Learning Deep Generative Models for Queuing Systems

César Ojeda, [1] **Kostadin Cvejosky,** [2 3] **Bogdan Georgiev,** [2 3]
Christian Bauckhage, [2 3] **Jannis Schuecker,** [5] **Ramsés J. Sánchez** [2 4]

[1]Berlin Center for Machine Learning and TU Berlin, 10587 Berlin, Germany
[2]Competence Center Machine Learning Rhine-Ruhr
[3]Fraunhofer Center for Machine Learning and Fraunhofer IAIS, 53757 Sankt Augustin, Germany
[4]B-IT, University of Bonn, Bonn, Germany
[5]Bayer AG, Germany
ojedamarin@tu-berlin.de, jannis.schuecker@bayer.com, sanchez@bit.uni-bonn.de
{kostadin.cvejoski, bogdan.georgiev, christian.bauckhage}@iais.fraunhofer.de

Abstract

Modern society is heavily dependent on large scale client-server systems with applications ranging from Internet and Communication Services to sophisticated logistics and deployment of goods. To maintain and improve such a system, a careful study of client and server dynamics is needed – e.g. response/service times, average number of clients at given times, etc. To this end, one traditionally relies, within the queuing theory formalism, on parametric analysis and explicit distribution forms. However, parametric forms limit the model's expressiveness and could struggle on extensively large datasets.

We propose a novel data-driven approach towards queuing systems: the Deep Generative Service Times. Our methodology delivers a flexible and scalable model for service and response times. We leverage the representation capabilities of Recurrent Marked Point Processes for the temporal dynamics of clients, as well as Wasserstein Generative Adversarial Network techniques, to learn deep generative models which are able to represent complex conditional service time distributions. We provide extensive experimental analysis on both empirical and synthetic datasets, showing the effectiveness of the proposed models.

Introduction

The ultimate success of any service provider rests on its ability to quickly and efficiently satisfy its customers: a mobility system is only successful as long as its users arrive on time; a block-chain is reliable provided low latency of its transaction times is ensured; Internet services can only retain users if they provide quick and fast response. To operate systems like these, one needs to understand not only *when* customers will require a service but also *how* the system is able to react and respond to demands. Moreover, one should be able to dynamically adapt the system to external events. Examples include sudden disruptions of a mobility system due to car accidents or weather conditions, or financial crises and breaking news affecting block-chain transactions.

From the point of view of deep learning, recent research has primarily focused on the client side of a service system. This research analyses Customer Dynamics through Point Process Theory, in which user behavior is modeled via parametric forms such as Poisson or Hawkes processes. In cases where these parametric forms turn out to be restrictive, some practitioners turn to models such as Recurrent Neural Networks (RNN) in order to learn representations encoding the customer dynamics, and use Maximum Likelihood to fit them (Du et al. 2016; Mei and Eisner 2017; Jing and Smola 2017); while others dispense with density forms altogether, and learn instead how to sample from the distribution of interest via Generative Adversarial schemes (Xiao et al. 2017, 2018).

The other side of service system modelling deals with the service itself, in particular with service times, and mainly lies outside the scope of deep learning. This research typically resorts to Queuing Theory, in which a customer expresses a demand and the system responds, according to the available servers and server load. Within this theory, the customer arrival dynamics is modeled with a point process, while the service system is specified through either a single queue (with a given scheduling policy) or networks of them, and, yet again, parametric forms are assumed for both arrival and service time distributions. Results are usually limited to moments of the system size distribution (Asmussen 2008; Daw and Pender 2018; Boxma, Kella, and Mandjes 2019) or are, in some cases, based on Bayesian inference (Sutton and Jordan 2011; Wang, Casale, and Sutton 2016; Perez, Hodge, and Kypraios 2018). Yet, the latter often have strong built-in assumptions (e.g. Poisson arrival processes) which makes them too restrictive, and involve expensive sampling schemes, which renders them not scalable enough to handle the millions of customers in modern service systems.

The work presented here aims to provide a scalable algorithm for service time inference suitable to the large datasets of modern systems, and a first bridge between queuing system modelling and deep learning through deep generative models. In recent years Generative Adversarial Networks (GANs) have shown to be able to capture complex data distributions, achieving impressive results in image and text

generation (Goodfellow et al. 2014; Mirza and Osindero 2014; Lin et al. 2017). Here, instead of assuming the underlying queues and service disciplines within the systems we study, we use adversarial techniques to learn *conditional* service time distributions. The result is a simple methodology, flexible enough to serve in a wide range of applications.

In what follows we develop a deep non-parametric[1] generative model for service times. The model leverages dynamic representations of *general* arrival point processes, incorporates covariate information and yields a practical solution to the $G/G/\infty$ problem (see notation below) of queuing theory. We evaluate our methodology on both, synthetic datasets simulated from known queueing models and various real-world datasets, ranging from Internet services (GitHub, Stackoverflow) to mobility service systems (New York taxi cab). We compare our model against other deep non-adversarial service time models, as well as against exact theoretical estimates of the queue length.

Background

The theory of queues deals with the study of the distribution of *service times* and *queue lengths in service systems*. One would like to know, for example, how much time a client is likely to wait for a service, or how much time this service is expected to last. It corresponds to a central topic in the field of operations research, as it is of fundamental interest to efficiently allocate time and resources in a given service system. Historically, the field emerged from the studies related to telephone exchange and call arrivals (Erlang 1909).

