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

TOWARDS NEW SIMULATION MODELS FOR DL-BASED VIDEO STREAMING IN EDGE NETWORKS

Abdolreza Abhari[1]

[1]Distributed Systems & Multimedia Processing Laboratory (DSMP Lab),
Department of Computer Science, Toronto Metropolitan University, Toronto, ON, CANADA

ABSTRACT

To evaluate novel solutions for edge computing systems, suitable distribution models for simulation are essential. The extensive use of deep learning (DL) in video analytics has altered traffic patterns on edge and cloud servers, necessitating innovative models. Queuing models are used to simulate the performance and stability of edge-enabled systems, particularly video streaming applications. This paper demonstrates that traditional Markovian M/M/s and general distribution G/G/s queuing models must be revamped for accurate simulation. We examined these queuing models by characterizing the real data with discrete and continuous distributions for arrival rates to homogenous servers in AI-based video analytics edge systems. Based on achieved results, traditional methods for finding general distributions are inadequate, and an automation method for finding empirical distribution is needed. Therefore, we introduce a novel approach using a generative adversarial network (WGAN) to generate artificial data to automate the process of estimating empirical distribution for modeling these applications.

1 INTRODUCTION

One of the advantages of the 5G and 6G networks compared to their predecessors is the use of edge computing (Tyokighir et al. 2024). Edge computing shifts computer storage and processing to the network's edge, closest to users and devices and most critically, as close as possible to data sources. The new generation of networks, advancements in IoT, and the use of edge and fog devices in conjunction with central cloud computing have all advanced the development of edge-enabled computing. Deep learning and AI-based methods are increasingly employed in the edge computing applications proposed for smart city infrastructure, smart sensors, blockchain/intelligent caching, and deep leap learning video analytics.

In traditional queuing modeling, arrival rates are usually considered integer discrete values from probability functions such as Poisson distribution. In our previous work (Abhari et al. 2022), we measured the arrival rates of a real edge-network application with the unit of frame per second. We use the collected data in this work to show that general discrete and continuous distributions do not exactly fit the data. We obtained real numbers for frame arrival rate by data collection and workload characterization of an edge-based security camera application. The details of the security camera application used for data collection are discussed in (Abhari et al. 2022). The arrival rates of video frames for different application parts were measured during the video streaming, including object detection and object tracking, on the edge servers. In this work, the collected data is modelled by Poisson and exponential distributions as well as Gamma and Uniform distributions, representing M/M/s and general distributions of G/G/s queuing, respectively. For the Markovian queuing system, its common formulas are used, and for G/G/s queuing systems, the estimation of queuing delay is done using the mathematical formulas adapted from Medhi (2003) and (Whitt 1983). The rationale for using Uniform distribution in G/G/s instead of exponential for service time

is that we are dealing with video streaming applications that need specific deadlines that heterogeneous servers with different service times can not provide when processing the frames.

One of the features of current video streaming applications is the use of deep learning artifacts such as CNN and DNN for video analytics. These data-intensive video analytic applications can not be implemented only by edge computing; they are traditionally called edge killer applications. Therefore, for large applications on top of complex networks, including edge, fog, and cloud computing with different IoT devices or cellular networks with mobile devices, the Deep Learning (DL) parts are divided and distributed to edge/fog devices and the cloud. The problem is that mixing different application workloads, latency, and jitters in complex networks constantly changes traffic and arrival patterns. The scope of this work is finding accurate distribution models for simulating edge computing when network latency (or queuing delay) is small and can be calculated because of the proximity of IoT devices (such as security cameras) and edge servers.

There are limited workload characterization studies (because of the difficulty of collecting real data) for edge-enabled DL applications in the literature. Finding the distribution models is necessary for generating artificial data for the comprehensive simulation of these systems. In our data collection, we measured different processing times of different DL tasks done on edge devices for a security camera video streaming application in the units of frames/sec. Similar to the new generation of edge-based applications in our tested application, a dynamic scheduler is used called VADRM method that divides different parts of each video job and sends the frames of that job to the specific device through the edge/fog network (Abhari et al. 2022). We can model this system with one queue under the scheduler and multiple servers.

