Transformer-Based Next-Step Prediction for Queue Length Distribution

Jieqi Di*
School of Industrial and Systems Engineering
Georgia Institute of Technology
jdi7@gatech.edu

Jiecheng Lu*
School of Industrial and Systems Engineering
Georgia Institute of Technology
jlu414@gatech.edu

Runhua Wu*
School of Industrial and Systems Engineering
Georgia Institute of Technology
rwu330@gatech.edu

Yuwei Zhou*
Kelley School of Business
Indiana University
yuwzhou@iu.edu

Abstract

Traditional approaches to model queueing dynamics from real-world setting rely on model selection, parameter estimation, and simulation. These methods struggle with complexities like time-varying arrivals, customer abandonment, and routing interactions, while parameter estimation errors and time-consuming simulations further limit their predictive accuracy and increase their costs. We propose a data-driven alternative that learns queueing dynamics directly from historical data using decoder-only Transformers. Our approach uses the autoregressive nature of queueing systems, where future states depend causally on past sequences through recursions. We treat queue evolution as a sequence prediction problem: inputting queue length distributions and predicting next-step distributions. Experiments on synthetic data demonstrate that our Transformer-based model successfully reproduces queueing dynamics across different queueing models and outperforms traditional sequence models like Recurrent Neural Networks (RNNs). This model-free approach eliminates the need for explicit model specification and parameter estimation while maintaining accuracy in capturing complex system behaviors. Future work will validate this methodology on real-world ride-hailing datasets.

1 Introduction

Queueing systems have emerged as a powerful modeling paradigm for solving operational questions in complex modern platforms and marketplaces, such as ride-sharing platforms, energy markets, and large data computing centers. To capture the dynamics of these complex real-world systems, researchers typically follow a pipeline: first collecting data, then hypothesizing queueing models, estimating parameters, and finally simulating to forecast performance or tune policies. However, this pipeline faces multiple challenges in complex environments. First, time-varying arrivals, customer abandonment, and routing interactions across stations make it difficult to build suitable models. Second, even when researchers identify promising models, parameter estimation under partial observability and non-stationarity remains fragile, with errors propagating through simulations. As a result, these simulations often fail to accurately mimic real-world behavior. Beyond accuracy concerns, computational cost poses another practical constraint. Complex real-world models become analytically intractable, forcing researchers to rely on time-consuming simulations. Furthermore, finding the right

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: MLxOR: Mathematical Foundations and Operational Integration of Machine Learning for Uncertainty-Aware Decision-Making.

model requires iterating through different queueing classes, each demanding numerous simulation replications. In a nut shell, this traditional approach demands tremendous effort and expertise to develop, validate, and maintain models that are capable of depicting the true dynamics of real-life stochastic systems.

Rather than wrestling with these modeling challenges, we propose learning system dynamics directly from data through next-step prediction. This approach exploits a fundamental property of queueing systems: their dynamics evolve causally, where each future state depends on the current state plus random events generated from arrival and service processes. For a single-server queue, this causal structure appears clearly in the following recursion:

$$L_{t_{i+1}} = \max\{0,\, L_{t_i} + A_{t_i} - D_{t_i}\}, \tag{1}$$

where L_{t_i} denotes the queue length at time t_i, A_{t_i} (D_{t_i}) denotes the number of arrivals (potential departures) between time t_i and t_{i+1} and t_i is the time immediately before the time of an arrival. In networks, analogous causal updates couple nodes through routing and control. By learning these conditional transitions from historical data, we can make predictions over time to reproduce complex system dynamics. This data-driven approach bypasses the traditional pipeline and eliminates the need for model selection, parameter estimation, and costly simulations.

To implement this data-driven approach, we propose a model-free decoder-only transformer. Causal recursions in queueing systems are inherently autoregressive, where future states depend on past sequences (see Palomo and Pender [2021] for details). This autoregressive structure aligns perfectly with recent autoregressive generative model structure like decoder-only Transformers, which excel at learning sequential dependencies through their causal attention mechanism [Vaswani et al., 2017]. In our setup, we treat queue evolution as a sequence prediction problem: at each time step, we input the queue length distribution (effectively a tokenized summary of the system state) and predict the distribution of the next-step queue length. To train and evaluate this model, we use synthetic data from simulations as both the training set and the performance benchmark. We implement a modern decoder-only Transformer architecture incorporating rotary positional embeddings, RMSNorm, and SwiGLU feed-forward blocks. Our numerical experiments demonstrate that this Transformer-based approach successfully reproduces queueing dynamics across various scenarios (stable, transient, and time-varying systems), showing better performance comparing to other sequence model baselines in deep learning like Recurrent Neural Networks (RNNs). Looking forward, we will test the performance of our learning model on real-world ride-hailing datasets to confirm its practical applicability.