In order to define a queuing system one must specify the nature of the client arrival process, as well as the specifics of the service system. The arrival of clients may follow a Poisson process, but its rate may jump due to external events. Likewise, one system might allow for a fixed number of clients to be served at a time, or the service times might dynamically change with every incoming client. The standard notation used to specify the characteristics of the different queuing systems consists of characters separated by slashes thus $\cdot/\cdot/\cdot$ (Kendall 1953). The first character describes the customer arrival process, namely the inter-arrival distribution. Typical examples are "M" for memoryless (Poisson), "D" for deterministic times and "G" for general distributions. The second character specifies the service time distribution, and the third one the number of servers available to the system. For example, the $M/M/1$ queue denotes a queuing system with Poisson arrivals, exponentially distributed service times and a single server. Various scheduling policies or service disciplines can also be employed by a given queue. For example, the customers can be served one at a time, in the order that they arrive (first-come first-served discipline); or they can all be served simultaneously, each receiving an equal fraction of the server capacity (processor sharing discipline). And service systems may just as well be modeled by networks of queues. We refer the reader to e.g. (Gross et al. 2011) for an introduction to these concepts.

Let us denote the arrival time of the ith client as $a_i \in \mathbb{R}^+$. After arriving to the system, the client waits to be served. We denote this waiting as $w_i \in \mathbb{R}^+$. Once the service is completed the client leaves the system at departure time $d_i \in \mathbb{R}^+$. The sequence of departure times also defines a point process, to which we refer in the following as *departure process*. The service time $s_i \in \mathbb{R}^+$ is defined as the amount of time the ith client spends being served after the waiting period, that is $s_i = d_i - a_i - w_i$. Finally, the response time $r_i \in \mathbb{R}^+$ corresponds to the total time the customer requires to be processed, including the waiting time i.e. $r_i = d_i - a_i$. In this work we provide a simple methodology to infer service time distributions for the $G/G/\infty$ queuing problem[2].

Notation. In what follows we shall use capital letters for random variables and lowercase letters for their values. Likewise we denote probability distributions with capital letters, e.g. $P(S)$, and densities with lowercase letters, e.g. $p(s)$.

Related Work

Queuing Theory has a long history of both exact and approximated results, and these apply to single queues as well as to networks of them (Williams 2016). Often these studies focus on Markovian queues, i.e. queues for which both inter-arrival and service times are exponentially distributed. In such cases the queue-length process (that is, the number of clients in service process) is given by a continuous-time Markov chain, which makes it an object suitable for theoretical analysis (Bertsimas 1990; Mandelbaum, Massey, and Reiman 1998). For example, recent efforts investigate infinite-server queues whose arrivals follow a (Markovian) Hawkes process, and whose service time distributions are either phase-type or deterministic (Daw and Pender 2018). Their results include exact moments and the moment generating function of the queue-length process. Similarly, infinite-server queues whose arrival process is driven by a Cox process have also been considered (Boxma, Kella, and Mandjes 2019).

In contrast to queue-length processes, the departure process is *not* in general a discrete-time Markov chain, even for Markovian queues. This makes sense simply because keeping a client longer in service, or finishing serving her sooner, can cause slow reallocation of resources which, in turn, may affect the departure times of clients who have arrived much later. Despite this fact, Sutton and Jordan (2011) were able to infer service time distributions using Markov Chain Monte Carlo (MCMC) methods, defined via deterministic transformations between data and independent service times with different service disciplines, to model queuing networks with missing data. In fact, they argued that such long-range non-Markovian effects within the departure process

[1] By non-parametric we mean that we do not impose any parametric form (as e.g. Gamma or exponential) on the distributions we model. Instead we learn directly how to generate data following the distribution of interest.

[2] Due to the non-parametric nature of our solutions, the rich distributional forms we obtain allow for a solution of the $G/G/k$ problem, where the number of available servers "k" is unknown. Under this interpretation of the data, one should reinterpret our results as delivering the response time of the system.

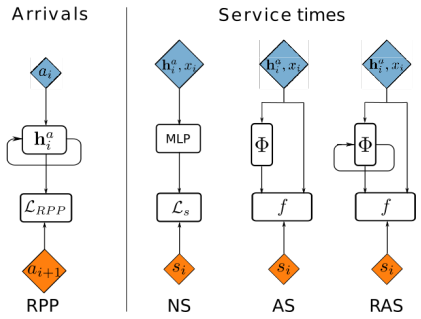


Figure 1: Left: the arrival times a_i are modeled with the RPP model with hidden state $\mathbf{h}_i^a$. Right: The service time models take $\mathbf{h}_i^a$ and covariates $\mathbf{x}_i$ as input. Here Φ labels the generator and f labels the critic.

take place only for "large" departure times, which were unlikely to occur. Later, Wang, Casale, and Sutton (2016) used Gibbs sampling techniques to model closed queueing networks (for which the total number of customers in the network is constant) that admit product-form equilibrium distributions; and Perez, Hodge, and Kypraios (2018) proposed a slice sampling technique for queueing networks which employs auxiliary variables, drawing some intuition from recent MCMC contributions (Rao and Teh 2013). These approaches, however, focus on specific type of systems and, due to the long mixing times and strong correlation properties across variables within their MCMC sampling schemes, suffer from scalability problems (Perez and Casale 2018). In fact, the large datasets we consider (of the order of $\gtrsim 10^6$ datapoints) encompass different time scales and may worsen the non-Markovian effects inherent in the departure process. Nonparametric alternatives to sampling methods are (Goldenshluger and Koops 2019; Senderovich et al. 2019).

From the neural network community, early approaches use neural networks as meta-models for queuing networks (Chambers and Mount-Campbell 2002), while more recent ones encode the average dynamics of closed queuing networks into RNNs (Garbi, Incerto, and Tribastone 2020). Also recently, and most related to our work, Chapfuwa et al. (2018) modeled time-to-event distributions using covariate information via Generative Adversarial Networks (GANs). We build on top of this work to capture the dynamic character of the client arrival process, and the independent dynamics of the server system.

