Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

SERVICE LEVEL PREDICTION IN NON-MARKOVIAN NONSTATIONARY QUEUES: A SIMULATION-BASED DEEP LEARNING APPROACH

Spyros Garyfallos[1], Yunan Liu[2], Pere Barlet-Ros[1], and Alberto Cabellos-Aparicio[1]

[1]Department of Computer Architecture, Universitat Politècnica de Catalunya, Barcelona, Catalunya, SPAIN
[2]Department of Industrial and Systems Engineering, North Carolina State University, Raleigh, NC, USA

ABSTRACT

Empirical studies have shown that real-life queueing systems, such as contact centers, exhibit non-Markovian and nonstationary behaviors. Consequently, analyzing their performance poses significant challenges. In this paper, we propose a *simulation-based autoregressive deep learning algorithm* (SADLA) for predicting service levels in non-Markovian, nonstationary queueing systems. Our method leverages modern recurrent neural networks, which are trained on synthetic data to capture the intrinsic spatio-temporal characteristics of queueing systems. Our findings demonstrate that SADLA achieves high prediction accuracy while reducing computational complexity by six orders of magnitude compared to traditional simulation methods. The implications of our research extend beyond accurate queue performance analysis; by embracing the learning capabilities of neural networks, our approach contributes to the advancement of the overall performance and resilience of real-life service systems.

1 INTRODUCTION

In general, queueing models can be broadly classified into two main categories: Markovian queues, characterized by Poisson arrivals and exponential service times, and non-Markovian queues, which involve nonexponential distributions. Recent empirical studies (Brown et al. 2005; Shi et al. 2014) have confirmed the practical realism of non-Markovian models, despite their analytical complexity. Emerging research emphasizes that it is important to incorporate these non-Markovian features in the analysis of such models because they can have high impacts on both the steady states (Aras et al. 2018; Whitt 2006a) and transient performance (Liu and Whitt 2012; Liu et al. 2016) of the queueing system. Moreover, real life services systems are rarely stationary; they follow the natural daily cycles of human patterns (e.g. high demand during daytime and conversely low demand during after-hours).

1.1 Challenges of Non-Markovian Nonstationary Queues

In *non-Markovian nonstationary* (NMNS) queues, one needs to capture both (i) stochastic variability, arising from complex probabilistic structures, and (ii) time variability, stemming from nonstationary model parameters like arrival rates. The former alone poses significant complexities and challenges, often defying analytic solutions because, unlike their Markovian counterparts, the system states now cannot be adequately represented by simple birth-and-death processes. For instance, to fully understand a multi-server $G/GI/n+GI$ queue with nonexponential service and abandonment times, it is necessary to track not only the total number of customers but also the elapsed waiting times of waiting customers and service times of those being served (Whitt 2006b; Liu and Whitt 2014b). Furthermore, describing queueing performance in nonstationary queues requires capturing transient dynamics over time, adding further complexity. After all, the performance analysis goes beyond computing steady-state distributions which do not exist; the transient performance, such as waiting time at a given time t, heavily depends on preceding values at $s < t$.

2024 Winter Simulation Conference (WSC) | 979-8-3315-3420-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/WSC63780.2024.10838828

To tackle the complexities of NMNS models, researchers commonly turn to approximation methods rooted in large-scale limits. One prominent approach is the fluid limit, which serves as a first-order approximation for these systems (Liu and Whitt 2012; Whitt 2006a; Liu and Whitt 2011; Liu and Whitt 2014a). The essence of a fluid model lies in capturing the temporal dynamics of the system while disregarding its stochastic variability. These limits are established through the functional law of large numbers (FLLN), necessitating a significant scaling up of the system's demand and service capacity. In essence, a fluid model conceptualizes all customers as infinitesimal units of fluid, allowing deterministic functions at the system level to represent their collective behavior. The computation of fluid models for non-Markovian queues often involves numerically solving systems of differential equations. However, fluid approximations degrade when the system size is not too large, and its deterministic nature prohibits it from capturing distributional information of the queueing system (e.g., the probability that the waiting is above a target). Diffusion limits (Liu and Whitt 2014b; Liu 2018; Liu et al. 2022; He 2020) offer stochastic refinements for the deterministic fluid models but are challenging to develop and implement, particularly for queueing systems with nonexponential service times (Aras et al. 2018; Liu et al. 2016).

Computer simulation methods have been widely used in practice for modeling NMNS queueing systems (Ma and Whitt 2016). However, they also come with certain disadvantages. First, simulating non-Markovian queueing systems requires detailed discrete-event simulations to track all event clocks (convenient methods for Markovian models such as uniformization do not apply), so it can be computationally intensive and requires significant computational resources and time, particularly for systems with large numbers of customers or complex interactions (e.g. multi-skill transfers and priority queues). Second, like any modeling approach, simulation methods rely on assumptions about system parameters, input distributions, and operational characteristics. Errors in these settings can introduce bias or inaccuracies into the simulation results. Last, exploring the effects of different system parameters or configurations may be more challenging with simulation methods because conducting sensitivity analyses or exploring a wide range of scenarios can be time-consuming and may require extensive computational resources.

In this paper, we contribute to the performance analysis of NMNS queueing systems having non-Markovian probability structure, nonstationary arrival process, and time-varying staffing level. We propose a *simulation-based autoregressive deep learning algorithm* (SADLA) for predicting desired transient system dynamics as a function of time (e.g., the mean queue length $\mathbb{E}[Q(t)]$ and the probability the waiting time violates a target $\mathbb{P}(W(t) > \tau)$). Our ideas are inspired by the recent successes of deep learning in in spaces where no precise analytic methods exist today, such as natural language processing (Gregor et al. 2015; Graves and Jaitly 2014). Specifically, we draw inspirations from the idea of recurrent neural networks (RNNs) and their applications in sequentially dependent information. Starting from the most recent observed system state, the use of RNNs allows SADLA to autoregressively predict any length of future horizons where the input system parameters can be estimated. This, combined with its low computational complexity, SADLA is ideal for near real-time application in practical NMNS queueing systems.