The remainder of this paper is organized as follows. Section 2 reviews related literature. Section 3 presents our methodology and problem setup. Section 4 details our numerical experiments and results. Section 5 concludes with key findings and future directions.

2 Related Work

Data-driven methods for queueing systems have gained significant attention in recent years. Walton and Xu [2021] provides a comprehensive review of learning algorithms in queueing networks. Most existing work focuses on control and optimization problems, employing deep reinforcement learning in Dai and Gluzman [2022] or online learning approaches in Jia et al. [2024]. As transformers and large language models become prevalent, researchers have begun exploring the intersection with queueing theory. Mitzenmacher and Shahout [2025] and Li et al. [2025] examine how to incorporate queue-theoretic principles directly into transformer architecture to enhance its performance.

Despite these advances, few studies tackle the challenge of learning evolving dynamics in complex queueing systems that lack closed-form solutions for waiting times or steady-state distributions. Sherzer et al. [2024] predicts queue length distributions using RNN structures, requiring initial distributions and moments of arrival and service processes as inputs. Ojeda et al. [2021] employs recurrent point processes to predict waiting times and next event times autoregressively. Mittal et al. [2025] propose a transformer-based framework that learns queueing network dynamics from event data. In contrast, our work directly learn from observed queue length distributions at each time step by employing a state-of-the-art transformer architecture.

3 Problem Setup and Methodology

In the following sections, we focus specifically on $G/G/1$ and $G_t/G/1$ queues as a starting point. In this section, we introduce our problem setup, which formally defines our next-step queue length

prediction problem. Moreover, we state the transformer architecture used to accomplish this forecasting task. Let the time horizon be a discrete set $\{1, 2, \ldots, T\}$ and define the stochastic processes L_t as the queue length at time t, where $t \in \{1, 2, \ldots, T\}$. We seek to infer the law of the queue length distribution $\mathbf{X}_{T+1}$ in the system, given the inputs are the law of the queue length distribution $\mathbf{X}_t$ at each time point $t \in \{1, 2, \ldots, T\}$. To provide the queue length distribution as valid inputs as a finite and fixed-dimensional vector, we truncate the dimension of $\mathbf{X}_t$ at d such that the probability of having more than d customers is smaller than $\delta = 10^{-6}$. Now, we formally define $\mathbf{X}_t$ as the stochastic process with values in the probability simplex $\Delta^{d-1} = \{\mathbf{x} \in \mathbb{R}^d_{\geq 0} : \|p\|_1 = 1\}$, denoting the queue length distribution at time t. Then, we have the i^{th} entry of the vector $\mathbf{X}_t$ represents the probability of the queue length equal to i at time t, namely $\mathbf{X}_t^i = \mathbb{P}(L_t = i)$. Write the *ground-truth one-step conditional for the queue length distribution* as

$$\mathbf{X}^\star_{T+1} = \mathbb{E}[L_{T+1} \mid \mathbf{X}_{1:T}] \in \Delta^{d-1} \text{ as the law of } \big(L_{T+1} \mid \mathbf{X}_{1:T}\big). \tag{2}$$

Our objective is to train a decoder-only Transformer $\texttt{TF}(\cdot|\mathbf{X}_{1:T})$ that predicts the logits of the next step, using a softmax function to transform it in to a predicted distribution vector $\widehat{\mathbf{X}}^\star_{T+1}$, approximating the ground truth $\mathbf{X}^\star_{T+1}$ with close accuracy.

Our model processes queue length distributions through a specialized transformer architecture designed for sequential prediction. Given a sequence of probability distributions representing queue lengths over time, we first apply a Distribution Sequence Embedding module that transforms the probability vectors into continuous representations suitable for neural network processing. This module performs parallel transformations (logit and log-probability mappings) that capture different aspects of the distributional information. These transformed features are normalized, concatenated, and projected through a multi-layer perceptron (MLP) to create initial embeddings that encode both the shape and scale of each distribution.