Deep Generative Service Times

Given a series of N arrivals and associated departure times $\{a_i, d_i\}_{i=1}^N$, each represented in continuous time $\mathbb{R}^+$, we consider the $G/G/\infty$ queuing system for which waiting times w_i are zero and service times are given by $s_i = d_i - a_i$. Let each arrival have a set of covariates $\{\mathbf{x}_i \in \mathbb{R}^c\}_{i=1}^N$ associated to it (as, for example, the pick-up and drop-off locations for taxi rides on which the ride's duration depends strongly). We assume our service times are sampled from an unknown distribution $P_{\mathcal{D}}(S|\mathbf{x}_i, \mathcal{H}_i)$, conditioned on the covariates and the history of arrivals $\mathcal{H}_i \equiv \{a_1, ..., a_i\}$. The task is to learn a generative model with *implicit* probability distribution P_θ, which approximates $P_{\mathcal{D}}$. In a nutshell, our approach consists in modelling the client's arrival and service times separately:

Arrival Times Modelling – We first model the sequence of arrival times $\{a_i\}_{i=1}^N$ using the Recurrent Point Process (RPP) model of Du et al. (2016), which is both scalable and robust to model misspecification. The RPP Section below outlines the approach.

Service Times Modelling – We then use the hidden states of the *trained* RPP model encoding $\mathcal{H}_i$, together with the covariates $\{\mathbf{x}_i\}$, as input to our service time model: the Recurrent Adversarial Service Times (RAS). The RAS model deterministically maps these inputs together with random vectors from some latent space to the space of observations, and is parametrized by a second RNN, which is expected to capture the independent dynamic character of the server system. Details are presented in the RAS Section below

As such, the Deep Generative Service Times is a hierarchical model which treats service systems as a single $G/G/\infty$ queue, while making no assumption about its service discipline. Fig. 1 provides a basic outline of our proposed models.

One can briefly motivate our modelling strategy by examining the relation between arrival and service times. On one hand, and in accordance with queuing theory, the client arrival process is independent of the service system states, thus justifying our initial RPP arrival modelling. On the other hand, experience suggests that the service time distribution of a given system depends on the (dynamic) rate at which clients arrive. Thus, our service time models should be able to exploit the dynamic information encoded in the customer arrival process. Furthermore, many server systems are dynamic and time-dependent in nature: service times of new clients depend on how fast previous clients were served (remember the non-Markovian character of the departure process discussed above); service times may be heavily influenced by previous disrupting events, etc. Our construction should be capable of incorporating such a dynamic character. Hence the recurrent component of RAS.

Recurrent Point Process (RPP)

Within queuing theory the customer arrivals define a one-dimensional Point Process. In what follows we define $f^*(t)$ as the likelihood of a Point Process induced by an intensity function $\lambda^*(t)$. We define λ^* through a RNN, following the procedure stated in (Du et al. 2016; Mei and Eisner 2017). Let us consider a point process with compact support $\mathcal{S} \subset \mathbb{R}$. Formally, the likelihood is written as an inhomogeneous Poisson process between arrivals, conditioned on the history of arrivals $\mathcal{H}_i$[3] (Daley and Vere-Jones 2007). For one-dimensional processes the conditional likelihood that the next arrival happens at time t reads

$$f^*(t) = \lambda^*(t) \exp\left\{ \int_{a_i}^{t} \lambda^*\left(t'\right) dt' \right\}, \qquad (1)$$

where the intensity function λ^* is a (locally) integrable function. The functional dependence of the intensity function is

[3] a.k.a. filtration

given by a RNN with hidden state $\mathbf{h}_i^a \in \mathbb{R}^h$, where an exponential function guarantees that the intensity is *non-negative*

$$\lambda^*(t) = \exp\left\{\mathbf{v}^t \cdot \mathbf{h}_i^a + w^t (t - a_i) + b^t\right\}. \quad (2)$$

Here the vector $\mathbf{v}^t \in \mathbb{R}^h$ and the scalars w^t and b^t are trainable variables. The update equation for the hidden variables of the RNN can be written as a general nonlinear function

$$\mathbf{h}_i^a = g_\theta(a_i, \mathbf{h}_{i-1}^a), \quad (3)$$

where θ denotes the network's parameters.

RPP Loss – Inserting Eq. (2) into (1) and integrating over time immediately yields the likelihood f^* as a function of $\mathbf{h}_j^a$. We learn the model parameters by maximizing the joint model log-likelihood

$$\mathcal{L}_{\text{RPP}} = \sum_{i=1}^{N} \log f^*(\delta_{i+1} | \mathbf{h}_i^a), \quad (4)$$

where $\delta_{i+1} = a_{i+1} - a_i$ denotes the inter-arrival time and N is the total number of observed arrivals.

Recurrent Adversarial Service Times (RAS)

In order to model general service time distributions, and avoid the need to specify any service discipline, we consider a generative model defined by the following two-step process: (i) sample a random vector $\mathbf{z} \in \mathbb{R}^D$ from a known and easy-to-sample distribution $P(Z)$ (e.g. a Normal distribution); (ii) pass $\mathbf{z}$, together with the RPP representation $\mathbf{h}_i^a$ and covariate $\mathbf{x}_i$ of the ith arrival, through a parametric function $\Phi_\theta : \mathbb{R}^D \times \mathbb{R}^h \times \mathbb{R}^c \to \mathbb{R}^+$, with parameter set θ, that generates service time samples following a distribution P_θ. We will move this distribution "close" to the empirical service time distribution $P_\mathcal{D}$ by varying θ to minimize some reasonable distance between them.

