Decomposing Temporal and Job-Induced Dynamics for Probabilistic Computing Workload Forecasting via Graph-Conditioned Dual-Branch Diffusion

Baozhen Luo[1*] **Minbo Ma**[1*†] **Honglin Zhang**[1] **Yuan Yuan**[1]
Quanlu Zhang[2] **Baodong Wu**[2] **Yong Li**[1†]
[1]Tsinghua University [2]Infinigence AI
luobz21@mails.tsinghua.edu.cn minboma@mail.tsinghua.edu.cn

Abstract

Workload forecasting is a fundamental task for realizing electricity-computing synergy in modern data centers. Yet machine-level workloads in production clusters usually exhibit substantial volatility, which is driven by asynchronous job arrivals, heterogeneous resource demands, and dynamic scheduling decisions. Such volatility makes probabilistic forecasting essential for uncertainty-aware operational decision-making. However, most existing workload forecasting methods remain deterministic, and both deterministic and probabilistic approaches overlook the job-induced cross-machine dependencies shaped by the scheduling layer. We empirically find that workload volatility exhibits structured and time-varying cross-machine patterns whose strength aligns with job co-occurrence. This finding motivates GD-Diff, a job-aware graph-conditioned diffusion framework for probabilistic workload forecasting. It constructs a job-induced weighted dynamic graph from scheduling logs, transforming job co-occurrence into time-varying inter-machine dependencies. Conditioned on this graph, our method uses a dual-branch noise prediction network within the diffusion process, where the temporal branch captures regular temporal evolution and the graph branch models job-induced dynamics. Experiments on real-world datasets show that the proposed method outperforms other baselines in both point and probabilistic forecasting The code is available at `https://github.com/linb21/GD-Diff`.

1 Introduction

Workload forecasting is central to electricity-computing synergy in modern data centers [6]. By anticipating future computing demand, it enables proactive resource scheduling [22, 30] and demand-aware coordination with renewable generation and storage [7]. This coordination is critical for aligning AI-driven electricity demand with the fluctuating supply of wind and solar power [10]. This alignment offers a practical pathway toward lower operating costs, improved grid reliability, and reduced carbon emissions.

However, forecasting production workloads remains challenging due to their substantial volatility. As shown in Figure 1(a), our analysis of real-world Alibaba cluster traces [11] illustrates the wide quantile ranges at the same time of day. Such volatility arises from the superposition of heterogeneous operational activities, including asynchronous job arrivals, diverse resource demands, and dynamic

[*]Equal contribution.
[†]Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

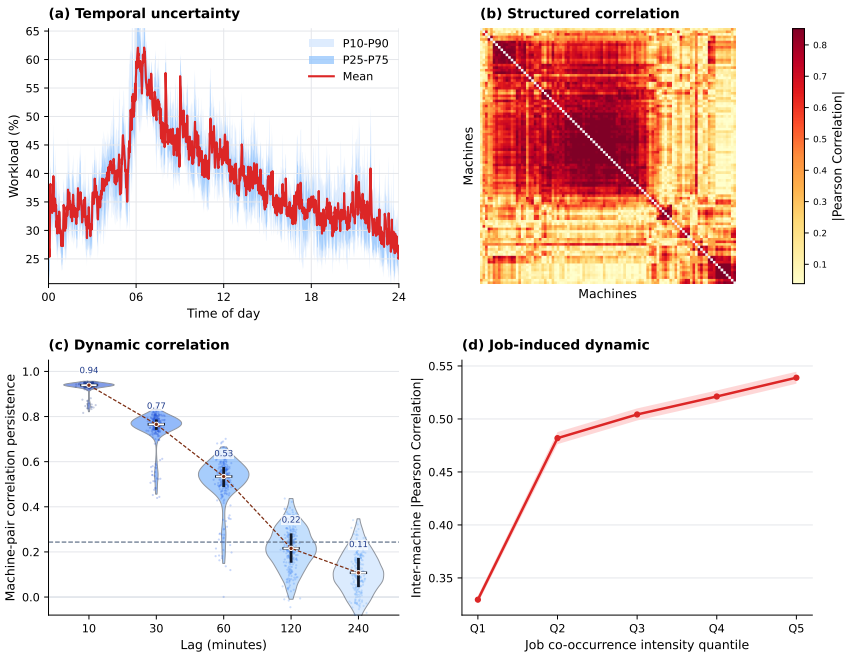

Figure 1: Workload volatility is structured, dynamic, and associated with job scheduling. The analysis uses 100 machines from Alibaba trace. (a) Cross-day variation in machine-averaged CPU utilization, aligned by time of day. The curve shows the cross-day mean, and the shaded bands indicate the P10–P90 and P25–P75 ranges. (b) The residual correlation matrix exhibits block structure across machines. (c) Machine-pair correlation persistence decreases with temporal lag, indicating time-varying dependencies. (d) As historical job co-occurrence intensity increases from Q1 (lowest) to Q5 (highest), the mean future absolute residual correlation also increases.

scheduling decisions [33, 39]. These factors introduce irregular variations that cannot be adequately captured by temporal correlations alone.

Existing methods typically treat workload forecasting as the canonical time series forecasting problem, extrapolating future resource utilization from historical observations with temporal modeling by statistical methods [3], recurrent neural networks [8, 40, 38], and Transformer-based architectures [23]. Most of these methods follow a deterministic point-estimation paradigm, but such a paradigm is insufficient for electricity-computing synergy, where downstream operational planning needs to account for the risk caused by fluctuating computing demand. This requirement motivates a probabilistic formulation of workload forecasting, in which future workloads are modeled as predictive distributions rather than single point estimates, drawing on the broader studies on probabilistic time-series forecasting [27, 12].

Despite this progress, existing approaches primarily infer uncertainty from temporal correlations, while largely overlooking the *job-induced cross-machine dependencies behind workload volatility.* Since distributed jobs often place related tasks on multiple machines, their execution dynamics can induce correlated workload changes across machines. To further probe this gap, we jointly analyze workload traces and job scheduling logs, and find that workload volatility exhibits structured, time-varying cross-machine correlations whose strength is closely associated with historical job co-occurrence, as shown in Figure 1(b-d). Specifically, the residual workload sequences show clear correlation blocks across machines, and such correlations are strongest at short temporal lags. More importantly, machine pairs with stronger historical job co-occurrence tend to have higher future residual correlations (see detailed evidence in Section 2.2). These observations indicate that workload volatility is partly induced by the scheduling layer, rather than being purely random temporal noise. We refer to this structured source of variation as *job-induced dynamics*, which indicates scheduling-induced dependencies beyond regular temporal evolution.

Motivated by these empirical findings, we propose GD-Diff, a job-aware graph-conditioned diffusion framework for probabilistic workload forecasting. It first constructs a job-induced weighted dynamic graph from scheduling logs as an empirically grounded structural prior, where edges encode time-varying inter-machine dependencies driven by job co-occurrence. Conditioned on this graph, our method introduces a dual-branch denoising network: a Transformer branch captures temporal dependencies in workload traces, while a Graph Convolution branch models job-induced cross-machine dependencies over the dynamic graph. Furthermore, an adaptive gated mechanism fuses their representations to guide the reverse diffusion process, enabling GD-Diff to generate diverse future workload trajectories. Overall, our main contributions are summarized as follows:

- We empirically identify that workload volatility exhibits job-induced dynamics, and formulate this finding as a job co-occurrence weighted graph, enabling time-varying inter-machine dependencies to be captured beyond what can be inferred from workload traces alone.
- We propose GD-Diff, a job-aware graph-conditioned diffusion framework that decomposes job-induced dynamics from temporal evolution through a dual-branch noise prediction network.
- We conduct extensive experiments on real-world cluster trajectory. Results show that the proposed method consistently outperforms deterministic and probabilistic forecasting baselines, while ablation studies validate the effectiveness of the job-induced dynamic graph and the dual-branch architecture.