The embedded sequence is then processed by a stack of transformer blocks, each consisting of causal self-attention followed by a feedforward MLP with residual connections. We incorporate rotary positional embeddings to encode chronological order, allowing the model to learn time-dependent patterns in queue evolution. We employ a multi-objective loss that combines multiple statistical discrepancies, including cross-entropy, Mean squared error (MSE), Wasserstein-1 distance, and continuous ranked probability score (CRPS). The losses are combined through learned uncertainty weighting using the same method as Kendall et al. [2018]:

$$\mathcal{L} = \sum_i \frac{1}{2\sigma_i^2}\mathcal{L}_i + \frac{1}{2}\log \sigma_i^2, \tag{3}$$

where $\mathcal{L}_i$ denotes each different loss function we have and $\{\sigma_i\}$ are learnable parameters that adaptively balance different error metrics in Kendall et al. [2018]. Additional details about our transformer architecture are provided in Appendix A.

4 Numerical Experiments: Training and Validation

In this section, we describe the generated datasets used to train the model and demonstrate the training process and the empirical results we achieved. We generate training datasets by simulating queueing systems and extracting queue length distributions over time. Specifically, we simulate multiple sample paths from $G/G/1$ queues (with stationary arrival and service processes) and $G_t/G/1$ queues (with time-varying arrival processes). For each simulation run, we record the queue length at regular time intervals. By aggregating these observations across many independent replications, we obtain empirical queue length distributions at each time point t. Each training instance is generated from a different queue configuration with distinct interarrival and service distributions and has 70 time steps. We use the first 32 time steps as input during training, while time steps 33-70 serve as held-out data for testing the model's autoregressive prediction capabilities. Additional details about the synthetic data generation process are provided in Appendix B.

Figure 1 visualizes the transformer's predictive performance on queue length distributions over time for a $G_t/G/1$ queue. In these heatmaps, each column represents the vector $\mathbf{X}_t$ at time t and each cell at position (t, i) represents the probability of queue lenght equal to i at time t: $\mathbf{X}_t^i = \mathbb{P}(L_t = i)$, with color intensity indicating probability mass. Brighter colors denote higher probabilities. The left panel shows the ground-truth synthetic data, while the right panel displays the transformer's autoregressive

predictions. The vertical line separates the context window (timesteps 1-32, used as input) from the prediction horizon (timesteps 33-70, generated autoregressively).

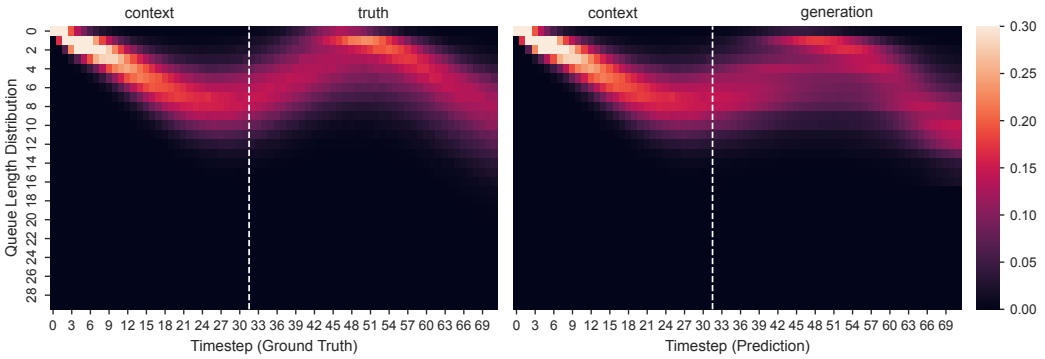

Figure 1: Ground-truth (left) versus transformer-predicted (right) queue length distributions

In Figure 1, we observe that the transformer demonstrates remarkable accuracy in capturing the queueing dynamics. It faithfully reproduces the overall pattern of the queue length distribution that evolves over time. Particularly noteworthy is the model's ability to maintain prediction quality even at timestep 70. This long-range prediction capability, which is traditionally challenging for sequence models, highlights the transformer's effectiveness in learning the underlying stochastic dynamics. Additional visualizations for different queueing configurations are provided in Appendix C, which further confirms the strength of our transformer architecture.