The Service-Time Generator Φ_θ – For every arrival time a_i, we first sample a normally distributed random variable $\mathbf{z}_i \sim \mathcal{N}(0, 1)$ and define the (stochastic) hidden representation $\mathbf{u}_i \in \mathbb{R}^D$ thus

$$\mathbf{u}_i = \text{ReLU}\left(\mathbf{W}_a^u \mathbf{h}_i^a + \mathbf{W}_x^u \mathbf{x}_i + \mathbf{W}_z^u \mathbf{z}_i + \mathbf{b}^u\right), \quad (5)$$

where $\mathbf{W}_a^u$, $\mathbf{W}_x^u$, $\mathbf{W}_z^u$ and $\mathbf{b}^u \subset \theta$ are trainable parameters. Note that this representation encodes information about both the covariate $\mathbf{x}_i$ and the arrival history $\mathcal{H}_i$ through $\mathbf{h}_i^a$, which is defined in Eq. (3).

Second, we model the system's *dynamic response* to newly arrived customers with a RNN whose hidden state $\mathbf{h}_i^\Phi \in \mathbb{R}^D$ is defined through the update equation

$$\mathbf{h}_i^\Phi = g_\theta(\mathbf{u}_i, \mathbf{h}_{i-1}^\Phi). \quad (6)$$

Finally, we compute service times via

$$s_i = \exp\left(\mathbf{W}_h^s \mathbf{h}_i^\Phi + \mathbf{W}_z^s \mathbf{z}_i + \mathbf{b}^s\right), \quad (7)$$

where $\mathbf{z}_i \sim \mathcal{N}(0, 1)$ and $\mathbf{W}_h^s$, $\mathbf{W}_z^s$ and $\mathbf{b}^s \subset \theta$ are trainable parameters, and the exponential is chosen to restrict the samples to $\mathbb{R}^+$. The composition of the functions in Eq. (5), (6) and (7) defines our generator $\Phi_\theta = \Phi_\theta(\mathbf{z}, \mathbf{h}_i^a, \mathbf{x}_i)$. Note that we introduced an additional noise source in Eq. (7) to increase the variance of the samples $\{s_i\}$ (Chapfuwa et al. 2018).

This generative process implicitly defines a conditional probability distribution of the form $P_\theta(S|\mathbf{x}_i, \mathcal{H}_i)$, which depends on both covariates and the stochastic arrival process. By implicit we mean here that, while we can sample data from $P_\theta(S|\mathbf{x}_i, \mathcal{H}_i)$ through the generative process, its likelihood is intractable (since Φ_θ is not invertible). To train implicit generative models of this type, one usually resorts to minimising a distance (or divergence) between P_θ and the empirical distribution $P_\mathcal{D}$, that can be computed through their samples. The Wasserstein-1 distance and the f-divergences are examples of these, and the Adversarial approaches in (Arjovsky, Chintala, and Bottou 2017) and (Goodfellow et al. 2014; Nowozin, Cseke, and Tomioka 2016), respectively, minimize them. Below we introduce the Wasserstein-1 distance and follow the Wasserstein GAN (WGAN) methodology to train our model.

Wasserstein Loss and Training – The Earth-Mover or Wasserstein-1 distance between two distributions $P_\mathcal{D}$ and P_θ is defined as

$$W(P_\mathcal{D}, P_\theta) = \inf_{\gamma \in \Pi(P_\mathcal{D}, P_\theta)} \mathbb{E}_{(x,y)\sim\gamma}\left[||x - y||\right], \quad (8)$$

where $\Pi(P_\mathcal{D}, P_\theta)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are, respectively, $P_\mathcal{D}$ and P_θ. Since the infimum above is, in general, intractable, we turn to the Kantorovich-Rubinstein duality (Villani 2009) which tells us that

$$W(P_\mathcal{D}, P_\theta) = \sup_{f \in \mathfrak{L}_1} \mathbb{E}_{s\sim P_\mathcal{D}}\left[f(s)\right] - \mathbb{E}_{s\sim P_\theta}\left[f(s)\right], \quad (9)$$

where the supremum is over all 1-Lipschitz functions, and is still (in general) intractable. The trick is then to restrict the search space to that of a parametrized family of functions f_φ, the *critic* function, which one models with neural networks and learns under the 1-Lipschitz contraint. Now, our goal is to learn distributions not only conditioned on a set of covariates, but also on the stochastic arrival process. This is why we cannot directly use, in a straightforward way, the well known optimal solution of Eq. (8) in one dimension (Villani 2003; Santambrogio 2015). We instead define our critic as function of both $\mathbf{h}_i^a$ and $\mathbf{x}_i$, i.e. $f_\varphi = f_\varphi(s, \mathbf{h}_i^a, \mathbf{x}_i)$, and effectively learn one distance for each conditional. Putting all this together, we write our RAS loss function as

$$\begin{aligned}\mathcal{L}(\theta) = \max_\varphi \mathbb{E}_{(a_i,\mathbf{x}_i)\sim P_\mathcal{D}} \Big\{ &\mathbb{E}_{s_i\sim P_\mathcal{D}}\left[f_\varphi(s_i, \mathbf{h}_i^a, \mathbf{x}_i)\right] \\ &- \mathbb{E}_{s_i\sim P_\theta}\left[f_\varphi(s_i, \mathbf{h}_i^a, \mathbf{x}_i)\right] \\ &- \mathbb{E}_{s_i\sim P^*}\left[\left(\max\left\{0, |\nabla_{s_i} f_\varphi(s_i, \mathbf{h}_i^a, \mathbf{x}_i)| - 1\right\}\right)^2\right]\Big\}, \end{aligned} \quad (10)$$

with $\mathbf{h}_i^a = \mathbf{h}_i^a(a_i)$ defined in Eq. (3) and P^* defined by sampling uniformly along the straight lines connecting samples from $P_\mathcal{D}$ and P_θ. Note that the last term in Eq. (10) approximately imposes the 1-Lipschitz contraint by forcing the gradient norm of f_φ to be at most 1 along the straight lines sampled from P^* (Gulrajani et al. 2017; Petzka, Fischer, and Lukovnikov 2018). Therefore, minimizing $\mathcal{L}(\theta)$ above with respect to θ under an optimal critic function minimizes the Wasserstein-1 distance between $P_\mathcal{D}$ and P_θ – this defines