1.2 A Motivating Example

We first give a quick illustration of the performance of SADLA. We consider a multi-server $M_t/H_2/n_t + E_2$ model, having a nonhomogeneous Poisson arrival process with non-stationary arrival rate $\lambda(t)$ (top panels of Figure 1), non-stationary number of servers $n(t)$ (second panels of Figure 1), service times following a hyperexponential (H_2) distribution with mean $1/\mu = 1$, and abandonment times following an Erlang-2 (E_2) distribution with mean $1/\theta = 1$. Specifically, the *probability density functions* (PDFs) for the abandonment and service times are:

$$f_a(x) = 4\theta^2 x e^{-2\theta x} \qquad \text{and} \qquad g(x) = p\mu_1 e^{-\mu_1 x} + (1-p)\mu_2 e^{-\mu_2 x},$$

where $p = 0.5(1-\sqrt{0.6})$, $\mu_1 = 2p$, and $\mu_2 = 2(1-p)$.

In Figure 1 we graph the transient trajectories of common performance metrics including: the mean queue length, mean number of busy servers, mean waiting time, and *tail probability of delay* (TPoD),

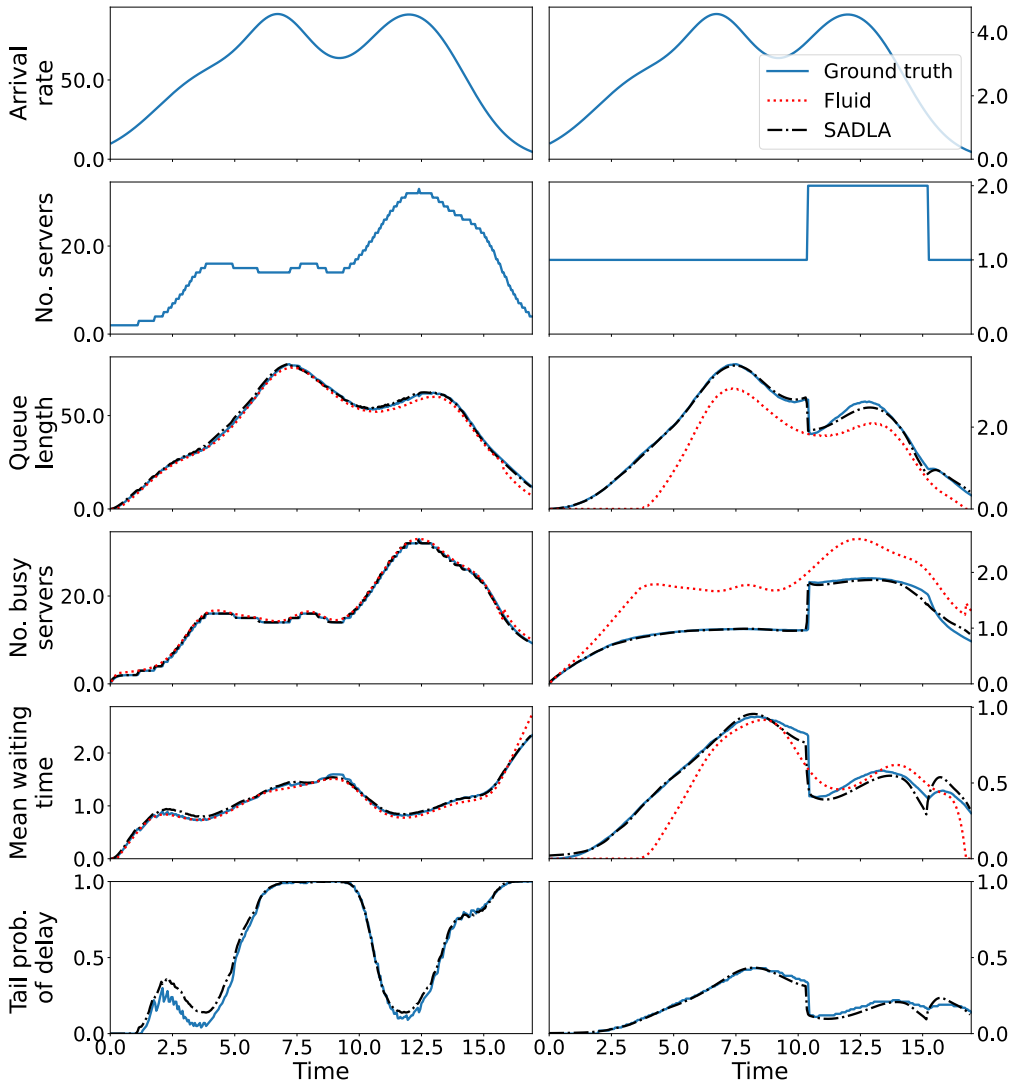

Figure 1: Performance comparison in an $M_t/H_2/n_t + E_2$ example: (i) SADLA (intermittent lines), (ii) ground truth (solid lines), and (iii) fluid model (dotted lines), in high-scale (left panels) and low-scale (right panels) settings. Fluid model is unable to compute the tail probability of delay (bottom plots).

i.e., the probability the waiting time violates a designated target $\tau > 0$: $\mathbb{P}(W(t) > \tau)$ (with $\tau = 1$ here). We consider two different system scales: a high-scale case with higher arrival rate and staffing level (left panels), and a low-scale case with lower arrival and staffing level (right panels). We compare our method's predictions (intermittent lines) to the fluid approximation, and the crude Monte-Carlo (MC) simulations (hereby referred to as the "ground truth"). We note the limitation of the fluid model to accurately capture the performance of the low-scale system, and to compute performance metrics beyond their "averaged-value" such as the TPoD, a widely used service-level function in today's service systems (Liu et al. 2022); see right-hand panel and bottom plots in Figure 1 respectively. We conduct additional numerical experiments in Section 3 to further evaluate the performance of SADLA. Also, we provide in-depth discussions on additional advantages of our method beyond its accuracy in Section 4.