2 Preliminary

2.1 Problem Formulation.

We address probabilistic multivariate time-series forecasting of machine-level workloads in production clusters. The data is denoted as $X = [X^{co}, X^{ta}] \in \mathbb{R}^{N\times(L+K)}$, where N is the number of machines, $X^{co} \in \mathbb{R}^{N\times L}$ the observed history over the past L steps, and $X^{ta} \in \mathbb{R}^{N\times K}$ the prediction target over the next K steps. The job scheduling records available at forecasting time t are denoted by $\mathcal{J}_{\le t}$. They contain only the observed history of job executions up to t, including job start records and the machines on which their tasks have been observed. The scheduling history $\mathcal{J}_{\le t}$ can be further encoded as a job-induced dynamic graph $\mathcal{G}^{co}$, whose nodes are machines and whose edges summarize historical job co-occurrence between machine pairs. Our goal is to learn a probabilistic forecaster $\mathcal{F}_\theta$ by solving

$$\mathcal{F}_\theta^* \leftarrow \arg\min_{\mathcal{F}_\theta} \mathbb{E}_{(X^{co},\, \mathcal{J}_{\le t},\, X^{ta})\sim\mathcal{D}} \left[\mathcal{L}\big(\mathcal{F}_\theta(X^{co}, \mathcal{G}^{co}), X^{ta}\big)\right], \quad \mathcal{G}^{co} = \Phi(\mathcal{J}_{\le t}), \tag{1}$$

where $\mathcal{D}$ denotes the training distribution, $\Phi(\cdot)$ denotes the graph construction procedure, and $\mathcal{L}(\cdot)$ denotes the probabilistic forecasting objective.

2.2 Job–Workload Mechanism and Empirical Evidence

In production clusters, machine-level workload is not an isolated time-series signal, but the continuous resource-usage outcome of a hierarchical job execution process. Users submit jobs to the cluster, and each job is decomposed into tasks, which are further instantiated as runnable instances by the cluster management system. The scheduler then places these instances onto physical machines according to resource requirements, machine availability, and placement constraints. During execution, each instance consumes resources over time, and the observed workload of a machine is the aggregate resource usage of all running instances assigned to that machine. This mechanism establishes a natural connection between job scheduling and workload variation: when instances from the same distributed job are placed on multiple machines, their execution can introduces temporally aligned workload variations across these machines, which we refer to as *job-induced dynamics*.

We next examine whether workload volatility exhibits cross-machine structures that are associated with job scheduling history. Figure 1(a) summarizes cross-day variation in the mean CPU utilization of 100 machines over eight days. At each minute of the day, the P10–P90 band captures the spread across days. Its width averages 11.94 percentage points, equivalent to 31.04% of the overall mean utilization, and reaches 29.81 percentage points at its widest. These variations indicate substantial workload uncertainty beyond the mean daily profile.

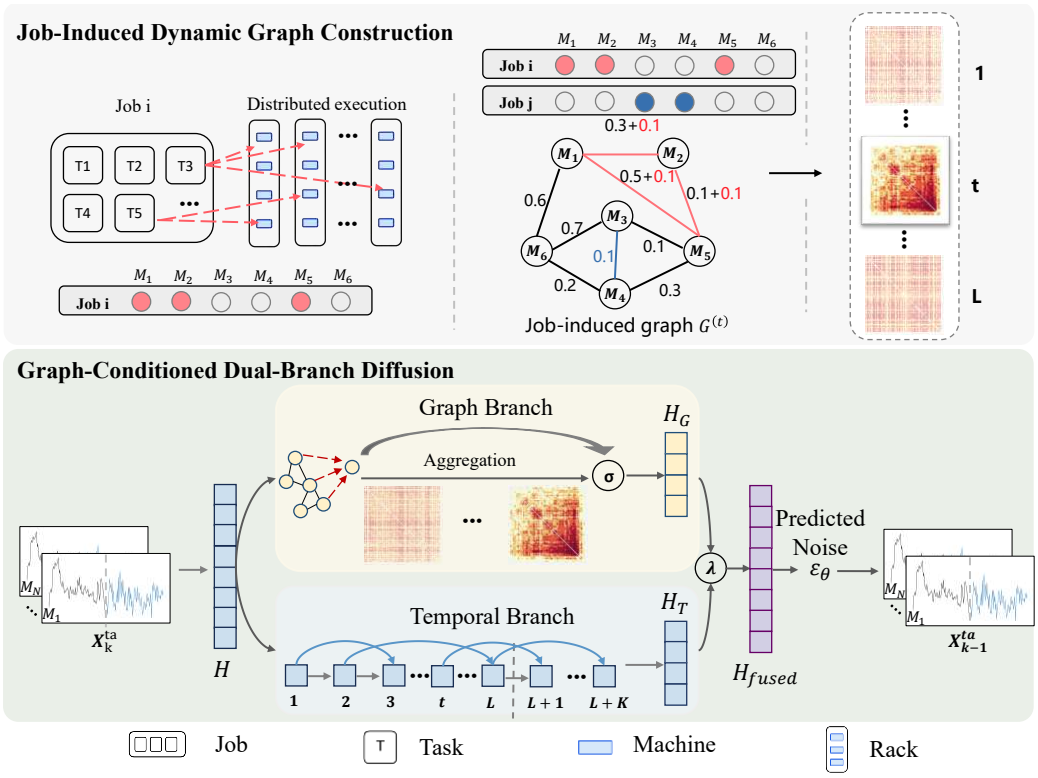

Figure 2: Overall architecture of the proposed Dual-Branch Diffusion Model. The framework decouples workload dynamics into temporal and job-induced components through a parallel noise prediction network.

For each machine pair (i, j), the residual correlation at window start t is defined as

$$R_{ij}(t) = \left|\mathrm{corr}\big(r_i(t:t+L),\, r_j(t:t+L)\big)\right|. \tag{2}$$

As shown in Figure 1(b), the residual correlation matrix $R(t)$ exhibits pronounced block structure, indicating that workload volatility carries structured inter-machine dependencies rather than independent machine-level noise. To examine whether these dependencies are static or dynamic, we compare $R_{ij}(t)$ with its lagged counterpart $R_{ij}(t+\tau)$ over time for each machine pair (i, j). The random baseline is obtained by randomizing machine-pair identities across time while preserving the correlation distribution at each time. Figure 1(c) shows that the correlation between $R_{ij}(t)$ and $R_{ij}(t+\tau)$ decays as the lag τ increases and eventually falls below the random baseline. This suggests that inter-machine dependencies are temporally dynamic rather than persistent static relations.

Finally, we relate these dynamic inter-machine dependencies to the scheduling history $\mathcal{J}_{\le t}$. For each machine pair, we use its historical job co-occurrence in $\mathcal{G}^{co}$ as a job-induced structural signal. Machine pairs are then divided into five quantile groups according to this signal, and the average future residual correlation is computed within each group. Figure 1(d) shows that the mean future absolute residual Pearson correlation increases monotonically from 0.329 in Q1 to 0.539 in Q5, corresponding to a relative increase of 63.6%. Thus, machine pairs with stronger historical job co-occurrence exhibit stronger future workload synchronization, supporting the use of scheduling-derived relations as a structural conditioning signal for forecasting.