Algorithm 1: Recurrent Adversarial Service Time

Requires: Dataset $\mathcal{D} = \{(a_i, s_i, \mathbf{x}_i)\}_{i=1}^N$, Critic Iterations Number n_c and the penalty weight λ.
while θ not converged **do**
  for $i = 1, ..., N$ **do**
    Draw $\{(a_i, s_i, \mathbf{x}_i)\} \sim P_{\mathcal{D}}$
    for $k = 1, ..., n_c$ **do**
      Draw $\mathbf{z} \sim P(Z)$ and $\delta \sim \text{Uniform}(0, 1)$,
      $\mathbf{h}_i^a \leftarrow g_\rho(a_i, \mathbf{h}_{i-1}^a)$,
      $\tilde{s}_i \leftarrow \Phi_\theta(\mathbf{z}, \mathbf{h}_i^a, \mathbf{x}_i)$,
      $\hat{s}_i \leftarrow \delta s_i + (1 - \delta)\tilde{s}_i$,
      $\mathcal{L}_{\theta,\varphi} \leftarrow f_\varphi(\tilde{s}_i, \mathbf{h}_i^a, \mathbf{x}_i) - f_\varphi(s_i, \mathbf{h}_i^a, \mathbf{x}_i)$
      $+\lambda \left(\max\left\{0, |\nabla_{\hat{s}_i} f_\varphi(\hat{s}_i, \mathbf{h}_i^a, \mathbf{x}_i)| - 1\right\}\right)^2$,
      $\varphi \leftarrow \text{Adam}(\nabla_\varphi \mathcal{L}_{\theta,\varphi})$
    end for
    Draw $\mathbf{z} \sim P(Z)$
    $\theta \leftarrow \text{Adam}(\nabla_\theta(-f_\varphi(\Phi_\theta(\mathbf{z}, \mathbf{h}_i^a, \mathbf{x}_i))))$
  end for
end while
return Φ_θ

the adversarial game. Alg 1 summarizes the RAS model algorithm. Please note that the parameters of the RPP model (labeled ρ in line 7 of Alg 1) are fixed and optimal.

Relations to Formal Analysis of Queuing Systems

Being non-parametric, our constructions do not allow a formal theoretical treatment in the spirit of traditional queuing system results (Williams 2016). However, one could employ neural network analysis to obtain estimates of service times, which could eventually be related to formal results from the queuing community. To illustrate, assume the simplified service time model $\tilde{s}_i := \theta^k \circ \sigma^k \dots \sigma^1 \circ \theta^1(\mathbf{h}_i)$, where $\mathbf{h}_i := \text{concat}(\mathbf{h}_i^a, \mathbf{z})$, with $\mathbf{h}_i^a$ defined in Eq. (3) and $\mathbf{z} \in \mathbb{R}^D$ sampled from some simple distribution $P(Z)$ as above; $\sigma = \text{ReLU}$ and θ^j labels the weights of the jth layer.

Lemma 1. *Suppose that the simplified model above fits the data within a mean average error of ε. Then the average service time is bounded above as follows*

$$\langle \tilde{s}_i \rangle \leq M^k \limsup_i \|\mathbf{h}_i^a\| + \frac{M^{k+1} - 1}{M - 1} + \varepsilon. \tag{11}$$

Here M is a positive constant bounding the operators norm $\|\theta^j\|$ for all j. This straightforward estimate is proved and related to the average queue length in the Supplementary Material.

Baseline Models

Given that there are not many scalable nonlinear models for service time estimation, we now introduce a set of service time models to be used as baselines in the Experiment Section below. These baselines will not only provide a fair comparison, but also help us understand the role of some of the properties of our Deep Generative Service Times model.

Adversarial Baselines – First of all, we compare against the Adversarial Time-to-Event (ATE) model of Chapfuwa et al. (2018) to test the importance of conditioning on the arrival history, as dictated by Queueing Theory. We also compare against RATE, a recurrent version of the ATE model defined as in Eqs. (5-7), but without $\mathbf{h}_i^a$ as input.

We would also like to test how relevant is the explicit time dependence of RAS. To do so we introduce the Adversarial Service (AS) model, defined by composing only the functions in Eqs. (5) and (7), but with $\mathbf{h}_i^\Phi$ replaced with $\mathbf{u}_i$ in the latter.

Maximum-Likelihood-Based Baselines – We introduce parametric (i.e. non-adversarial) service time models: the Neural Service model (NS)

$$s_i \sim P_\theta(S|\mathbf{h}_i^a, \mathbf{x}_i), \tag{12}$$

where $\mathbf{h}_i^a$ is the hidden state of the (trained) RPP model, $\mathbf{x}_i$ are covariates and P_θ is chosen to be one of the following five distributions: *Gamma* (NS-G), *Exponential* (NS-E), *Pareto* (NS-P), *Chi-square* (NS-C) or *Log-normal* (NS-L), whose parameter set $\mathcal{P}_\theta$ are defined via Multi-Layer Perceptrons, with θ labelling the trainable variables. These output distributions are common solutions to stationary service time distributions in theoretical models (Asmussen 2008). The NS models are thus an attempt to have a scalable representation of these theoretical approaches. Finally, let us also define the Neural Time-to-Event (NTE) models, which correspond to a non-adversarial version of ATE. We define them with an equation similar to Eq. (12), but without conditioning on $\mathbf{h}_i^a$. We train all these models via maximum log-likelihood.

Experimental Setting

In this section we introduce our experimental framework. We provide synthetic datasets with well-established models for both arrivals and service processes, as well as empirical datasets. modelling these datasets demonstrates the ability of our approach to handle diverse application areas in a flexible and scalable manner.