1.3 Summary of Contributions

We summarize the main contributions of this paper:

i) SADLA is able to autoregressively predict non-Markovian nonstationary systems' transient performance functions (e.g., waiting time and queue length) in a step-wise lookahead manner, taking advantage of the intrinsic characteristics of the system spatio-temporal dynamics. This step-wise lookahead property of our method enables applications of real-time memoryless predictions: while the queueing system at hand can be non-Markovian that requires detailed past information, our method depends only on the most recent snapshot state of the system's state. The snapshot state representation and the autoregressive feature of our method can stimulate future research beyond the scope of performance analysis, such as real-time optimization for the staffing function.

ii) We confirm the effectiveness of our method by conducting comprehensive numerical experiments with non-Markovian systems with time-varying arrivals and staffing. Besides the mean waiting time, SADLA is able to compute other service-level metrics predominantly considered in service systems, such as the number of busy servers, mean queue length, and TPoD. Our results show that SADLA yields high-fidelity solutions that outperform the heavy-traffic fluid approximations, while it is six orders of magnitude computationally more efficient than Monte-Carlo simulations.

1.4 Related Literature

This research is related to three bodies of literature.

NMNS queues. Non-Markovian queueing systems pose greater challenges compared to Markovian counterparts. Mandelbaum et al. (1998) laid the groundwork for transient performance analysis in Markovian queueing networks, yet analyzing non-Markovian models remains notably complex. Whitt (2006b) introduced a groundbreaking fluid $G/G/n+G$ model to address queues with nonexponential service and abandonment times, pioneering a new research direction. This model's scope has expanded in subsequent works: Liu and Whitt (2012) incorporated time-varying arrivals and staffing levels, Aras et al. (2017) extended it to infinite-server queues, and Liu and Whitt (2011), Liu and Whitt (2014a) explored network structures and queues with transitory arrivals. Refinements to these fluid models have been pursued through stochastic FCLT limits, including time-varying OU processes (Liu and Whitt 2014b), patience-time scaled diffusion approximation (He 2020), and Gaussian approximations for stationary overloaded queues (Liu et al. 2016; Aras et al. 2018). In contrast to existing literature relying on heavy-traffic limits for non-Markovian models, this paper introduces a novel data-driven deep learning approach.

Integrating machine learning into queueing theory. Machine learning techniques are increasingly applied to model and analyze queueing systems, though the literature in this area is still relatively small. Recent studies have explored the use of online learning and reinforcement learning methodologies to support real-time decision-making in queueing systems. These include pricing strategies (Jia et al. 2024), capacity sizing (Chen et al. 2023; Chen et al. 2024), and control policies like routing (Liu et al. 2019; Shah et al. 2020) and scheduling (Dai and Gluzman 2021; Krishnasamy et al. 2021). This paper aligns more closely with recent research employing neural networks to study queueing systems. Baron et al. (2022) utilized quasi-birth-death queues to train a neural network to estimate steady-state probabilities in the M/G/1 queue. Ata et al. (2024) applied neural networks to solve Hamilton-Jacobian-Bell equations in queueing control problems. Rusek et al. (2020) used neural networks to investigate deterministic telecommunication queueing networks. Differentiating from prior work, this paper introduces a deep learning approach for NMNS queues. The goal is to characterize transient performance in a step-wise lookahead manner, drawing inspiration from RNNs.

Recurrent neural networks. The focus of research on RNNs revolves around their capability to model sequential data by retaining hidden states that effectively capture temporal dependencies. Recent advancements in this field explore sophisticated architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). These architectures aim to address the vanishing gradient problem, where the impact of past observations diminishes with each recurrence. By doing so, they enhance the network's ability to learn long-term dependencies. The versatility of RNNs is evident across various domains, including natural language processing, time series prediction, and generative tasks. They excel

in encapsulating intricate sequential patterns (Karpathy et al. 2015; Chung et al. 2015). More recently, transformer architectures (Vaswani et al. 2017) have emerged as a modern and robust alternative. Unlike RNNs, transformers do not require maintaining a hidden state, enabling higher training parallelism and the ability to learn complex spatial relationships beyond temporal order. Distinguishing itself from previous literature, this paper introduces an RNN applied to transient queueing performance analysis. Notably, it utilizes only snapshot state information of the system, differing from the sequential processing characteristic of traditional RNNs.

2 METHODOLOGY

We consider the $G_t/GI/n_t + GI$ NMNS queueing model having a NMNS arrival process (the G_t) with rate $\lambda(t)$, time-varying staffing level (i.e., number of servers) $n(t)$ (the n_t), *independent and identically distributed* (I.I.D.) service times following a general distribution G (the first GI), and customer abandonment according to I.I.D. random variables following a general distribution F (the $+GI$). Our goal is to characterize the transient dynamics by computing the time-indexed trajectory of queueing performance metric, denoted by S_t, for $t \in [0,T]$. Examples of S_t include the mean waiting time, mean queue length, and TPoD.

The transient analysis is complex and challenging because the dynamics at time t is determined by several factors such as the previous system state (because existing customers will carry over at future times) and present model inputs such as arrival rate and staffing level. Our proposed deep learning method, called SADLA, can recursively predict the next-step state S_{t+1} in a discrete time setting using (i) present state S_t, and (ii) next-step model input β_{t+1} (e.g., $\lambda(t+1)$ and $n(t)$). Specifically, the heart of SADLA is in form of

$$S_{t+1} = \mathscr{F}(S_t, \beta_{t+1}), \quad t = 0,1,2,\ldots, \tag{1}$$

where the function $\mathscr{F}$ is trained using deep learning models, which will be introduced later. Comparing to conventional queueing theory methodologies where one needs to carefully transform arrival rate and staffing level to designated queueing performance, SADLA leans to give solutions via an end-to-end approach because it directly focus on the desired system metrics. As illustrated in Figure 2, SADLA follows three steps: (i) identifying proper bases of nonstationary model parameters within the parameter space, (ii) using these parameter bases to generate sufficient system trajectories via computer simulations, and (iii) using the simulation results as training data to construct our neural network prediction model. We next elaborate on these steps in details.

