM-2025 inference	Temporal only	13	58.3059	241.36	2.3035	-0.0241
DLRM-2025 inference	Resource demand	24	76.2043	250.46	2.8774	-0.1028
DLRM-2025 inference	Resource + workload mix	36	45.4265	236.89	1.8929	0.0134
DLRM-2025 inference	Full incl. topology	45	37.6261	234.83	1.6447	0.0305
SpotGPU-2026	Temporal only	13	506.16	719.05	5.7728	0.4575
SpotGPU-2026	Resource demand	24	498.01	717.62	5.6814	0.4597
SpotGPU-2026	Resource + workload mix	36	490.91	702.59	5.5946	0.4820
SpotGPU-2026	Full incl. topology	45	516.95	724.11	5.8801	0.4498

Figure 9. Feature ablation for pooled transfer.



Limitations

The experiment uses three Alibaba trace releases and does not claim provider-independent generality. The phrase cross-cloud in this paper denotes cross-release and cross-cluster cloud domains with different workload classes inside a hyperscale environment. A multi-provider replication would require comparable public GPU traces from other

operators with compatible timestamps, resource requests, and topology fields.

The 2025 inference target is a DLRM serving trace rather than an LLM text-generation trace. It is still a production GPU inference workload, but its long-running behavior makes persistence unusually strong. The conclusion for LLM inference bursts should be tested on a trace with token-level or request-level GPU occupancy. The 2026 spot-GPU target lacks actual placement paths, so topology is represented by GPU-model capacity rather than observed host-switch placement. That proxy is useful for capacity mix but weaker than Lingjun's host/DSW/PSW/ASW topology.

The adaptation methods are intentionally compact and reproducible. DANN uses a small full-batch neural model, and CORAL uses covariance alignment followed by Ridge. Larger sequence models, online learners, or probabilistic state-space models could improve accuracy, but they would also add tuning complexity. The present study fixes simple hyperparameters to make negative transfer visible and repeatable.

The evaluation uses request and allocation traces, not hardware performance counters. It forecasts requested or active GPU capacity rather than achieved FLOPS, token throughput, or training progress. This is the correct layer for capacity reservation, but it does not measure model-quality outcomes or job-level slowdown. The paper also uses a six-hour horizon. Shorter horizons favor persistence, and longer horizons increase the value of arrival and duration features. The code can be rerun with a different horizon, but the reported claims are tied to the six-hour setting.

The target support split is chronological and fixed at 15% for the main tables. This represents a warm-up period, not a randomized sample. The few-shot curve demonstrates that different support fractions change results. The fixed split is reproducible and operationally realistic, but it is not a guarantee that 15% support is optimal for every cloud domain.

Conclusion

This paper conducted a full reproducible experimental evaluation of cross-domain GPU capacity forecasting from Lingjun 2023 AI training jobs to 2025 DLRM inference and 2026 spot-GPU target traces. The study normalized all datasets into hourly capacity windows, forecast active GPU demand six hours ahead, and compared source-only, few-shot, pooled, residual, reweighted, feature-aligned, and adversarial transfer methods. The measured distribution shift is large enough that source-only transfer is unreliable. For the 2025 inference target, persistence is best because the service pool is long running and stable; for the 2026 spot-GPU target, pooled few-shot transfer is best and reduces MAE by 39.4% compared with source-only ExtraTrees. The practical conclusion is that cloud operators should not reuse an older training trace without target support. A small chronological target sample, combined with resource, workload, and topology-aware features, converts transfer learning from a negative-transfer risk into a useful capacity-planning tool.

The final artifact package supports review and reuse: raw CSVs, preprocessing, experiments, figures, tables, processed windows, and run summaries are included. The manuscript's numerical claims are tied to these files, and all statements about the experimental setup are definite and reproducible.

References

[1] Q. Weng, W. Xiao, Y. Yu, W. Wang, C. Wang, J. He, Y. Li, L. Zhang, W. Lin, and Y. Ding, "MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters," in Proc. 19th USENIX Symp. Networked Systems Design and Implementation (NSDI), 2022, pp. 945–960.

[2] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Heterogeneity and Dynamicity of Clouds at Scale: Google Trace Analysis," in Proc. ACM Symp. Cloud Computing (SoCC), 2012, pp. 1–13.