To make the problem simple, we assume the ideal situation where all the streams of frames are coming from the camera in a sequence with an interarrival times distribution model and placed on the queue, and the factor determining how much they should wait in the queue is whether the servers are busy or not. So, with this assumption, the different types of jobs and bulk allocation of all frames of one job to a server will be simplified, and the analytical model can be achieved by finding the distribution of arrival rate and interarrival times between each frame in a queuing system. To find the effectiveness of modeling, in this work, two M/M/s and G/G/s queue models (s represents the number of servers) are solved to find the queuing delay, which is the edge network latency. Then, the obtained results are compared with simulation results conducted with the iFogSim edge/fog simulator done by (Abhari et al. 2022), which are presented in Section 4.

2 RELATED WORK

Simulation is a well-known method when testing in a real system is expensive, and it is a proven method for evaluating complex networks. The most important part of the simulation is modeling. Although there are several state-of-the-art simulators for edge /fog and cloud computing, only a few of them use G/G/s queuing, which is examined in this work. This section first looks at the literature about queuing modeling and then discusses the current edge simulators. Queuing theory has been widely used to model and simulate the performance of edge/cloud networks and related video streaming-based systems (Pu et al. 2023), IoT task offloading (Fan et al. 2023), and Software-defined networking (SDN) systems (Amadeo et al. 2023). However, our recent research shows that queuing modeling has received little attention when the arrival rate and service time do not follow the Markovian process. Markovian queuing models are used for stability and network delay with interarrival and service times of Poisson and Exponential distributions (Giorno et al. 2018). The classical text indicates that the G/G/s queue has no easy solutions and should be solved by approximating M/M/s and then validating its results by simulation.

Here, edge/cloud simulators and the models they provide for edge computing applications are discussed. CloudSim (Calheiros et al. 2010 and CloudSim) is a holistic open-source software framework for modeling cloud computing environments and performance testing application services. It is the most widely used cloud simulator in the research community. However, it does not support the simulation and modeling of IoT and edge computing environments, nor its extensions CloudSim Plus (Silva Filho et al. 2017) or an open source CloudSim 3 (CloudSim and Cloud Plus).

iFogSim (Gupta et al. 2017) is a Java-based simulator that simulates IoT and fog computing environments and evaluates resource management techniques in terms of latency, network congestion, energy consumption, and cost. The physical topologies can also be built programmatically through Java APIs. To demonstrate the effectiveness of iFogSim for evaluating resource management techniques, the research paper (Jha et al. 2020) discussed two case studies: a latency-sensitive online game and intelligent surveillance through distributed camera networks. The results of this study demonstrated that iFogSim could enable simulations on the scale required in the context of IoT. However, iFogSim1 and iFogSim2 tools do not support any distribution modeling for arrival rates or workloads and use only the Edgeward algorithm for module allocation of fog devices. The same lab recently released Fogbus2 (Deng et al. 2021), a Python-based fog simulator that provides a mechanism for scheduling heterogeneous IoT applications and implements several scheduling policies that still lack queue modeling with predefined distributions.

A recent survey paper compares several edge simulators with their practical capabilities, such as FogNetSim++, EdgeCloudSim, YAFS, LEAF or EdgeSimPy. The comparison shows that some simulators are built for specific applications, and only a few continue to extend their features to solve general applications (Fahimullah et al. 2023). The novel modeling method proposed in Section 5 of this paper suggests using a neural network (called WGAN) to generate artificial data for DL-based video streaming applications and potentially for simulations of many other recent applications in 5G or 6G involvements.