Apart from demonstrating the individual prediction accuracy, we compare our transformer model with other sequence model baselines in deep learning. Figure 2 shows the curves for both total loss and MSE loss in testing, which compares our transformer architecture with a Gated Recurrent Unit (GRU), a widely-used RNN variant for sequence modeling and autoregression. The transformer consistently achieves lower losses across all training epochs, with the performance gap on total loss widening as training progresses. We speculate that the transformer can outperform the RNN structure because its attention mechanism can capture long-range time-dependent patterns (which are common in queueing systems) that RNNs struggle to model effectively.

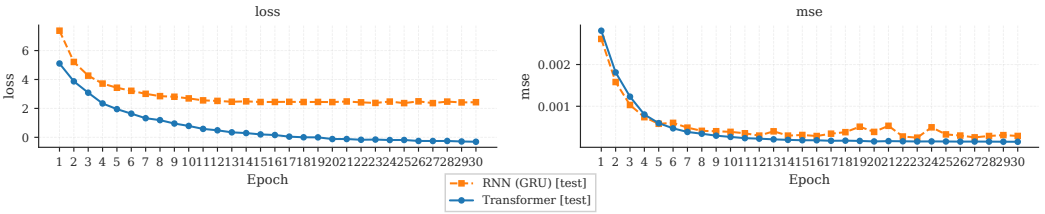

Figure 2: Testing loss comparisons between transformers and RNN (GRU)

5 Conclusion and Future Directions

We present a data-driven approach to learning real-life queueing dynamics using decoder-only Transformer. By exploiting the inherently autoregressive structure, our method learns to predict next-step queue length distributions directly from historical observations. Experiments on synthetic data from various queueing systems demonstrate that our transformer-based approach accurately reproduces queue dynamics and shows better performance than other sequence prediction model. For future work, we intend to establish theoretical guarantees for the transformer's predictive performance in queueing systems. We also plan to validate our approach on real-world ride-hailing datasets discussed in Alwan et al. [2024]. Ultimately, we hope that the success of our approach opens new possibilities for data-driven decision-making in operational contexts without requiring explicit model specification or costly simulations.

References

Amir Alwan, Baris Ata, and Yuwei Zhou. A queueing model of dynamic pricing and dispatch control for ride-hailing systems incorporating travel times. *Queueing Systems*, 106:1–66, 02 2024.

Jim Dai and Mark Gluzman. Queueing network controls via deep reinforcement learning. *Stochastic Systems*, 12(1):30–67, 2022.

Huiwen Jia, Cong Shi, and Siqian Shen. Online learning and pricing for service systems with reusable resources. *Operations Research*, 72(3):1203–1241, 2024.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Yueying Li, Jim Dai, and Tianyi Peng. Throughput-optimal scheduling algorithms for LLM inference and AI agents, 2025. Preprint.

Daksh Mittal, Zheng Shunri, Dong Jing, and Namkoong Hongseok. Data-driven stochastic modeling using autoregressive sequence models: Translating event tables to queueing dynamics, 2025. Preprint.

Michael Mitzenmacher and Rana Shahout. Queueing, predictions, and large language models: Challenges and open problems. *Stochastic Systems*, 15(3):195–219, 2025.

César Ojeda, Kostadin Cvejoski, Bogdan Georgiev, Christian Bauckhage, Jannis Schücker, and Ramsés J. Sánchez. Learning deep generative models for queuing systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2021.

Sergio Palomo and Jamol Pender. Learning lindley's recursion. In *Proceedings of the Winter Simulation Conference*, page 644–655, 2021.

Eliran Sherzer, Opher Baron, Dmitry Krass, and Yehezkel Resheff. Approximating $G(t)/GI/1$ queues with deep learning. *European Journal of Operational Research*, 322, 12 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Neil Walton and Kuang Xu. Learning and information in stochastic networks and queues. *INFORMS TutORials in Operations Research*, pages 161–198, 2021.

A Transformer Architecture

In this section, we introduce our transformer architecture in detail.

The Distribution Sequence Embedding two parallel transformations: Logit transformation and Log-probability transformation. We apply a logit mapping with clamping for numerical stability for the logit transformation:

$$\mathbf{z}_t = \log\left(\frac{\mathbf{X}_t}{1-\mathbf{X}_t}\right),$$

and we compute $\mathbf{u}_t = \log(\mathbf{x}_t)$ for log-probability transformation. Both $\mathbf{z}_t$ and $\mathbf{u}_t$ are independently normalized by Layer Normalization and concatenated. The concatenated vector is projected through a two-layer MLP:

$$\mathbf{h}_t^{(0)} = \text{MLP}\big(\big[\text{LN}(\mathbf{z}_t),\ \text{LN}(\mathbf{u}_t)\big]\big) \in \mathbb{R}^h \text{ where } h \text{ is the embeding dimension,}$$

yielding the initial embedding sequence $\mathbf{H}^{(0)} = (\mathbf{h}_1^{(0)}, \ldots, \mathbf{h}_T^{(0)})$.