Synthetic Datasets – In order to provide a controlled environment to test the behavior of our methodology we introduce the following datasets for different arrival and service processes:

(1) *Arrival processes*: we consider two different arrival processes, namely (i) the Hawkes Process (H), which is a model for self-exciting phenomena where user arrivals increase the probability of other users to arrive (Hawkes and Oakes 1974), and which is defined via the conditional intensity function

$$\lambda_H(t) = \lambda^* + e^{-\beta t}(\lambda_0 - \lambda^*) + \sum_{a_i : t > a_i} \alpha\, \mu(t - a_i), \tag{13}$$

where λ_0 and λ^* are the initial and baseline values of the process intensity for exogenous (arrival) events, α, β are its jump and decay parameter, and the memory kernel $\mu(t) = e^{-\beta t}$ yields the intensity given by past arrivals a_i; (ii) the Non-linear Hawkes Process (NH), which is an extension of the Hawkes process that allows for inhibitory behavior through a nonlinear function over the history of arrivals (Zhu 2013).

	NH-PT		NH-PS		H-PS		Github		NY		Stackoverflow	
mean ($\bar{s}$)	0.052		0.0004		2.125e-5		0.0113		0.0068		0.0193	
	error	KS	error	KS	error	KS	error	KS	error	KS	error	KS
NTE	0.410	0.289	0.0450	0.410	2.44e-2	0.766	0.071	0.388	0.062	0.321	0.380	0.502
NS	0.209	0.154	0.0006	0.082	**2.18e-5**	0.401	0.096	0.341	0.025	0.154	0.378	0.466
ATE	0.219	0.193	0.0371	0.870	3.96e-3	0.199	0.217	0.517	0.078	0.126	0.382	0.233
AS	0.215	0.113	0.0016	0.448	1.24e-4	0.121	**0.071**	0.039	0.006	0.094	0.383	**0.226**
RATE	0.218	0.124	0.0031	0.062	1.37e-4	0.136	0.112	0.240	0.098	0.165	0.388	0.492
RAS	**0.207**	**0.094**	**0.0005**	**0.042**	1.09e-4	**0.110**	0.072	**0.034**	**0.005**	**0.030**	**0.369**	0.281

Table 1: Models evaluation on all data sets (in arb. units). KS - Kolmogorov–Smirnov test.

(2) *Service models*: we introduce two service system disciplines, namely (i) the Phase-Type Distribution (PT) (Daw and Pender 2018), in which one decomposes the service as a series of exponential service steps, and which is defined with the time taken between the initial and absorbing state[4] in a Continuous Time Markov Chain (CTMC); (ii) the Processor Sharing Distribution (PS) (Kleinrock 1976), a queuing model in which the system handles an infinite amount of clients simultaneously, but must reallocate resources with each new client arrival or departure. One can think that each client in the system instantaneously receives $1/Q(t)$ of service power, at any time, where $Q(t)$ is the queue length (clients still on service) (Sutton and Jordan 2011).

The combinations of the two arrival and the two service models yields four different synthetic datasets, which we label H-PT, H-PS, NH-PT and NH-PS. We simulated about 5×10^6 datapoints for the Hawkes models, and 5×10^5 datapoints for its nonlinear extensions. For space reasons we present our results on the H-PT model in the Sup. Material.

Empirical Datasets – We gathered datasets from a variety of internet services, as to provide an in-depth analysis of the temporal patterns of users in different domains.

(1) *Stackoverflow*: a question-answering platform for programmers. We define the customer arrivals as the points in time when questions are posted by the users of the web page, and the service time as the elapsed time between a question and its subsequent accepted answer. As covariates we take the first five tags of each question. This view establishes the ensemble of users which provide answers as the service system. We analyse a total of 2×10^7 questions.

(2) *Github*: The version control repository and Internet hosting service. We define the creation of an issue in a given repository as the customer arrivals. The departure times are the moments the given issues are closed. We chose no covariates for this dataset. The set of users associated with a given repository can be then thought of as the service system. We analyse the top (ranked by the number of issues) 500 repositories in the platform in 2015, for a total of 1.5×10^6 different issues.

(3) *New York City Taxi Dataset* (NY): The dataset contains data of individual taxi trips in New York city. Customers arrivals are defined as the starting time of the trip and the departure time as the time when the trip ends. As covariates we choose the starting and ending points of the trips w.r.t the NY taxi zones[5]. Here the service system is given by both the taxi providing the service and the transportation network of roads, streets and highways pertaining to the city of New York. We analyse about 1.1×10^7 trips.

Reproducibility and Training Details

We split all datasets into training and test sets, the latter being defined as $\sim 5\%$ of the complete dataset[6]. Details of the neural networks architectures, learning parameters and any other hyperparameters as required in the model specification can be found in the Supplementary Material.

Results

In order to quantify the performance of our model, we first focus on two aspects, namely its prediction capabilities and its ability to uncover rich distributional forms. Thus we consider the prediction error defined as $1/N \sum_i |s_i - \langle \tilde{s_i} \rangle|$, where N is the size of the test dataset, s_i denotes the empirical value and $\langle \tilde{s_i} \rangle$ denotes the average prediction obtained via Monte Carlo sampling. To quantify the descriptiveness of the obtained distributional form, we then calculate the Kolmogorov-Smirnov (KS) statistics between the empirical and generated distributions, and also provide Q-Q plots against the empirical distributions. Later we also test how the model compares with exact theoretical estimates of a related observable: the queue length (i.e. the number of clients on service).