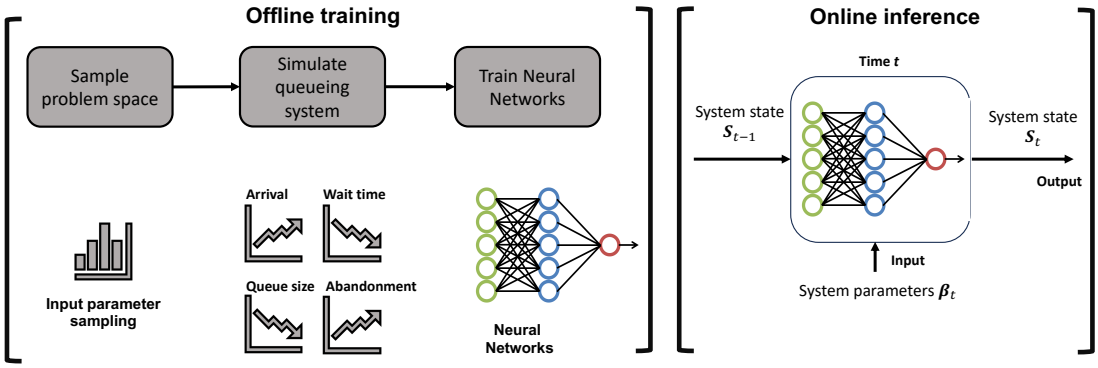

Figure 2: The pipeline of our method.

Let $\mathscr{B}$ be the space of all nonstationary parameters. We assume $\mathscr{B}$ is finite and large enough to contain all possible values of these parameters. For the example of the arrival rate and number of servers, $\mathscr{B} = [\underline{\lambda}, \overline{\lambda}] \times [\underline{n}, \overline{n}]$ for some $\underline{\lambda}, \underline{n} \geq 0$, and $\overline{\lambda}, \overline{n} < \infty$. Using the $\mathscr{B}$ basis, we are able to generate temporal trajectories of these parameters for computer simulation. We use a time series composition using additive

Gaussian processes (Bilancia et al. 2021), giving us a training data sampling method which we use to generate our simulated training data. The motivation for this trajectory data generation approach is that historical data of actual queueing systems can be used to extract the Gaussian process parameters' distributions, leading to synthetic data that resemble the actual system.

Next, we use the generated system parameter trajectories in Monte-Carlo simulation for generating the training data. For each simulation scenario, we begin with an empty system and obtain the simulated results of desired performance metrics (e.g., mean waiting time) over a set of N independent simulation paths. We operate in a finite time interval $[0,T]$ (e.g., $T = 17$ in Figure 1), and discretize time with step size Δt. The total number of simulated trajectories depends on the complexity and dimensionality of the system (e.g. total number of parameters of $\mathscr{B}$). In section 3 we provide quantitative studies on the total number of training data required for successfully fitting a two-parameter queueing system, $\lambda(t)$ and $n(t)$.

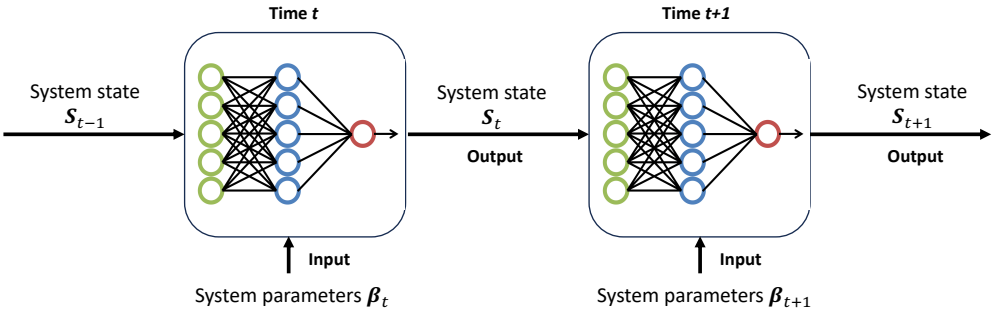

Figure 3: The feedforward topology of time-indexed RNNs.

The heart of our deep learning method builds on an autoregressive prediction procedure $\mathscr{F}$ (see equation (1)), where the prediction at $t+1$ draws from that at t. This sequential temporal dependency of the training information motivates us to leverage the recurrent neural network (RNN) architecture that will be trained to autoregressively predict the desired future performance metrics. To train our neural networks, we use the simulated trajectories from the above step, extracting as input features the system parameters β_{t+1} at time $t+1$ and the system state S_t at time t, and the system state S_{t+1} as the predicted (output) labels. Figure 3 provides a schematic illustration of our recursive method, formed as an unrolled (unfolded) computational graph into a full network.

To accurately learn how a complex queueing system behaves, it is important to find the proper way to represent its state S_t. On the one hand, this representation should give us sufficient information to describe the system's dynamics. For example, in NMNS models, we usually need to keep track additional information beyond the queue length, such as customers' ages in queue and in service (see (Whitt 2006a)). On the other hand, the increased dimensionality of the state representation will lead to higher model scales and efforts for training our neural networks. RNNs have the capability to encode and store an internal state representation, given a sufficiently long past context window something that allows them to reconstruct this internal representation.

We experiment with different permutations of performance metrics as model features to find the most expressive combination that can most effectively represents the system's internal state. In our experiments, the most parsimonious system representation is $S_t \equiv (\mathbb{E}[W(t)], \mathbb{E}[Q(t)], \mathbb{E}[B(t)])$, which tracks the mean number of waiting customers, mean waiting time and server occupancy (i.e., fraction of busy servers) at t. Such a state representation captures information from both sides of the queue: When the system is overloaded (underloaded), $(\mathbb{E}[W(t)], \mathbb{E}[Q(t)])$ $((\mathbb{E}[W(t)], \mathbb{E}[B(t)]))$ is the predominant feature that describes the system's congestion level. While a one-sided descriptor might suffice in a large-scale model - where a positive waiting time implies occupancy close to 100% and a positive fraction of idle servers indicates 0 waiting time, small-scale systems require both descriptors to jointly determine waiting time. As depicted in the right-hand panel of Figure 1, the occupancy is not close to 100% even when the waiting time is strictly positive.