3 Methodology

Overview. GD-Diff models this distribution as a conditional denoising diffusion probabilistic model, in which a noise predictor ϵ_θ iteratively denoises Gaussian samples into target workloads while conditioning jointly on past observations X^{co} and a job-induced dynamic graph $\mathcal{G}^{co}$. As illustrated in

Figure 2, GD-Diff comprises three components: (1) a job-induced dynamic graph $\mathcal{G}^{co}$ constructed from $\mathcal{J}_{\leq t}$ that exposes inter-machine dependencies driven by distributed jobs (Section 3.1); (2) a dual-branch design of ϵ_θ that decouples regular temporal evolution from graph-driven structural fluctuations (Section 3.2); and (3) the conditional diffusion framework that turns ϵ_θ into a probabilistic forecaster (Section 3.3).

3.1 Job-Induced Dynamic Graph Construction

To explicitly encode job-induced inter-machine dependencies, we construct a job-induced weighted dynamic graph from scheduling logs based on machine-level job co-occurrence.

Edges from distributed jobs. At each historical time step $\tau \leq t$, let $\mathcal{V}_m^{(\tau)} \subseteq \{1, \ldots, N\}$ denote the set of machines on which job J_m has tasks observed running at τ, as recorded in $\mathcal{J}_{\leq t}$. An active job spanning multiple machines induces co-occurrence edges among those machines. For any pair (i, j) with $i, j \in \mathcal{V}_m^{(\tau)}$, machines i and j are considered to jointly execute job J_m at time τ. Thus, a job observed on q machines contributes to the $\binom{q}{2}$ pairwise co-occurrence edges among them.

Job co-occurrence intensity. Aggregating the edge contributions of all jobs at time τ, we define the *job co-occurrence intensity* between machines i and j as

$$C_{i,j}^{(\tau)} = \sum_m \mathbb{I}\Big(i \in \mathcal{V}_m^{(\tau)} \wedge j \in \mathcal{V}_m^{(\tau)}\Big), \quad i, j \in \{1, \ldots, N\}, \tag{3}$$

Therefore, $C_{i,j}^{(\tau)}$ counts how many jobs are jointly observed on machines i and j at time τ. A larger $C_{i,j}^{(\tau)}$ indicates stronger job-induced inter-machine dependency, and is empirically associated with stronger future residual correlations, as shown in Figure 1(d). The matrix $C^{(\tau)} \in \mathbb{R}^{N \times N}$ therefore forms a weighted job co-occurrence graph at time τ.

Dynamic graph sequence. Repeating the above construction at each historical time step yields a sequence of raw job co-occurrence graphs, $\mathcal{C}_{t-L+1:t}^{co} = \{C^{(\tau)}\}_{\tau=t-L+1}^{t}$, which captures how job-induced inter-machine dependencies evolve with scheduling over time. Only job-machine assignments observed no later than the forecasting time t are used; information after t, including future job arrivals, future task placements, and future completion events, is excluded.

Graph normalization and sparsification. To stabilize the scale of edge weights, we normalize the co-occurrence intensities using min–max statistics: $\widetilde{C}_{i,j}^{(\tau)} = (C_{i,j}^{(\tau)} - C_{\min}^{\text{train}})/(C_{\max}^{\text{train}} - C_{\min}^{\text{train}})$. We then maintain, for each machine, only the k strongest normalized co-occurrence dependencies:

$$G_{i,j}^{(\tau)} = \widetilde{C}_{i,j}^{(\tau)} \cdot \mathbb{I}\Big(j \in \mathrm{TopK}\Big(\widetilde{C}_{i,:}^{(\tau)}, k\Big)\Big). \tag{4}$$

The resulting dynamic graph sequence $\mathcal{G}^{co} = \{G^{(\tau)}\}_{\tau=t-L+1}^{t}$ encodes time-varying job-induced dependencies and is fed into the graph branch of the dual-branch noise predictor.

3.2 Dual-Branch Noise Prediction Network

As discussed in Section 2, workload volatility stems from two heterogeneous sources: regular temporal evolution driven by periodic user behavior, and job-induced dynamics driven by distributed scheduling. To disentangle them in the noise prediction process, we design ϵ_θ as a dual-branch network: a *temporal branch* captures regular temporal evolution along the time axis, and a *graph branch* captures job-induced dynamics along the machine axis through the dynamic graph $\mathcal{G}^{co}$.

Decomposed dual dynamics. Given a hidden representation H derived from X^{co} and X_k^{ta}, the two branches extract complementary representations in parallel with the diffusion-step embedding $e_k = \mathrm{Emb}(k)$ injected into both:

$$H_{\mathcal{T}} = \mathrm{Transformer}\big(H + e_k\big), \tag{5}$$

$$H_{\mathcal{G}} = \mathrm{GatedGCN}\big(H + e_k,\ \mathcal{G}^{co}\big). \tag{6}$$

The temporal branch operates along the time axis to capture regular temporal evolution. The graph branch propagates messages along the machine axis under the job co-occurrence edges in $\mathcal{G}^{co}$, with a learned gate that modulates how much neighbor information is injected.

Adaptive gated fusion. The relative dominance of regular temporal evolution versus job-induced dynamics shifts over time. We fuse the two representations with a time-variant gate

$$\lambda = \sigma\big(\mathcal{W}_{gate}(H_{\mathcal{T}} + H_{\mathcal{G}})\big) \in (0, 1), \tag{7}$$

producing the fused representation

$$H_{fused} = \lambda \odot H_{\mathcal{T}} + (1 - \lambda) \odot H_{\mathcal{G}}. \tag{8}$$

from which the predicted noise $\hat{\epsilon}$ is obtained. A value of λ close to 1 indicates a state dominated by regular temporal evolution, while a value close to 0 signals strong job-induced dynamics.

3.3 Graph-Conditioned Diffusion Framework

With the noise predictor ϵ_θ defined above and the dynamic graph $\mathcal{G}^{co}$ defined in Section 3.1, we now situate them within the conditional DDPM framework [13, 29] that completes GD-Diff. The forward process gradually corrupts the clean target X_0^{ta} into Gaussian noise over T steps under a fixed variance schedule $\{\beta_k\}_{k=1}^{T}$. The reverse process learns to invert this corruption by repeatedly applying ϵ_θ, jointly conditioning on the observed history X^{co} and the dynamic graph $\mathcal{G}^{co}$. The model is trained with the standard noise-matching objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{X_0^{ta},\, X^{co},\, \mathcal{G}^{co},\, \epsilon,\, k}\Big[\, \big\|\epsilon - \epsilon_\theta\big(X_k^{ta},\, k \,\big|\, X^{co},\, \mathcal{G}^{co}\big)\big\|_2^2 \,\Big], \tag{9}$$

where $k \sim \mathrm{Uniform}\{1, \ldots, T\}$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. At inference, the trained ϵ_θ is applied iteratively to denoise Gaussian samples into plausible workload trajectories, yielding both point estimates and prediction intervals for risk-aware downstream decision-making.

4 Experiments

4.1 Experimental Setup

Experimental Setup. We evaluate GD-Diff on two public production cluster traces: Alibaba Cluster Trace 2018 [11] and Google Cluster Trace 2019 [36]. For both datasets, we construct a 100-machine subset, with 1-minute resolution for Alibaba and 5-minute resolution for Google according to their resolution. The look-back window is fixed to $L = 60$, and we evaluate two forecasting horizons, $K \in \{6, 30\}$. We compare GD-Diff with deterministic forecasting models, including GRU [5], AGCRN [2], TimeMixer [31], iTransformer [20], PatchTST [24], and TimeFilter [14], as well as generative forecasting models, including CSDI [29], DiffSTG [34], TMDM [19], and D^{3}U [16]. All models are trained with three random seeds $\{2024, 2025, 2026\}$ on NVIDIA RTX 4090 GPUs, and we report the mean and standard deviation across runs. At inference, GD-Diff draws 100 samples from the diffusion process. We use MAE and RMSE to evaluate point forecasting accuracy and CRPS to evaluate probabilistic forecasting quality, where lower values indicate better performance for all metrics. Full hyperparameter settings are provided in Appendix A.