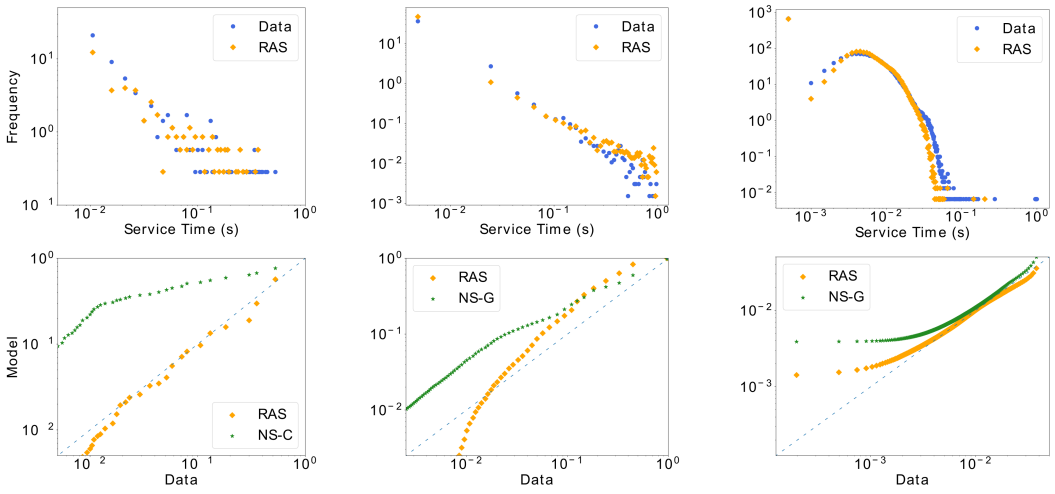
Comparison with Baselines – The comparison of the different models based on the predictive error and the KS statistics for both synthetic and empirical datasets is shown in Table 1. Here we present only the best[7] NTE and NS models, out of their five different possibilities (see Baseline Section above). The complete set of results for all ten models can be found in the Suplementary Material. These results show that the RAS model outperforms most baselines w.r.t. both measures, demonstrating a clear advantage of our solutions. In particular, we note that: (i) all service time models are found to describe better the empirical distributions, as compared to the time-to-event models, which ignore the arrival information and hence the hierarchical nature of queuing systems. Indeed, conditioning the generative

[4] a type of first passage time

[5] https://www1.nyc.gov/site/tlc/about/data-and-research.page

[6] We can provide the reader with the datasets used in this work upon request.

[7] with respect to the KS test.



(a) Github (b) Stackoverflow (c) NY Taxi

Figure 2: (Top) Comparison between empirical service time distributions (test set) and data generated by RAS. (Bottom) Q-Q plots against empirical distributions for both best NS model and RAS model.

models on the customer RPP process significantly improves their performance in either metric (see e.g. NS vs NTE; AS vs ATE; RAS vs RATE). These models successfully use the vector representation $\mathbf{h}_i^a$ encoding the arrival dynamics; (ii) RAS outperforms the AS model in almost all datasets. This corroborates our assumption that including a model for the service system's dynamic response helps in describing the system's behavior.

The strength of RAS is even more apparent in their distributional forms. Fig. 2 shows histograms of data generated with RAS against the empirical data distribution, as well as Q-Q plots of both the best NS model and RAS model versus the data. Similar plots for the synthetic datasets can be found in the Supplementary Material. Note that the timescales in these datasets encompass up to four orders of magnitude. The RAS model provides a much better fit to the data than the NS models (see Q-Q plots). In particular, RAS uniquely captures both long and short term behavior. The latter is apparent from the upper left corner of the histogram plots. The NS model, in contrast, only provides correct long tail behavior as it is constrained by the distributional forms of the outputs (see also the Figures in the Sup. Mat.).

Comparison with Theory – Let us now consider again the Hawkes/Phase-Type/∞ queue for which the time-dependent average queue length has been shown to be

$$\langle Q(t)\rangle = \lambda_\infty(-\mathbf{S}^\top)^{-1}(\mathbb{1} - e^{\mathbf{S}^\top t})\boldsymbol{\omega} - (\lambda_0 - \lambda_\infty)\times (\mathbf{S}^\top + (\beta-\alpha)\mathbb{1})^{-1}(\mathbb{1}\, e^{-(\beta-\alpha)t} - e^{\mathbf{S}^\top t})\,\boldsymbol{\omega}, \quad (14)$$

where λ_0, α and β are parameters of the Hawkes process (see Eq. (13)) and $\lambda_\infty = \frac{\beta\lambda^*}{\beta-\alpha}$ is the the steady-state value of its intensity. The matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ is the transient state subgenerator for a Phase-Type distribution with n phases, $\mathbb{1}$ is the $n \times n$ identity matrix and $\boldsymbol{\omega} \in \mathbb{R}^n$ is the initial arrival distribution over the phases (Daw and Pender 2018). We simulated a Hawkes/Phase-Type/∞ queue and trained RAS on the resulting dataset (see Sup. Mat. for details on the parameters we used). We can then estimate $\langle Q(t)\rangle$ using RAS by sampling one service time for each arrival time in our test set, counting and keeping track of the number of clients in service and averaging over the test set. Figure 3 shows our results as compared with the exact moment, Eq. (14), and the simulated test set. Our methodology is thus able to capture the dynamic queue length of the service system, thereby inferring information about its workload.

Conclusion

In this work we presented a novel (deep) non-parametric solution for service-time distribution learning in Queuing Systems with general arrival distributions. Our methodology outperformed all baselines and reproduced complex service time distributions, inferring both multi-modal and long-tail features. Future lines of work include incorporating richer representations encoding the arrival process. In mobility systems, for example, the geographic information and user interactions can be encoded through hidden relation networks.