We construct two neural networks for the learning of two types of performance metrics in a queueing system: (i) queue-related features such as the numbers of waiting customers and busy servers; and (ii) wait-time related state. Let $S_t \equiv (S_t^Q, S_t^W)$, where S_t^Q and S_t^W represent queue related and wait-time related metrics. For example, if $S_t = (\mathbb{E}[Q(t)], \mathbb{E}[B(t)], \mathbb{E}[W(t)], \mathbb{P}(W(t) > \tau))$, then $S_t^Q = (\mathbb{E}[Q(t)], \mathbb{E}[B(t)])$ and $S_t^W = (\mathbb{E}[W(t)], \mathbb{P}(W(t) > \tau))$. Such a separation follows the distinct nature of these two types: The former are discrete counts that represent the population-level information of the system, while the latter are continuous quantities that characterize the individual customer-level experience. Nevertheless, queue contents and waiting times are often related via Little's law; see Kim and Whitt (2013). Below we specify the updating procedure for the two state representations:

$$\begin{aligned} S_{t+1}^Q &= S_t^Q + \mathscr{F}^Q(S_t, \beta_{t+1}), \\ S_{t+1}^W &= \mathscr{F}^W(S_{t+1}^Q, S_t^W, \beta_{t+1}), \qquad t = 0, 1, 2, \ldots, \end{aligned} \tag{2}$$

where $\mathscr{F}^Q$ and $\mathscr{F}^W$ denote the two neural networks.

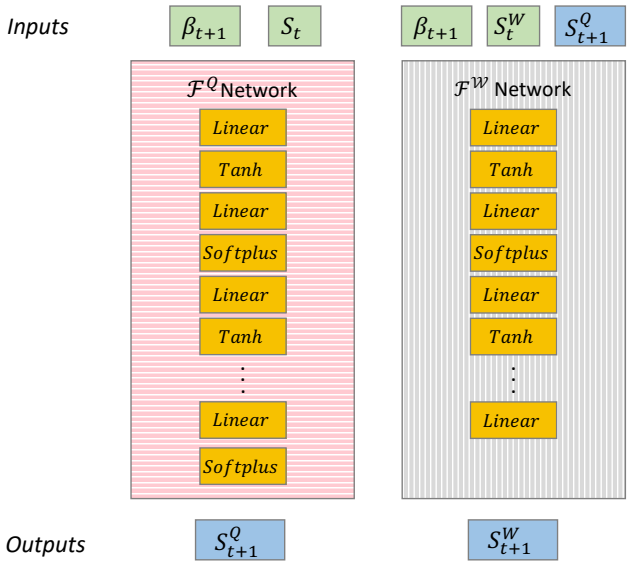

Figure 4: The architecture of the neural networks.

In Figure 4, we give an illustration for our two tandem neural networks' architecture as defined in (2), alongside with their corresponding inputs and outputs. We provide a high-level overview of the designed neural network architecture that consists of alternating linear layers with bias, Tanh and Softplus non-linearities. The $\mathscr{F}^Q$ network is designed to predict the ΔS_{t+1}^Q *difference* over the previous state, which is finally added to the S_t^Q. In that sense, the $\mathscr{F}^Q$ network is trained to predict a discretized ordinary first derivative of S^Q, conditional to the network's inputs. This is a manifestation that the S^Q state tends to *evolve towards its long-term equilibrium steady state*, and generally in a smooth way. For example, if β is kept stationary for a long time, the system will gradually evolve its S^Q towards the equilibrium state S_β^Q. $\mathscr{F}^W$ on the other hand is trained to *map* its inputs S_t^W, β_{t+1} and S_{t+1}^Q (the output of $\mathscr{F}^Q$) to the updated S_{t+1}^W state. This network represents a learned Little's Law function. This mapping design is justified by the fact that the differences between S_{t+1}^Q and S_{t+1}^W pairs are expected to be small. In other words, we do not expect to have two very different waiting time samples when all other system state is equal. These differences predominately exist in fast evolving β_{t+1}, such as sudden additions of another server that will cause a sudden drop of the waiting time.

3 NUMERICAL EXPERIMENTS

In this section we conduct a comprehensive set of numerical studies to evaluate the performance of SADLA. In Section 3.1, we describe the detailed settings of our numerical experiments and in Section 3.2 we describe how we train our neural networks. In Section 3.3, we apply our method to compute commonly used service-level metrics of an $M_t/GI/n_t+GI$ base example where we treat the fluid approximation as a performance benchmark. In Section 3.4 we treat a more challenging case where the system alternates between underloading and overloading.

3.1 Experiment Setting

Our model's training data are generated via computer simulations. We evaluate our method in demand and supply patterns of an $G_t/GI/n_t+GI$ system that resemble real-life service centers such as contact center queues. Our demand patterns are motivated by realistic settings having two peaks during a day (e.g., the call center arrivals of an Israeli call center (Brown et al. 2005)). Specifically, the time-varying arrival rate in our base example (as presented in Section 1) is constructed using a basis function B_k defined below:

$$B_k(t) = \frac{C}{\max\limits_{\{t\in T\}} P_k(t)} P_k(t) \quad \text{where} \quad P_k(t) = \sum_{n=1}^{k} \mathcal{N}_i(t) \quad \text{and} \quad \mathcal{N}_i(t) = \frac{c_i}{\sigma_i\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(t-\mu_i)^2}{\sigma_i^2}\right), \quad (3)$$

The parametric equation (3) is an additive composition of k parametric Gaussian processes, rescaled so that $B_k(t) \le C$. In our experiments, we assume prior knowledge of the bounds of the arrival rate $\lambda(t) \le 100$ and the number of servers $n(t) \le 100$. In addition, we assume that both demand and supply trajectories can decompose to $k = 10$ Gaussian bases. Under the above assumption, we generate random trajectories of $\lambda(t)$ and staffing level $n(t)$, where the parametric Gaussian components parameters c, μ and σ are drawn from predefined distributions. Specifically, for each Gaussian component i we draw $c_i \sim U(0.4,3) \times U(10,100)$ and $\sigma_i \sim U(0.4,3)$, where $U(a,b)$ denotes a uniform distribution over the interval (a,b). For composing the $\lambda(t)$ trajectories we use $\mu_i \sim U(4,14)$ and for $n(t)$ we use $\mu_i \sim U(2,16)$.

We then treat these trajectories as our data generation simulation scenarios. For each simulation scenario, we begin with an initially empty queueing system and obtain the Monte-Carlo estimated trajectories of the selected service-level statistics through $N_s = 5,000$ independent simulation paths. We operate in a finite time interval $[0,T]$ (e.g., $T = 17$ in Figure 1), and discretize time with step size $\Delta t \equiv 0.05$ time units.