4.2 Main Results

Table 1 reports deterministic forecasting results on Google and Alibaba datasets, averaged over three random seeds. GD-Diff achieves the best MAE in all four (dataset, horizon) settings and the best RMSE in three of four, demonstrating that job-induced structural conditioning consistently improves the central forecast while remaining competitive under squared-error evaluation. Among the baselines, recent Transformer-based methods (TimeFilter, PatchTST, iTransformer) form the strongest pool. AGCRN, equipped with a self-adaptive AVWGCN graph, reaches the level of the strongest temporal baselines, indicating that data-driven adaptive graphs recover part of the inter-machine dependency, whereas DiffSTG, built on a predefined adjacency, cannot adapt to the dynamic dependencies driven by job scheduling. Table 2 reports probabilistic metric CRPS. GD-Diff attains the lowest CRPS across all settings, outperforming all probabilistic diffusion baselines including CSDI, TMDM, and

Table 1: Deterministic forecasting results on Google Cluster Trace and Alibaba Cluster Trace. All results are reported as mean ± standard deviation over three random seeds. Lower values indicate better performance. The best results are highlighted in bold and the second-best results are underlined.

Dataset	Model	60→6		60→30	
		MAE	RMSE	MAE	RMSE
Google	GRU	0.0794 ± 0.0005	0.1082 ± 0.0007	0.0848 ± 0.0014	0.1158 ± 0.0018
	TimeMixer	0.0543 ± 0.0003	0.0815 ± 0.0003	0.0643 ± 0.0004	0.0942 ± 0.0005
	iTransformer	0.0544 ± 0.0001	0.0818 ± 0.0000	0.0642 ± 0.0000	0.0949 ± 0.0000
	AGCRN	0.0535 ± 0.0000	$\underline{0.0806 \pm 0.0000}$	$\underline{0.0627 \pm 0.0001}$	$\underline{0.0922 \pm 0.0001}$
	PatchTST	0.0541 ± 0.0000	0.0815 ± 0.0000	0.0637 ± 0.0000	0.0945 ± 0.0000
	TimeFilter	0.0539 ± 0.0001	0.0813 ± 0.0002	0.0643 ± 0.0001	0.0949 ± 0.0002
	TMDM	0.0664 ± 0.0008	0.0952 ± 0.0009	0.0740 ± 0.0020	0.1050 ± 0.0029
	DiffSTG	0.0701 ± 0.0059	0.0980 ± 0.0077	0.0855 ± 0.0134	0.1176 ± 0.0173
	D3U	0.0585 ± 0.0039	0.0851 ± 0.0025	0.0672 ± 0.0028	0.0973 ± 0.0022
	CSDI	$\underline{0.0533 \pm 0.0005}$	0.0807 ± 0.0006	0.0634 ± 0.0010	0.0940 ± 0.0015
	GD-Diff	$\mathbf{0.0524 \pm 0.0001}$	$\mathbf{0.0795 \pm 0.0001}$	$\mathbf{0.0615 \pm 0.0007}$	$\mathbf{0.0905 \pm 0.0009}$
Alibaba	GRU	7.7882 ± 0.3104	12.0414 ± 0.3677	8.5387 ± 0.1820	12.8191 ± 0.4931
	TimeMixer	4.5947 ± 0.0349	6.4605 ± 0.0464	5.7344 ± 0.0792	8.2124 ± 0.0939
	iTransformer	4.5817 ± 0.0048	6.4548 ± 0.0080	5.7588 ± 0.0137	8.1880 ± 0.0143
	AGCRN	4.8083 ± 0.0217	6.6749 ± 0.0343	6.0902 ± 0.0325	8.7310 ± 0.0532
	PatchTST	4.5402 ± 0.0045	$\underline{6.3860 \pm 0.0050}$	$\underline{5.6522 \pm 0.0197}$	$\underline{8.0808 \pm 0.0109}$
	TimeFilter	$\underline{4.5156 \pm 0.0053}$	$\mathbf{6.3829 \pm 0.0110}$	5.7179 ± 0.0382	8.2162 ± 0.0964
	TMDM	5.3834 ± 0.1039	7.4365 ± 0.1260	6.6318 ± 0.4736	9.2438 ± 0.5939
	DiffSTG	8.0316 ± 0.6057	11.0979 ± 1.5308	7.3189 ± 0.4783	10.1301 ± 0.6582
	D^3U	4.9502 ± 0.2565	6.6798 ± 0.1718	5.9452 ± 0.1019	8.3266 ± 0.0356
	CSDI	4.7331 ± 0.0604	6.6396 ± 0.0859	6.1252 ± 0.2708	9.2129 ± 0.7587
	GD-Diff	$\mathbf{4.4354 \pm 0.0286}$	6.4351 ± 0.0577	$\mathbf{5.4736 \pm 0.0265}$	$\mathbf{8.0390 \pm 0.0685}$

Table 2: Probabilistic forecasting results on Google Cluster Trace and Alibaba Cluster Trace. All results are reported as mean ± standard deviation over three random seeds. Lower CRPS indicates better probabilistic forecasting quality. The best results are highlighted in bold and the second-best results are underlined.

Model	Google		Alibaba	
	60→6	60→30	60→6	60→30
TMDM	0.0513 ± 0.0005	0.0551 ± 0.0013	4.2221 ± 0.1048	4.8792 ± 0.3192
DiffSTG	0.05698 ± 0.0040	0.06874 ± 0.0089	6.37434 ± 0.5247	6.03319 ± 0.3533
D^3U	0.0431 ± 0.0024	0.0495 ± 0.0010	3.5133 ± 0.1503	$\underline{4.2715 \pm 0.0180}$
CSDI	$\underline{0.0393 \pm 0.0003}$	$\underline{0.0468 \pm 0.0008}$	$\underline{3.4727 \pm 0.0775}$	4.4099 ± 0.2149
GD-Diff	$\mathbf{0.0381 \pm 0.0000}$	$\mathbf{0.0449 \pm 0.0006}$	$\mathbf{3.2374 \pm 0.0012}$	$\mathbf{4.0320 \pm 0.0303}$

the recent D^3U. This indicates that architectural refinements to the diffusion framework alone do not translate into gains on cluster workload forecasting; calibrated uncertainty additionally requires modeling the structural source of volatility, which GD-Diff captures through its job-induced graph and preserves through the dual-branch noise predictor, driving consistent state-of-the-art performance.

4.3 Ablation Study

To examine the contribution of the two core designs in GD-Diff, we compare the full model against three controlled variants:

- **w/ Attn**: replaces the graph branch with a self-attention module across machines, modeling inter-machine interactions without an explicit graph.
- **w/ SG**: replaces $\mathcal{G}^{co}$ with a learnable latent graph inferred from workload data, removing the explicit scheduling-derived structure.
- **w/o Dual**: replaces the parallel dual-branch design with a serial T→G structure (the temporal module followed by the graph module), keeping $\mathcal{G}^{co}$ unchanged.