The embedded sequence is processed by a stack of n Transformer-style blocks, each consisting of a causal self-attention layer followed by a feedforward MLP:

$$\mathbf{H}^{(\ell)'} = \mathbf{H}^{(\ell)} + \text{Attn}\big(\text{LN}(\mathbf{H}^{(\ell)})\big),$$

$$\mathbf{H}^{(\ell+1)} = \mathbf{H}^{(\ell)'} + \text{MLP}\big(\text{LN}(\mathbf{H}^{(\ell)'})\big),$$

for $\ell = 0, \ldots, n-1$. To encode chronological order for input data, we use Rotary Positional Embeddings (RoPE) within the self-attention block via a trainable skew-symmetric generator $A \in SO(n)$. At each time point t, the rotation matrix is defined as

$$R_t = \exp(tA) \in SO(n).$$

Queries and keys are rotated accordingly:

$$\mathbf{q}_t = R_t \mathbf{q}'_t, \quad \mathbf{k}_t = R_t \mathbf{k}'_t.$$

The attention mechanism follows the masked softmax rule:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}} + M\right) V,$$

where d_h is the head dimension and M is a causal mask ensuring autoregressive conditioning.

B Data Generation for Synthetic Datasets

In this section, we describe the data generation process for our experiments.

We construct two sets of synthetic datasets to evaluate the performance of our proposed approach: one with homogeneous arrivals and another with non-homogeneous arrivals. In both cases, we focus on a single-server queue model (i.e., $G/G/1$ and $G_t/G/1$). We detail the generation procedure for each setting separately.

Data generation for $G/G/1$. We consider seven distributions to generate both inter-arrival times and service times. The distributions and their parameterizations are summarized in Table 1. For parameter generation, $\mathrm{Unif}(0,1)$ denotes a uniform random variable on the interval $[0,1]$, and $\mathrm{Cat}(S)$ denotes a random variable that is uniformly chosen from set S. The distribution for generating inter-arrival and service times is selected at random from those listed in Table 1.

Distribution	Parameters	Generation
Gamma	(shape, scale)	(1+4Unif(0,1), Unif(0,1))
Erlang	(k, rate)	(Cat($\{2,3,4,5\}$), 1+4Unif(0,1))
Lognormal	(mean, variance)	(-Unif(0,1), Unif(0,1))
Mixture of two normals	(mean1, variance1, mean2, variance2, p_1)	(Unif(0,1), Unif(0,1), Unif(0,1), Unif(0,1), Unif(0,1))
Hyper exponential	(rate1, rate2, p_1)	(Unif(0,1), 1+Unif(0,1), Unif(0,1))
Uniform	(a, b)	(Unif(0,1), 1+Unif(0,1))
Weibull	(shape, scale)	(2Unif(0,1), 2Unif(0,1))

Table 1: Distributions used to generate inter-arrival and service times. The "Generation" column specifies how the parameters are sampled for each distribution. Note that we use p_1 to denote the probability of the first distribution in the mixture of two normals and the hyper exponential distributions.

Data generation for $G_t/G/1$. The data generation procedure for $G_t/G/1$ is similar to that of $G/G/1$, except that the inter-arrival times now exhibit periodic variation. The periodicity is introduced through sinusoidal functions. Specifically, Table 2 presents the candidate distributions along with their time-varying parameters. Here, T denotes the cycle length.