[3] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale Cluster Management at Google with Borg," in Proc. European Conf. Computer Systems (EuroSys), 2015, pp. 1–17.

[4] Siming Zhao, Haozhe Wang, and Neil Davison, “Profit-Maximizing Cost-Sensitive Credit Scoring with LLM-Extracted Policy Constraints”, JACS, vol. 4, no. 3, pp. 91–108, Mar. 2024, doi: 10.69987/JACS.2024.40307.

[5] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy: Fair Scheduling for Distributed Computing Clusters," in Proc. ACM Symp. Operating Systems Principles (SOSP), 2009, pp. 261–276.

[6] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling," in Proc. EuroSys, 2010, pp. 265–278.

[7] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant Resource Fairness: Fair Allocation of Multiple Resource Types," in Proc. USENIX NSDI, 2011, pp. 323–336.

[8] J. Guo, Z. Chang, S. Wang, H. Ding, Y. Feng, L. Mao, and Y. Bao, "Who Limits the Resource Efficiency of My Datacenter: An Analysis of Alibaba Datacenter Traces," in Proc. IEEE/ACM Int. Symp. Quality of Service (IWQoS), 2019, pp. 1–10.

[9] Yunhe Li, “Risk-Sensitive Offline Reinforcement Learning for Stable ABR QoE Improvements on Real HSDPA and LTE Traces”, JACS, vol. 3, no. 4, pp. 1–11, Apr. 2023, doi: 10.69987/JACS.2023.30401.

[10] J. Quiñonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, Eds., Dataset Shift in Machine Learning. Cambridge, MA, USA: MIT Press, 2009.

[11] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," IEEE Trans. Knowl. Data Eng., vol. 22, no. 10, pp. 1345–1359, 2010.

[12] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A Theory of Learning from Different Domains," Mach. Learn., vol. 79, no. 1–2, pp. 151–175, 2010.

[13] M. Sugiyama, M. Krauledat, and K.-R. Müller, "Covariate Shift Adaptation by Importance Weighted Cross Validation," J. Mach. Learn. Res., vol. 8, pp. 985–1005, 2007.

[14] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A Kernel Two-Sample Test," J. Mach. Learn. Res., vol. 13, pp. 723–773, 2012.

[15] B. Sun, J. Feng, and K. Saenko, "Return of Frustratingly Easy Domain Adaptation," in Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops, 2016, pp. 2058–2065.

[16] Y. Ganin et al., "Domain-Adversarial Training of Neural Networks," J. Mach. Learn. Res., vol. 17, no. 59, pp. 1–35, 2016.

[17] Yuanzheng Chen, Yitian Zhang, and Matt Sherman, “Going Concern and Bankruptcy Prediction under Extreme Class Imbalance: Cost-Sensitive Learning, Resampling, and Focal Loss with Explainable Financial-Ratio Portraits”, JACS, vol. 4, no. 4, pp. 80–96, Apr. 2024, doi: 10.69987/JACS.2024.40407.

[18] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," Ann. Statist., vol. 29, no. 5, pp. 1189–1232, 2001.

[19] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, 2016, pp. 785–794.

[20] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," Technometrics, vol. 12, no. 1, pp. 55–67, 1970.

[21] V. Vovk, A. Gammerman, and G. Shafer, Algorithmic Learning in a Random World. New York, NY, USA: Springer, 2005.

[22] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial Discriminative Domain Adaptation," in Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2017, pp. 7167–7176.

[23] T. B. Brown et al., "Language Models are Few-Shot Learners," in Proc. Advances in Neural

Information Processing Systems, vol. 33, 2020, pp. 1877–1901.

[24] J. Wei et al., "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," in Proc. Advances in Neural Information Processing Systems, vol. 35, 2022, pp. 24824–24837.

[25] Daren Zheng, Boning Zhang, and Julie Geibel, “VerifySafe: Toxicity-Safe Agent Responses under Adversarial Prompts with Evidence-Based Self-Verification”, JACS, vol. 4, no. 1, pp. 67–82, Jan. 2024, doi: 10.69987/JACS.2024.40106.

[26] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in Proc. Int. Conf. Learning Representations, 2015.