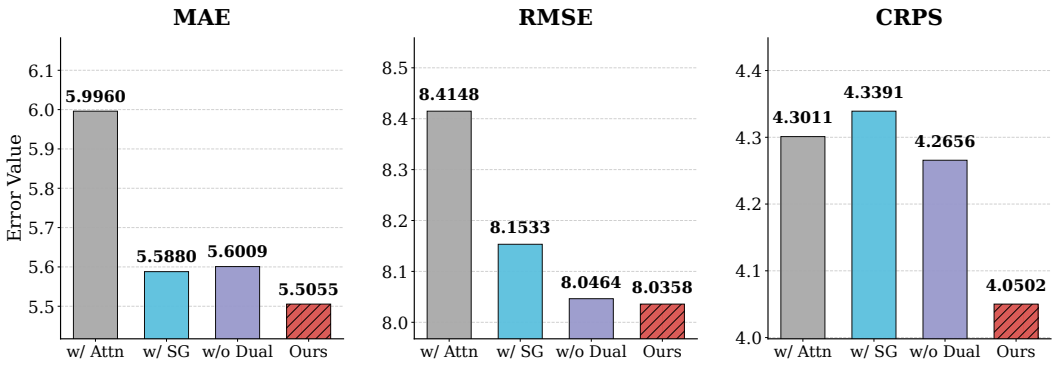

Figure 3: Ablation study results comparing GD-Diff against three controlled variants. Reported values represent the average performance across 30-step forecasting horizons in Alibaba dataset.

Effect of the job-induced graph. The w/ SG and w/ Attn variants both trail GD-Diff. Whether the model learns a latent graph from workload data or learns attention across machines without any graph, neither approach recovers the complex, dynamic inter-machine dependencies. This confirms that GD-Diff's gains come from the inter-machine structure induced by job co-occurrence.

Effect of the parallel dual-branch design. The w/o Dual variant, which feeds the temporal output into the graph module instead of running the two in parallel, also underperforms GD-Diff: once placed downstream of temporal smoothing, the graph branch receives representations from which the sharp, scheduling-driven signals it is meant to capture have already been partially absorbed. The parallel dual-branch design avoids this by feeding both branches the raw hidden representation directly, preserving the decomposition between regular temporal evolution and job-induced dynamics until fusion.

4.4 Adaptive Fusion Analysis

To further inspect how GD-Diff combines temporal and graph-based representations, we analyze the learned fusion gate in Eq. 8 on the Alibaba dataset. Figure 4 reports the average graph-branch mixing weight, $1-\lambda$, across residual layers and sequence positions.

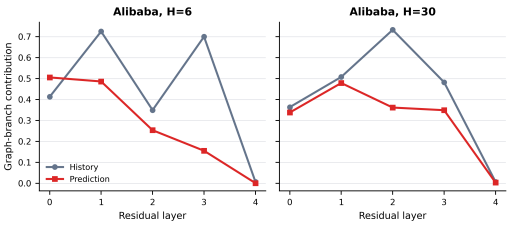

Figure 4: Adaptive branch fusion behavior on Alibaba. The plotted value is the graph-branch mixing weight, $1-\lambda$, where λ denotes the temporal-branch weight in the fusion module. Curves are averaged over test windows and nodes at the final denoising step.

The results show that the graph branch is not used uniformly across layers. Some intermediate layers assign non-negligible weights to graph-based representations, while the final layer almost fully relies on the temporal branch. This suggests that GD-Diff uses graph information selectively within the denoising network, rather than injecting it with a fixed strength at every layer. We also observe different fusion patterns between short-horizon and long-horizon forecasting, indicating that the learned gate adapts to the forecasting setting.

4.5 Performance under high workload variability.

As shown in Figure 5, GD-Diff's relative improvements increase with workload variability on Google under the $60 \rightarrow 30$ setting. On the top 5% of windows with the largest workload changes, GD-Diff reduces MAE by 12.94% compared with TimeFilter and CRPS by 11.88% compared with CSDI, demonstrating a more pronounced advantage under substantial workload fluctuations.

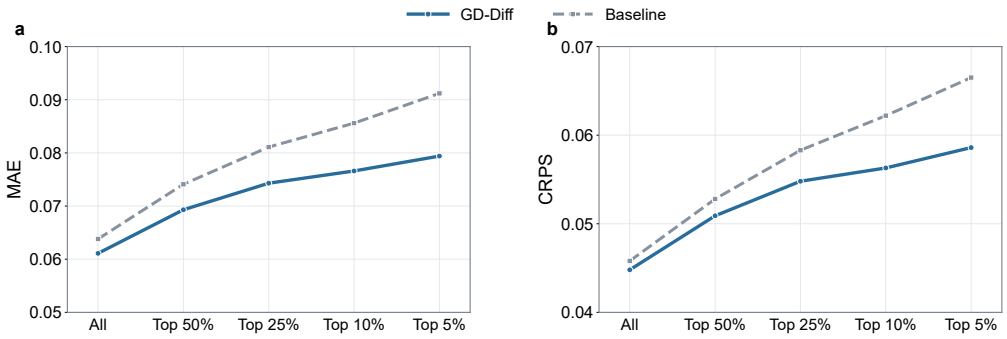

Figure 5: Forecasting performance on subsets with increasing workload variability on Google $60 \rightarrow 30$. Top $p\%$ denotes the windows with the largest history-to-future changes in mean machine workloads. The baseline is TimeFilter for MAE (a) and CSDI for CRPS (b). Lower values indicate better performance.

5 Related Work

5.1 Workload Forecasting in Cloud Data Centers

Workload forecasting has been widely studied as a foundation for resource provisioning, auto-scaling, and energy-aware data center management. Early approaches typically used statistical models to extrapolate workload traces, but they were limited in capturing nonlinear patterns and sensitive to volatility [3, 1, 17]. Deep learning methods further improved forecasting accuracy by modeling nonlinear temporal dependencies with recurrent networks [8, 40], hybrid CNN/RNN architectures [4, 41], multivariate predictors [38], and recent convolution-augmented transformer models [23].

These methods establish strong trace-based forecasting baselines, but most of them output deterministic point estimates and therefore cannot quantify the uncertainty needed by risk-aware decisions. Beyond uncertainty, a more fundamental limitation is that existing workload forecasters primarily learn from historical utilization traces. Even when multivariate correlations are considered, the dependencies are usually inferred from workload values alone, without using the scheduling-layer information that directly shapes machine-level resource usage. As a result, these methods have limited ability to explain or exploit the structured workload volatility induced by distributed jobs.

5.2 Probabilistic Forecasting

Probabilistic forecasting models the conditional distribution of future values and provides calibrated uncertainty estimates for downstream decisions [9]. Representative methods include quantile-based sequence models such as MQ-CNN [35] and autoregressive distributional models such as DeepAR [28]. In cloud computing, recent work has begun to emphasize uncertainty-aware workload prediction for more reliable resource management [27, 12]. However, parametric methods may be constrained by distributional assumptions, while quantile-based methods only approximate selected parts of the predictive distribution. Denoising diffusion probabilistic models provide a flexible generative framework for modeling complex data distributions [13]. For time-series forecasting, TimeGrad [26] combines diffusion models with autoregressive recurrent networks, CSDI [29] formulates forecasting as conditional score-based imputation, and recent diffusion forecasters such as TMDM [19] and D^3U [16] further improve temporal representation and uncertainty decomposition. These methods learn expressive conditional distributions over time series, but their conditioning signals are mainly historical observations and temporal embeddings.

5.3 Spatio-temporal Graph Forecasting

Spatio-temporal graph forecasting models cross-series dependencies by combining temporal modeling with graph-based message passing. It has been successful in domains with explicit topologies, such

as traffic networks, where road connectivity or learned spatial relations can guide graph neural networks [18, 32]. When the underlying topology is unavailable or incomplete, adaptive graph learning has been introduced to infer latent spatial relations directly from data. Representative deterministic models include MTGNN [37] and AGCRN [2], which learn hidden dependencies through adaptive graph learning mechanisms such as node embeddings or graph learning modules. Recent probabilistic variants further integrate graph neural networks with diffusion models. For example, DiffSTG [34] combines denoising diffusion with spatio-temporal graph modeling, while DiffGSL [15] learns adaptive graph structures when the topology is unknown.