Distribution	Time-varying parameters	Generation
Gamma	shape	$5\mathrm{Unif}(0,1) + 5\mathrm{Unif}(0,1) \times \sin(2\pi t/T)$
Erlang	rate	$1 + 4\mathrm{Unif}(0,1) + (1 + 4\mathrm{Unif}(0,1)) \times \sin(2\pi t/T)$
Lognormal	mean	$\mathrm{Unif}(0,1) + \mathrm{Unif}(0,1) \times \sin(2\pi t/T)$
Mixture of two normals	p_1	$\mathrm{Unif}(0,1) + \mathrm{Unif}(0,1) \times \sin(2\pi t/T)$
Hyper exponential	p_1	$\mathrm{Unif}(0,1) + \mathrm{Unif}(0,1) \times \sin(2\pi t/T)$
Uniform	(a, b)	$(\mathrm{Unif}(0,1) + \mathrm{Unif}(0,1) \times \cos(2\pi t/T), a + \mathrm{Unif}(0,1))$

Table 2: Distributions used to generate inter-arrival and service times for $G_t/G/1$. The "Generation" column specifies how the time-varying parameters are sampled. For parameters that remain fixed, values are generated as in Table 1

To ensure that the resulting systems remain stable, we restrict our attention to the parameters of both inter-arrival and service time distributions such that expected utilization lies between 0.26 and 0.6.

With the two datasets generated as described above, we conduct simulations for each test case and record the distribution of queue lengths based on multiple replications. Specifically, we generate N arrivals sequentially with their associated inter-arrival and service times, determine the queue length using the recursion (as introduced in (1)), and record the queue length at each time interval Δt for every replication. The queue length distribution is then obtained from the average queue length across R replications. In our experiments, we set $N = 500$, $\Delta t = 1$, and $R = 10,000$.

C Additional Visualizations from Numerical Experiments

In this section, we present additional visualizations from our training and testing instances. We use two different queueing configurations (where data is generated as discussed in Appendix B) to test the strength of our learning model using a transformer architecture.

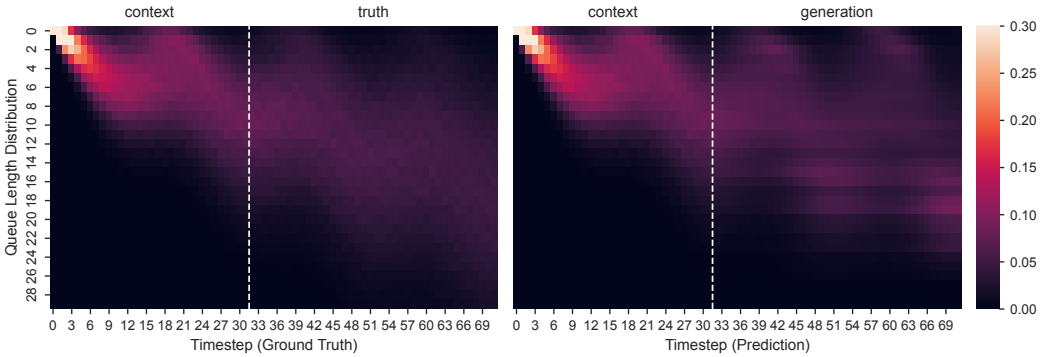

Figure 3: Ground-truth (left) versus transformer-predicted (right) queue length distributions under configuration 1

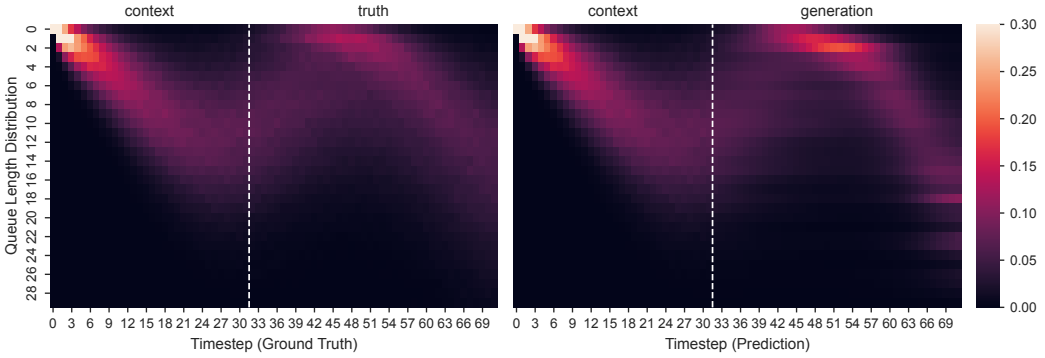

Figure 4: Ground-truth (left) versus transformer-predicted (right) queue length distributions configuration 2

From Figures 3 and 4, we observe that our autoregressive model captures the evolving dynamics for a queueing system with different configurations. This suggests the transformer's effectiveness in learning the underlying stochastic dynamics rather than merely memorizing patterns for special instances.