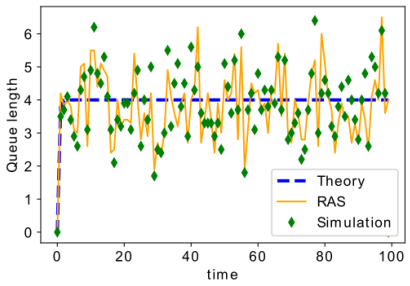


Figure 3: Time-dependent averaged queue length $\langle Q(t)\rangle$ for a Hawkes/PhaseType/∞ (H-PT) queue.

Acknowledgments

The authors of this work were supported by the Fraunhofer Research Center for Machine Learning (RCML) and by the Competence Center for Machine Learning Rhine Ruhr (ML2R), which is funded by the Federal Ministry of Education and Research of Germany (grant no. 01—S18038A). Part of the work was also funded by the BIFOLD-Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref 01IS18037A). We gratefully acknowledge this support.

References

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein Generative Adversarial Networks. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 214–223. International Convention Centre, Sydney, Australia: PMLR.

Asmussen, S. 2008. *Applied probability and queues*, volume 51. Springer Science & Business Media.

Bertsimas, D. 1990. An Analytic Approach to a General Class of G/G/s Queueing Systems. *Operations Research* 38(1): 139–155.

Boxma, O.; Kella, O.; and Mandjes, M. 2019. Infinite-server systems with Coxian arrivals. *Queueing Systems* 92(3): 233–255.

Chambers, M.; and Mount-Campbell, C. 2002. Process optimization via neural network metamodeling. *International Journal of Production Economics* 79(2): 93 – 100.

Chapfuwa, P.; Tao, C.; Li, C.; Page, C.; Goldstein, B.; Carin, L.; and Henao, R. 2018. Adversarial Time-to-Event Modeling. In *ICML*.

Daley, D. J.; and Vere-Jones, D. 2007. *An introduction to the theory of point processes: volume II: general theory and structure*. Springer Science & Business Media.

Daw, A.; and Pender, J. 2018. Queues driven by Hawkes processes. *Stochastic Systems* 8(3): 192–229.

Du, N.; Dai, H.; Trivedi, R.; Upadhyay, U.; Gomez-Rodriguez, M.; and Song, L. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1555–1564. ACM.

Erlang, A. K. 1909. The theory of probabilities and telephone conversations. *Nyt. Tidsskr. Mat. Ser. B* 20: 33–39.

Garbi, G.; Incerto, E.; and Tribastone, M. 2020. Learning Queuing Networks by Recurrent Neural Networks. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, 56–66. Association for Computing Machinery. ISBN 9781450369916.

Goldenshluger, A.; and Koops, D. T. 2019. Nonparametric Estimation of Service Time Characteristics in Infinite-Server Queues with Nonstationary Poisson Input. *Stochastic Systems* 9(3): 183–207.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*, 2672–2680. Curran Associates, Inc.

Gross, D.; Shortle, J.; Thompson, J.; and Harris, C. 2011. *Fundamentals of Queueing Theory*. Wiley Series in Probability and Statistics. Wiley.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved Training of Wasserstein GANs. In *NIPS*.

Hawkes, A. G.; and Oakes, D. 1974. A cluster process representation of a self-exciting process. *Journal of Applied Probability* 11(3): 493–503.

Jing, H.; and Smola, A. J. 2017. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 515–524. ACM.

Kendall, D. G. 1953. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *Ann. Math. Statist.* 24: 338–354.

Kleinrock, L. 1976. *Queueing systems, volume 2: Computer applications*, volume 66. wiley New York.

Lin, K.; Li, D.; He, X.; Zhang, Z.; and Sun, M.-T. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*, 3155–3165.

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

Mei, H.; and Eisner, J. M. 2017. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, 6754–6764.

Mirza, M.; and Osindero, S. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* .

Nowozin, S.; Cseke, B.; and Tomioka, R. 2016. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems 29*, 271–279.

Perez, I.; and Casale, G. 2018. Approximate Bayesian inference with queueing networks and coupled jump processes. *ArXiv* abs/1807.08673.

Perez, I.; Hodge, D.; and Kypraios, T. 2018. Auxiliary variables for Bayesian inference in multi-class queueing networks. *Statistics and Computing* 28(6): 1187–1200.

Petzka, H.; Fischer, A.; and Lukovnikov, D. 2018. On the regularization of Wasserstein GANs. In *International Conference on Learning Representations*.

Rao, V.; and Teh, Y. W. 2013. Fast MCMC Sampling for Markov Jump Processes and Extensions. *Journal of Machine Learning Research* 14(67): 3295–3320.

Santambrogio, F. 2015. *Optimal transport for applied mathematicians*. Birkhäuser Springer.

Senderovich, A.; Beck, J. C.; Gal, A.; and Weidlich, M. 2019. Congestion Graphs for Automated Time Predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 4854–4861.

Sutton, C.; and Jordan, M. I. 2011. Bayesian inference for queueing networks and modeling of internet services. *The Annals of Applied Statistics* 254–282.

Villani, C. 2003. *Topics in Optimal Transportation*. American Mathematical Society.

Villani, C. 2009. *Optimal Transport: Old and New*. Springer.

Wang, W.; Casale, G.; and Sutton, C. 2016. A bayesian approach to parameter inference in queueing networks. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 27(1): 1–26.

Williams, R. J. 2016. Stochastic Processing Networks. *Annual Review of Statistics and Its Application* 3(1): 323–345.

Xiao, S.; Farajtabar, M.; Ye, X.; Yan, J.; Song, L.; and Zha, H. 2017. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, 3247–3257.

Xiao, S.; Xu, H.; Yan, J.; Farajtabar, M.; Yang, X.; Song, L.; and Zha, H. 2018. Learning conditional generative models for temporal point processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Zhu, L. 2013. Central limit theorem for nonlinear Hawkes processes. *Journal of Applied Probability* 50(3): 760–771.