Directly applying these methods to workloads forecasting is non-trivial because machine dependencies in production clusters are not fixed physical links. They are implicit, time-varying, and shaped by distributed job execution. Predefined static graphs may therefore fail to match scheduling-induced dependencies, while purely self-adaptive graphs must infer non-stationary structure from workload traces alone and may miss the operational mechanism behind correlated fluctuations.

6 Conclusion

We presented GD-Diff, a graph-conditioned dual-branch diffusion model for probabilistic workload forecasting in production clusters. By constructing a job-induced dynamic graph from scheduling logs and decomposing the noise predictor into a temporal branch and a graph branch, GD-Diff explicitly disentangles regular temporal evolution from job-induced dynamics. Experiments on real-world datasets show that GD-Diff consistently improves both point accuracy and probabilistic calibration over state-of-the-art deterministic and generative baselines.

GD-Diff incurs additional inference cost through iterative denoising and repeated sampling, which may limit its use in latency-sensitive settings. Our evaluation trains separate models on Alibaba and Google, while zero-shot transfer to unseen clusters or machines remains unexplored. Improving sampling efficiency and cross-cluster transferability are directions for future work. Nevertheless, this cost remains practical for the minute-to-hour operational decisions that motivate electricity-computing synergy [21, 25]. GD-Diff thus provides a reliable forecasting foundation for risk-aware resource management in this setting.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Grant No. 2026ZD1212207 and No. U23B2030). We also gratefully thank Infinigence AI for providing valuable domain expertise and computational resources in support of this research.

References

[1] Danilo Ardagna, Sara Casolari, Michele Colajanni, and Barbara Panicucci. Dual time-scale distributed capacity allocation and load redirect algorithms for cloud systems. *Journal of Parallel and Distributed Computing*, 72(6):796–808, 2012.

[2] LEI BAI, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17804–17815. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/ce1aad92b939420fc17005e5461e6f48-Paper.pdf`.

[3] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.

[4] Lei Chen, Weiwen Zhang, and Haiming Ye. Accurate workload prediction for edge data centers: Savitzky-golay filter, cnn and bilstm with attention mechanism. *Applied Intelligence*, 52(11): 13027–13042, 2022.

[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical*

Methods in Natural Language Processing (EMNLP), pages 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179.

[6] Philip Colangelo, Ayse K Coskun, Jack Megrue, Ciaran Roberts, Shayan Sengupta, Varun Sivaram, Ethan Tiao, Aroon Vijaykar, Chris Williams, Daniel C Wilson, et al. Ai data centres as grid-interactive assets. *Nature Energy*, 11(2):254–261, 2026.

[7] Mehiar Dabbagh, Bechir Hamdaoui, Ammar Rayes, and Mohsen Guizani. Shaving data center power demand peaks through energy storage and workload shifting control. *IEEE Transactions on Cloud Computing*, 7(4):1095–1108, 2017.

[8] Martin Duggan, Karl Mason, Jim Duggan, Enda Howley, and Enda Barrett. Predicting host cpu utilization in cloud computing using recurrent neural networks. In *2017 12th international conference for internet technology and secured transactions (ICITST)*, pages 67–72. IEEE, 2017.

[9] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[10] Jiaju Guo, Wenjuan Hou, Xiaoyue Wang, Xueliang Zhang, Shaohong Wu, Linsheng Yang, Wenhui Jiang, and Lunwei Zhang. Toward a new spatial paradigm of electricity–computing synergy under renewable energy endowments and geographical environmental constraints. *Applied Energy*, 409:127495, 2026.

[11] Jing Guo, Zihao Chang, Sa Wang, Haiyang Ding, Yihui Feng, Liang Mao, and Yungang Bao. Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. In *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2019. doi: 10.1145/3326285.3329074.

[12] Haitian Hang, Xiu Tang, Jianling Sun, Lingfeng Bao, David Lo, and Haoye Wang. Robust auto-scaling with probabilistic workload forecasting for cloud databases. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 4016–4029. IEEE, 2024.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[14] Yifan Hu, Guibin Zhang, Peiyuan Liu, Disen Lan, Naiqi Li, Dawei Cheng, Tao Dai, Shu-Tao Xia, and Shirui Pan. Timefilter: Patch-specific spatial-temporal graph filtration for time series forecasting. *arXiv preprint arXiv:2501.13041*, 2025.

[15] Chanyoung Jung and Yun Jang. Diffgsl: A graph structure learning diffusion model for dynamic spatio-temporal forecasting. In *2024 IEEE International Conference on Big Data (BigData)*, page 5785–5793. IEEE, 2024.

[16] Qi Li, Zhenyu Zhang, Lei Yao, Zhaoxia Li, Tianyi Zhong, and Yong Zhang. Diffusion-based decoupled deterministic and uncertain framework for probabilistic multivariate time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025.

[17] Ta-Hsin Li. A hierarchical framework for modeling and forecasting web server workload. *Journal of the American Statistical Association*, 100(471):748–763, 2005.

[18] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.

[19] Yuxin Li, Wenchao Chen, Xinyue Hu, Bo Chen, Baolin Sun, and Mingyuan Zhou. Transformer-modulated diffusion models for probabilistic multivariate time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[21] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew. Greening geographical load balancing. *ACM SIGMETRICS Performance Evaluation Review*, 39(1): 193–204, 2011.

[22] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A Lozano. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of grid computing*, 12(4): 559–592, 2014.

[23] Fan'an Meng, Hongjun Dai, Guoqing Cong, Bo Zhu, and Hailiang Zhao. Catscaler: A convolution-augmented transformer scaling framework for cloud-native applications. *IEEE Transactions on Services Computing*, 2025.

[24] Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. *arXiv preprint arXiv:2211.14730*, 2022.

[25] Ana Radovanović, Ross Koningstein, Ian Schneider, Bokan Chen, Alexandre Duarte, Binz Roy, Diyue Xiao, Maya Haridasan, Patrick Hung, Nick Care, et al. Carbon-aware computing for datacenters. *IEEE Transactions on Power Systems*, 38(2):1270–1280, 2022.

[26] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8857–8868. PMLR, 2021.

[27] Andrea Rossi, Andrea Visentin, Diego Carraro, Steven Prestwich, and Kenneth N Brown. Forecasting workload in cloud computing: towards uncertainty-aware predictions and transfer learning. *Cluster Computing*, 28(4):258, 2025.

[28] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

[29] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34:24804–24816, 2021.

[30] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the tenth european conference on computer systems*, pages 1–17, 2015.

[31] Shiyu Wang, Haixu Wu, Xiaoming Shi, Tengge Hu, Huakun Luo, Lintao Ma, James Y Zhang, and Jun Zhou. Timemixer: Decomposable multiscale mixing for time series forecasting. *arXiv preprint arXiv:2405.14616*, 2024.

[32] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. Traffic flow prediction via spatial-temporal graph neural network. In *Proceedings of the Web Conference 2020*, pages 1082–1092, Taipei, Taiwan, 2020. ACM.

[33] Yuxin Wang, Yuhan Chen, Zeyu Li, Xueze Kang, Yuchu Fang, Yeju Zhou, Yang Zheng, Zhenheng Tang, Xin He, Rui Guo, et al. Burstgpt: A real-world workload dataset to optimize llm serving systems. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 5831–5841, 2025.

[34] Haomin Wen, Youfang Lin, Yutong Xia, Huaiyu Wan, Qingsong Wen, Roger Zimmermann, and Yuxuan Liang. Diffstg: Probabilistic spatio-temporal graph forecasting with denoising diffusion models. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, pages 1–12, Hamburg, Germany, 2023. ACM.