3.2 Training Hyperparameters

We train the two networks of Figure 4 separately using simulation data. Separating the training process for the two tandem neural networks helps improve the model's overall explainability; this allows us to easily examine and measure their fitness, accuracy and limitations, and to fine-tune their hyperparameters for improved overall performance. The total number of simulation scenarios used in training and validation is 4000, split into a 80-20 ratio, producing 10^6 training samples. For training the $\mathscr{F}^W$ network we tune its hyperparameters to 10000 batch size, 5000 neurons width, 10^{-7} learning rate and 50 epochs. For the $\mathscr{F}^Q$ network we use smaller width of 2000 and 5×10^{-7} learning rate.

This hyperparameter tuning can seem peculiar when compared with other popular deep learning applications such as computer vision. This specific tuning is due to the following two main factors: on the one hand, the training dataset comes from Monte-Carlo simulations that are inherently noisy due to the involved stochasticity. This noise level could be mitigated by significantly increasing the number of simulated paths, something that reduces the data variance. Nevertheless, this comes with a significant increase of the computational cost. On the other hand, the size of involved training data are relatively small compared to other popular deep learning applications such as computer vision, something that would lead to fast overfitting of our networks, since each epoch would contain a very small number of total gradient calculations on noisy data. Our proposed hyperparameter tuning (very large batch sizes with very small

learning rates in combination with smooth non-linearities) allows our networks to mitigate both the inherent noise in the data and overfitting on the relatively small training dataset, while producing smooth outputs.

3.3 Effectiveness and Efficiency

Supplementing Figure 1, we next give quantitative measurements for our base example. In Table 1, we calculate the *scaled time-averaged mean absolute error* (STAMAE), defined as

$$\mathscr{E} \equiv \frac{\sum_{t=1}^{T} |y_t - \hat{y}_t|}{\frac{1}{T}\sum_{t=1}^{T} y_t}$$

for a service-level predictor $\hat{y}_t$ (with y_t its ground truth value) over the entire predictive horizon of T. We do so under two scenarios: high scale and low scale. Except for the TPoD which is beyond the capability of the fluid approximation, SADLA is able to outperform the fluid method by achieving a lower STAMAE, especially under the low scale example. It is evident that the solution accuracy degrades when the system scale is small, while SADLA is much more robust to the system scale; SADLA is a scale free method.

Table 1: Scaled time-averaged mean absolute error comparison between our method and fluid model.

	High scale		Low scale	
	SADLA	Fluid	SADLA	Fluid
Queue length	**0.008**	0.035	**0.025**	0.304
No. busy servers	**0.002**	0.031	**0.028**	0.520
Mean waiting time	**0.025**	0.043	**0.057**	0.203
Tail prob. of delay	**0.049**	NA	**0.078**	NA

Next, we evaluate its computational cost. We benchmark SADLA's running time with that of the crude Monte-Carlo simulations. To avoid unfairness due to the parallelism difference between CPUs and GPUs, in this comparison we evaluate SADLA on a CPU with the same specifications with the ones used for simulation. SADLA's average inference runtime is 0.35 seconds on a single CPU versus 900 seconds on a high performance computing system with 72 CPUs for simulation, six orders of magnitude computationally faster. For completeness, SADLA's offline training runtime for the experiments presented in this paper is approximately 15 minutes, while generating the simulated training data takes approximately 2 days.

3.4 Alternating Between Underloading and Overloading

Fluid models intend to characterize the system dyanmics in separate underloaded and overloaded intervals, so its solution accuracy often degrade as the system is transitioning between the two states (Liu and Whitt 2012). We next test SADLA's performance in such settings. We consider our base example with a sinusoidal arrival rate $\lambda(t) = 4\sin(\pi t(12-1/T)) + 10\sin(\pi t/T)$ and staffing function $n(t) = \lceil 4 + 4\sin(\pi t/T) \rceil$, with $T = 17$. In Figure 5 we plot the SADLA performance curves and again benchmark with their corresponding fluid approximations.

Because fluid approximations are deterministic functions, they ideally assume that either positive waiting times and all servers are busy, or zero waiting time and some servers are idle. As illustrated in Figure 5, fluid approximations significantly underestimate the waiting time and overestimate the number of busy servers. In contrast, SADLA provides much more accurate predictions for all performance metrics.

4 CONCLUDING REMARKS

Motivated by the need for tools to analyze the practical NMNS queueing systems, we develop a new deep learning method, called SADLA which can be used to compute the time-indexed transient trajectories of common service-level metrics such as the mean queue length, mean waiting time, and TPoD. Inspired by