[35] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[36] John Wilkes. Yet more Google compute cluster trace data. Google research blog, April 2020. Posted at `https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html`.

[37] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 753–763, 2020.

[38] Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. esdnn: deep neural network based multivariate workload prediction in cloud computing environments. *ACM Transactions on Internet Technology (TOIT)*, 22(3):1–24, 2022.

[39] Bin Yang, Wei Xue, Tianyu Zhang, Shichao Liu, Xiaosong Ma, Xiyang Wang, and Weiguo Liu. End-to-end i/o monitoring on leading supercomputers. *ACM Transactions on Storage*, 19(1): 1–35, 2023.

[40] Haitao Yuan, Jing Bi, Shuang Li, Jia Zhang, and MengChu Zhou. An improved lstm-based prediction approach for resources and workload in large-scale data centers. *IEEE Internet of Things Journal*, 11(12):22816–22829, 2024.

[41] Feiyu Zhao, Weiwei Lin, Shengsheng Lin, Haocheng Zhong, and Keqin Li. Tfegru: time-frequency enhanced gated recurrent unit with attention for cloud workload prediction. *IEEE Transactions on Services Computing*, 2024.

A Implementation Details

This appendix lists the hyperparameters of GD-Diff used in our experiments. When the two datasets differ, we report the Alibaba configuration as the canonical reference.

Data and forecasting setting. For both datasets, we use $N{=}100$ machines. Alibaba Cluster Trace 2018 is sampled at 1-minute resolution over 8 days (2018-01-01 to 2018-01-08), while Google Cluster Trace 2019 is sampled at 5-minute resolution over one month (2019-05-01 to 2019-06-01). Each trace is split chronologically into train/validation/test sets with ratio $0.7/0.1/0.2$. The look-back window is fixed to $L{=}60$ steps, and we evaluate two forecasting horizons, $K \in \{6, 30\}$.

Model configuration. The conditional noise predictor ϵ_θ consists of 5 residual blocks with $C{=}64$ hidden channels, 8 attention heads, and a diffusion-step embedding of dimension 128. The DDPM uses $T{=}50$ diffusion steps with a quadratic β-schedule from $\beta_1{=}10^{-4}$ to $\beta_T{=}0.5$. The graph branch is implemented as a GatedGCN with top-k sparsification, with $k = 10$ and five residual blocks for Alibaba, and $k = 15$ and four residual blocks for Google. Edge weights are normalized via min–max statistics.

Optimization and inference. We train GD-Diff with Adam at an initial learning rate of 10^{-3} and batch size 4, for at most 100 epochs with early stopping patience 20. The learning rate is decayed by $\gamma{=}0.1$ at epochs $\{15, 20\}$. Each model is trained with three random seeds $\{2024, 2025, 2026\}$, and we report mean$\pm$std. GD-Diff draws 100 samples from the diffusion process for all experiments.

B Raw Results

For reproducibility, Table 3 reports the raw forecasting results under each random seed used to compute the mean and standard deviation in the main tables.

C Efficiency analysis

Table 4 reports the computational cost of representative forecasting models under the same experimental setting. Compared with deterministic point predictors such as GRU and PatchTST, GD-Diff requires higher GPU memory and longer inference time, reflecting the inherent cost of iterative diffusion sampling. Among diffusion-based forecasters, however, GD-Diff remains at the same order of computational cost as CSDI and D^3U, while providing more reliable probabilistic forecasting. Since the target electricity-computing synergy scenarios involve minute-to-hour operational decisions, the measured inference time remains practical for risk-aware workload forecasting.

Table 3: Per-seed raw forecasting results on Google Cluster Trace and Alibaba Cluster Trace. Unlike the main tables, this table reports the raw values under each random seed. For each dataset, horizon, metric, and seed, the best result is highlighted in bold and the second-best result is underlined.

Dataset	Horizon	Model	MAE by seed ↓			RMSE by seed ↓			CRPS by seed ↓		
			Seed 2024	Seed 2025	Seed 2026	Seed 2024	Seed 2025	Seed 2026	Seed 2024	Seed 2025	Seed 2026
Google	60→6	GRU	0.0790	0.0802	0.0792	0.1073	0.1091	0.1082	–	–	–
	60→6	TimeMixer	0.0540	0.0546	0.0542	0.0812	0.0818	0.0816	–	–	–
	60→6	iTransformer	0.0543	0.0544	0.0544	0.0818	0.0818	0.0819	–	–	–
	60→6	AGCRN	0.0536	0.0535	0.0535	0.0806	0.0805	0.0806	–	–	–
	60→6	PatchTST	0.0541	0.0541	0.0541	0.0815	0.0815	0.0815	–	–	–
	60→6	TimeFilter	0.0541	0.0538	0.0540	0.0815	0.0811	0.0814	–	–	–
	60→6	TMDM	0.0667	0.0654	0.0673	0.0951	0.0942	0.0963	0.0512	0.0506	0.0520
	60→6	DiffSTG	0.0683	0.0639	0.0780	0.0958	0.0898	0.1083	0.0556	0.0529	0.0624
	60→6	D^3U	0.0555	0.0560	0.0640	0.0830	0.0837	0.0887	0.0412	0.0416	0.0465
	60→6	CSDI	0.0534	0.0527	0.0538	0.0804	0.0801	0.0815	0.0394	0.0389	0.0396
	60→6	GD-Diff	**0.0523**	**0.0524**	**0.0524**	**0.0794**	**0.0795**	**0.0796**	**0.0381**	**0.0382**	**0.0381**
	60→30	GRU	0.0829	0.0864	0.0852	0.1133	0.1176	0.1163	–	–	–
	60→30	TimeMixer	0.0648	0.0639	0.0642	0.0948	0.0936	0.0943	–	–	–
	60→30	iTransformer	0.0642	0.0642	0.0642	0.0949	0.0949	0.0950	–	–	–
	60→30	AGCRN	0.0626	0.0628	0.0626	0.0921	0.0923	0.0922	–	–	–
	60→30	PatchTST	0.0637	0.0637	0.0637	0.0945	0.0945	0.0944	–	–	–
	60→30	TimeFilter	0.0644	0.0642	0.0642	0.0952	0.0949	0.0948	–	–	–
	60→30	TMDM	0.0712	0.0759	0.0750	0.1013	0.1082	0.1055	0.0532	0.0563	0.0557
	60→30	DiffSTG	0.0741	0.1043	0.0781	0.1034	0.1419	0.1075	0.0617	0.0813	0.0632
	60→30	D^3U	0.0657	0.0648	0.0711	0.0959	0.0955	0.1004	0.0486	0.0490	0.0508
	60→30	CSDI	0.0624	0.0632	0.0647	0.0923	0.0938	0.0959	0.0458	0.0468	0.0477
	60→30	GD-Diff	**0.0611**	**0.0625**	**0.0608**	**0.0900**	**0.0918**	**0.0898**	**0.0448**	**0.0457**	**0.0444**
Alibaba	60→6	GRU	7.6585	7.4898	8.2162	12.1292	11.5537	12.4415	–	–	–
	60→6	TimeMixer	4.6228	4.5455	4.6158	6.5060	6.3969	6.4785	–	–	–
	60→6	iTransformer	4.5786	4.5781	4.5885	6.4490	6.4493	6.4662	–	–	–
	60→6	AGCRN	4.7915	4.7944	4.8389	6.6267	6.6940	6.7040	–	–	–
	60→6	PatchTST	4.5350	4.5398	4.5459	6.3894	6.3789	**6.3897**	–	–	–
	60→6	TimeFilter	4.5119	4.5118	4.5231	**6.3750**	6.3751	6.3984	–	–	–
	60→6	TMDM	5.2651	5.5180	5.3669	7.2960	7.6017	7.4119	4.0965	4.3531	4.2169
	60→6	DiffSTG	7.2046	8.6383	8.2518	9.5883	13.1964	10.5089	5.6990	6.9784	6.4456
	60→6	D^3U	4.8556	4.6943	5.3008	6.6341	6.4959	6.9093	3.4463	3.3721	3.7216
	60→6	CSDI	4.7828	4.6481	4.7685	6.6452	6.7418	6.5317	3.4946	3.3688	3.5547
	60→6	GD-Diff	**4.4718**	**4.4327**	**4.4018**	6.5085	**6.3675**	6.4293	**3.2357**	**3.2381**	**3.2384**
	60→30	GRU	8.6897	8.2828	8.6437	13.3319	12.1534	12.9718	–	–	–
	60→30	TimeMixer	5.8443	5.6607	5.6981	8.3120	8.0866	8.2386	–	–	–
	60→30	iTransformer	5.7535	5.7452	5.7776	8.1850	8.1722	8.2067	–	–	–
	60→30	AGCRN	6.1303	6.0897	6.0506	8.6666	8.7969	8.7294	–	–	–
	60→30	PatchTST	5.6501	5.6293	5.6773	8.0925	**8.0663**	8.0835	–	–	–
	60→30	TimeFilter	5.6883	5.6934	5.7719	8.1647	8.1326	8.3512	–	–	–
	60→30	TMDM	6.1562	6.4611	7.2780	8.6718	8.9971	10.0623	4.5671	4.7530	5.3177
	60→30	DiffSTG	7.8886	6.7183	7.3497	10.9415	9.3294	10.1193	6.4721	5.6069	6.0206
	60→30	D^3U	5.8444	5.9065	6.0848	8.2782	8.3389	8.3628	4.2474	4.2768	4.2904
	60→30	CSDI	5.8261	6.4819	6.0677	8.3449	10.1932	9.1007	4.2036	4.7063	4.3197
	60→30	GD-Diff	**5.4406**	**5.4747**	**5.5055**	**7.9568**	8.1245	**8.0358**	**3.9892**	**4.0565**	**4.0502**