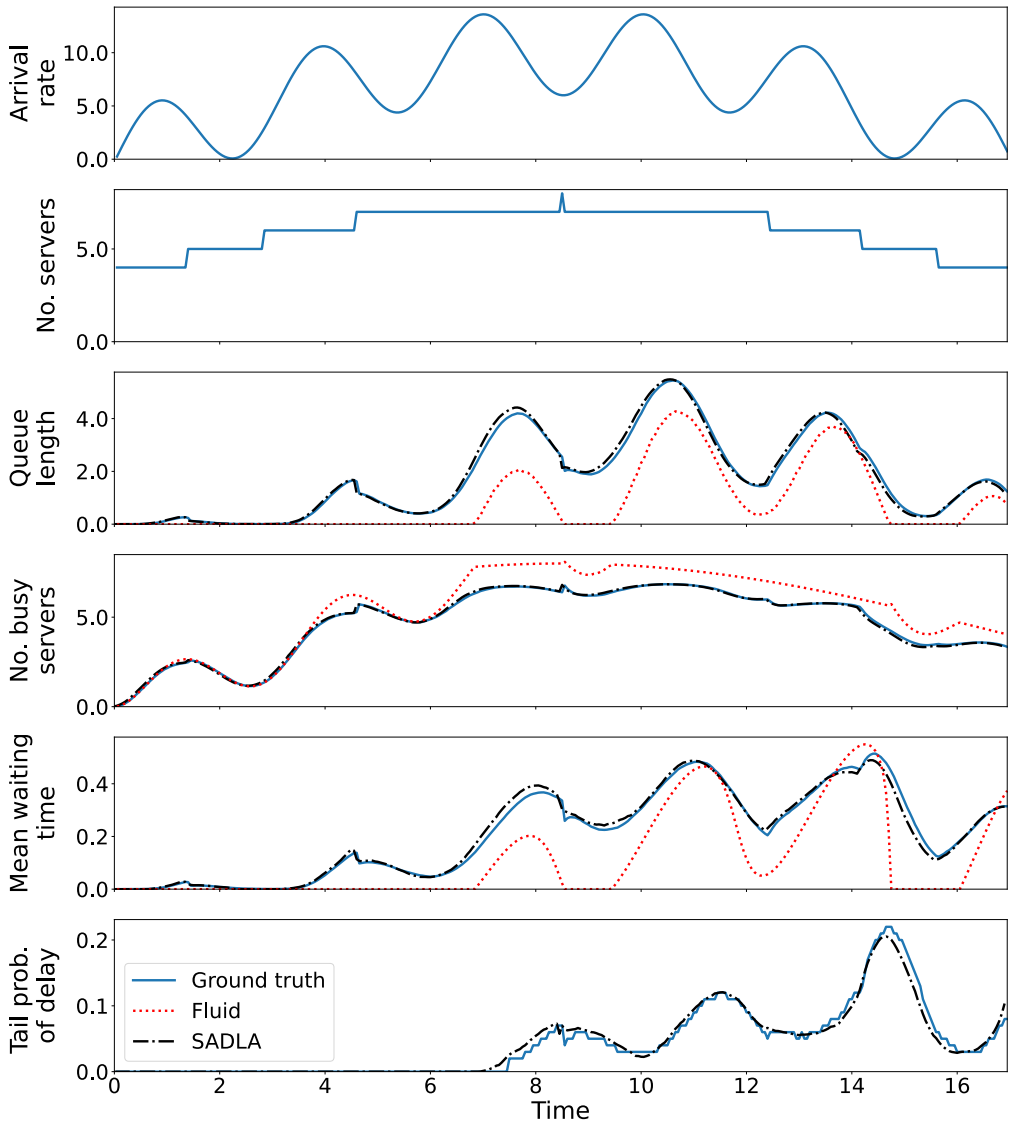

Figure 5: Performance comparison in an underloading and overloading alternating example: (i) SADLA (intermittent lines), (ii) ground truth (solid lines), and (iii) fluid model (dotted lines).

the ideas of recurrent neural networks, SADLA "predicts" the desired system state using the model input at the present step and predicted state at the previous time step. Our results confirm the effectiveness of SADLA and demonstrate that its advantages over the fluid approximations, the state-of-the-art analysis tool for NMNS models, especially in smaller scale systems. In addition, SADLA can be used to compute performance metrics that draw from distributional information beyond their means, such as the TPoD $\mathbb{P}(W(t) > \tau)$, the probability the waiting time exceeds a desired delay target $\tau > 0$, which is beyond the capability of the fluid approximations.

Limitations and future directions. First, SADLA aims to learn the transient performance for theoretic queueing models using their precise model information such as the arrival rate and service distributions; it is a simulation-based method which relies on a simulator of the theoretic model to generate the required training data. An interesting next step is to generalize our deep learning framework for real-world queueing models which can be trained and tuned by real data (e.g., via trace simulation) insteaded of

synthetic data. Second, as an initial attempt of integrating deep learning into queueing theory, the present version of SADLA emphasizes only treats the single-class non-Markovian nonstationary model. We plan to extend our methodology to more practical queues having multiple customer classes and network structures. This generalization requires carefully identifying the state representations. For example, in order to predict the performance at t of some station i in a feedforward network, besides its state information at $t-1$, we may also need the state information of station $i-1$ (i.e., its upstream station). We are particularly interested in using more modern deep learning architectures such as attention transformers for learning the dynamics of more complex queueing systems. Next, the snapshot feature of SADLA's state representation enables it for real-time applications, where SADLA's nearly instantaneous performance prediction can serve as building blocks for solving optimal staffing problems subject to designated service-level constraints. For example, if the predicted TPoD will violate some target, the practitioner can quickly adjust the staffing function in order to meet the desired target. Finally, another interesting future direction is to integrate other advanced deep learning techniques into SADLA for exploring generalization capabilities, such as transfer learning for fine tuning a pre-trained foundational model to specific queue characteristics.

REFERENCES

Aras, A. K., X. Chen, and Y. Liu. 2018. "Many-Server Gaussian Limits for Non-Markovian Queues with Customer Abandonment". *Queueing Systems* 89(1):81–125.

Aras, A. K., Y. Liu, and W. Whitt. 2017. "Heavy-Traffic Limit for Initial Content Process". *Stochastic Systems* 7(1):95–142.

Ata, B., J. M. Harrison, and N. Si. 2024. "Drift control of high-dimensional RBM: A computational method based on neural networks". *arXiv preprint arXiv:2309.11651*.

Baron, O., D. Krass, E. Sherzer, and A. Senderovich. 2022. "Can Machines Solve General Queueing Problems?". In *2022 Winter Simulation Conference (WSC)*, 2830–2841 https://doi.org/10.1109/WSC57314.2022.10015451.

Bilancia, M., F. Manca, and G. Sansaro. 2021. *A Time Series Decomposition Algorithm Based on Gaussian Processes*, 577–592. Springer International Publishing.

Brown, L., N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn *et al*. 2005. "Statistical analysis of a telephone call center: A queueing science perspective". *J. Amer. Statist. Assoc.* 100:36–50.

Chen, X., Y. Liu, and G. Hong. 2023. "An Online Learning Approach to Dynamic Pricing and Capacity Sizing in Service Systems". *Oper. Res.*. forthcoming.

Chen, X., Y. Liu, and G. Hong. 2024. "Online learning and optimization for queues with unknown demand curve and service distribution". *arXiv preprint arXiv:2303.03399*.

Chung, J., C. Gulcehre, K. Cho, and Y. Bengio. 2015. "Gated feedback recurrent neural networks". In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, Volume 37, 2067–2075.

Dai, J. G. and M. Gluzman. 2021. "Queueing network controls via deep reinforcement learning". *Stochastic Systems* 12(1):30–67.

Graves, A. and N. Jaitly. 2014. "Towards End-To-End Speech Recognition with Recurrent Neural Networks". In *Proceedings of the 31st International Conference on Machine Learning*, edited by E. P. Xing and T. Jebara, Volume 32 of *Proceedings of Machine Learning Research*, 1764–1772: PMLR.

Gregor, K., I. Danihelka, A. Graves, D. Rezende and D. Wierstra. 2015. "DRAW: A Recurrent Neural Network For Image Generation". In *Proceedings of the 32nd International Conference on Machine Learning*, edited by F. Bach and D. Blei, Volume 37 of *Proceedings of Machine Learning Research*, 1462–1471: PMLR.

He, S. 2020. "Diffusion Approximation for Efficiency-Driven Queues When Customers Are Patient". *Operations Research* 68(4):1265–1284.

Jia, H., C. Shi, and S. Shen. 2024. "Online Learning and pricing for Service Systems with Reusable Resources". *Oper. Res.* 72(3):1203–1241.

Karpathy, A., J. Johnson, and L. Fei-Fei. 2015. "Visualizing and understanding recurrent networks". *arXiv preprint arXiv:1506.02078*.

Kim, S.-H. and W. Whitt. 2013. "Estimating waiting times with the time-varying little's law". *Probability in the Engineering and Informational Sciences* 27(4):471–506.

Krishnasamy, S., R. Sen, R. Johari, and S. Shakkottai. 2021. "Learning unknown service rates in queues: A multiarmed bandit approach". *Operations Research* 69(1):315–330.

Liu, B., Q. Xie, and E. Modiano. 2019. "Reinforcement Learning for Optimal Control of Queueing Systems". In *57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 663–670.

Liu, Y. 2018. "Staffing to Stabilize the Tail Probability of Delay in Service Systems with Time-Varying Demand". *Operations Research* 66:514–534.

Liu, Y., X. Sun, and K. Hovey. 2022. "Scheduling to Differentiate Service in a Multiclass Service System". *Operations Research* 70(1):527–544.

Liu, Y. and W. Whitt. 2011. "A Network of Time-Varying Many-Server Fluid Queues with Customer Abandonment". *Operations Research* 59(4):835–846.

Liu, Y. and W. Whitt. 2012. "The $G_t/GI/s_t+GI$ many-server fluid queue". *Queueing Systems* 71(4):405–444.

Liu, Y. and W. Whitt. 2014a. "Algorithms for Time-Varying Networks of Many-Server Fluid Queues". *INFORMS J. on Computing* 26(1):59–73.

Liu, Y. and W. Whitt. 2014b. "Many-server heavy-traffic limits for queues with time-varying parameters". *Annals of Applied Probability* 24:378–421.

Liu, Y., W. Whitt, and Y. Yu. 2016. "Approximations for Heavily-Loaded $G/GI/n+GI$ Queues". *Naval Research Logistics* 63:187–217.

Ma, N. and W. Whitt. 2016. "Efficient simulation of non-Poisson non-stationary point processes to study queueing approximations". *Statistics & Probability Letters* 109:202–207.

Mandelbaum, A., W. A. Massey, and M. I. Reiman. 1998. "Strong approximations for Markovian service networks". *Queueing Systems* 30(1/2):149–201.

Rusek, K., J. Suarez-Varela, P. Almasan, P. Barlet-Ros and A. Cabellos-Aparicio. 2020. "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN". *IEEE Journal on Selected Areas in Communications* 38(10):2260–2270.

Shah, D., Q. Xie, and Z. Xu. 2020. "Stable Reinforcement Learning with Unbounded State Space". In *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, edited by A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Volume 120 of *Proceedings of Machine Learning Research*, 581–581: PMLR.

Shi, P., M. Chou, J. G. Dai, D. Ding and J. Sim. 2014. "Models and insights for hospital inpatient operations: Time-dependent ED boarding time". *Management Sci.* 62(1):1–28.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, *et al*. 2017. "Attention is All you Need". In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Volume 30: Curran Associates, Inc.

Whitt, W. 2006a. "Fluid Models for Multiserver Queues with Abandonments". *Operations Research* 54(1):37–54.

Whitt, W. 2006b. "A multi-class fluid model for a contact center with skill-based routing". *Aeu-international Journal of Electronics and Communications* 60:95–102.

AUTHOR BIOGRAPHIES

SPYROS GARYFALLOS received his Electrical Engineering and Computer Science diploma in 2004 (B.Sc. and M.Sc.) and MBA in 2007 from National Technological University of Athens (NTUA), and his M.Sc in Data Science from UC Berkeley, California in 2020. He is a Ph.D. candidate at the Computer Architecture Department of Universitat Politècnica de Catalunya in the space of modeling time varying queueing systems using deep learning. His email address is spiros.garifallos@gmail.com.

YUNAN LIU is an associate professor in the Department of Industrial and Systems Engineering at North Carolina State University. He earned his PhD in Operations Research from Columbia University. His research interests include queueing theory, stochastic modeling, simulation, applied probability, online learning, and optimal control, with applications to call centers, healthcare, and transportation. His work was awarded first place in the INFORMS Junior Faculty Interest Group Paper Competition in 2016. His email address is yliu48@ncsu.edu. His website is https://yunanliu.wordpress.ncsu.edu/.

PERE BARLET-ROS (PhD 2008) is a Professor with the Computer Architecture Department of the UPC and Scientific Director at the Barcelona Neural Networking Center (BNN-UPC). For the last 10 years, his research has focused on the development of novel machine learning technologies for network management and optimization, traffic classification and network security, which have been integrated in several open-source and commercial products. His email address is pere.barlet@upc.edu.

ALBERTO CABELLOS-APARICIO is a professor at the Computer Architecture Department, Universitat Politècnica de Catalunya. He is the co-founder of the Barcelona Neural Networking and the NaNoNetworking Center in Catalunya. He has been a visiting researcher at Cisco Systems and Agilent Technologies and a visiting professor at the Royal Institute of Technology, the Massachusetts Institute of Technology and UC Berkeley. He has participated in several national (Cicyt), EU (H2020), USA (NSF) and industrial projects. His email address is alberto.cabellos@upc.edu.