D Additional Sensitivity and Architecture Analyses

We further examine the effects of graph neighborhood size, the number of generated samples, and the choice of branch architectures. These analyses use the Google dataset with training seed 2024. We report MAE and RMSE for point forecasting, fair CRPS for probabilistic forecasting, and QICE for quantile calibration. QICE is computed using ten equal-probability bins for each machine and then averaged across machines. Lower values indicate better performance for all four metrics. Results are reported for individual runs rather than averages over training seeds.

D.1 Sensitivity to Graph Neighborhood Size

GD-Diff maintains similar forecasting performance across a moderate range of graph neighborhood sizes. Table 5 compares $k \in \{5, 10, 15, 20, 50\}$ under both forecasting horizons. MAE, RMSE, and CRPS vary only modestly for $k = 5$–20, with the lowest values generally obtained at $k = 10$ or 15. Increasing k to 50 provides no consistent benefit and produces a clearer deterioration under the $60 \rightarrow 6$ setting. These observations support using a compact neighborhood to represent job-

Table 4: Efficiency comparison. GPU memory is the peak allocated memory. For diffusion-based models, inference time is measured with 100 generated samples.

Model	**Params**	**Mem. (MB)**	**Train**	**Infer. (ms/sample)**
GRU	47K	494	11.7 s	$\sim$1.5
PatchTST	162K	518	35.2 s	$\sim$45.9
CSDI	415K	2050	6.2 min	$\sim$1180
D3U	1.47M	550	45.7 s	$\sim$680
Ours	628K	2292	6 min	$\sim$1110

Table 5: Sensitivity to graph neighborhood size on Google with seed 2024 and 100 generated samples. The default setting is $k = 15$. Lower values indicate better performance.

	$60 \rightarrow 6$				$60 \rightarrow 30$			
k	MAE	RMSE	CRPS	QICE	MAE	RMSE	CRPS	QICE
5	0.0528	0.0803	0.0386	0.0178	0.0610	0.0901	0.0446	0.0234
10	0.0524	0.0799	0.0382	0.0144	0.0608	0.0898	0.0445	0.0234
15	0.0523	0.0794	0.0381	0.0155	0.0611	0.0900	0.0448	0.0255
20	0.0526	0.0798	0.0384	0.0190	0.0614	0.0903	0.0450	0.0258
50	0.0537	0.0803	0.0393	0.0290	0.0614	0.0902	0.0450	0.0266

induced dependencies, while suggesting that retaining more neighbors does not necessarily improve forecasting.

D.2 Sensitivity to the Number of Generated Samples

Increasing the number of generated samples improves point accuracy and quantile calibration, with diminishing gains beyond approximately 50 samples. Table 6 reports results for sample counts from 3 to 150 under both forecasting horizons. MAE, RMSE, and QICE decrease as more samples are used, whereas fair CRPS remains nearly unchanged. The small differences between 100 and 150 samples support the use of 100 samples in our main evaluation. These results indicate that a moderate sample count provides a stable empirical approximation of the learned predictive distribution.

D.3 Alternative Branch Architectures

The Transformer–GatedGCN combination achieves the best overall results among the evaluated branch architectures. We replace the temporal Transformer with either GRU or TCN, or replace GatedGCN in the graph branch with GAT, while retaining the job-induced graph and parallel dual-branch structure. As shown in Table 7, the GRU variant achieves MAE and RMSE close to those of GD-Diff, but has higher CRPS and QICE. The GAT and TCN variants also produce higher errors. These results support the chosen branch architectures, particularly for probabilistic accuracy and quantile calibration.

Table 6: Sensitivity to the number of generated samples on Google with seed 2024 and $k = 15$. The default sample count is 100. Lower values indicate better performance.

	$60 \rightarrow 6$				$60 \rightarrow 30$			
Samples	MAE	RMSE	CRPS	QICE	MAE	RMSE	CRPS	QICE
3	0.0606	0.0891	0.0381	0.0794	0.0693	0.0998	0.0448	0.0847
10	0.0547	0.0820	0.0381	0.0339	0.0635	0.0927	0.0448	0.0425
20	0.0535	0.0806	0.0381	0.0234	0.0622	0.0912	0.0447	0.0326
30	0.0530	0.0801	0.0381	0.0198	0.0618	0.0907	0.0448	0.0295
50	0.0526	0.0797	0.0381	0.0172	0.0614	0.0903	0.0448	0.0272
80	0.0524	0.0795	0.0381	0.0159	0.0612	0.0901	0.0448	0.0260
100	0.0523	0.0794	0.0381	0.0155	0.0611	0.0900	0.0448	0.0255
150	0.0523	0.0793	0.0381	0.0150	0.0610	0.0899	0.0448	0.0250

Table 7: Alternative temporal and graph branches on Google ($60 \rightarrow 6$, seed 2024). Each variant replaces one branch of GD-Diff. The lowest reported values are shown in bold. Lower values indicate better performance.

Temporal branch	Graph branch	MAE	RMSE	CRPS	QICE
Transformer	GAT	0.0527	0.0797	0.0384	0.0192
TCN	GatedGCN	0.0534	0.0808	0.0391	0.0206
GRU	GatedGCN	**0.0523**	**0.0794**	0.0384	0.0227
Transformer	GatedGCN	**0.0523**	**0.0794**	**0.0381**	**0.0155**

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**

- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: [Yes]

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: [Yes]

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: [Yes]

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: [Yes]

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: [N/A]

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: [N/A]

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.