3 ANALYSIS OF DISTRIBUTION MODELS

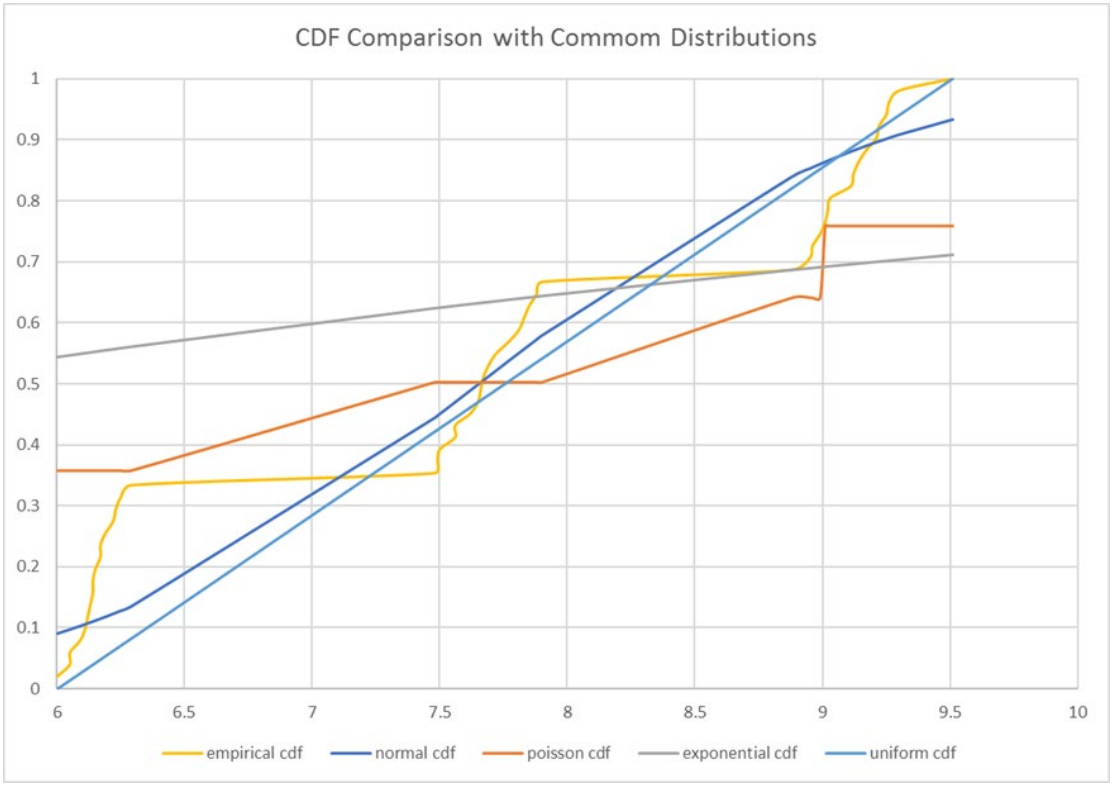
As mentioned above, we measured the speed of different parts of a video analytic application, which was distributed to edge servers for security camera IoT devices. In that measurement, the arrival rates of frames in different edge servers are collected in the unit of frames per second. We have compared many distributions with common discrete models used to examine their goodness of fit, which is shown below.

Regarding continuous distributions, Beta and Gamma distributions are better candidates for simulating the collected arrival rates of traffic from IoT devices to edge servers, which are measured in frames/sec. In the queuing models, knowing interarrival times determines whether it is the Markovian models. For example, if the interarrival time is exponential and the arrival rate is Poisson, then the model is M/G/s. If service time is also exponential, then the queuing model is M/M/s for *s* servers. In the case of collected data that is shown below, the arrival rate is not Poisson distribution and interarrival time and service times are unknown; thus, the curve fitting method was used for finding the empirical distribution model from the following video analytics data, which are measured in edge devices in the units of frames per second. Since in edge-based video streaming applications, different parts of video analytics may be sent to the edge devices, the collected data shown below is categorized:

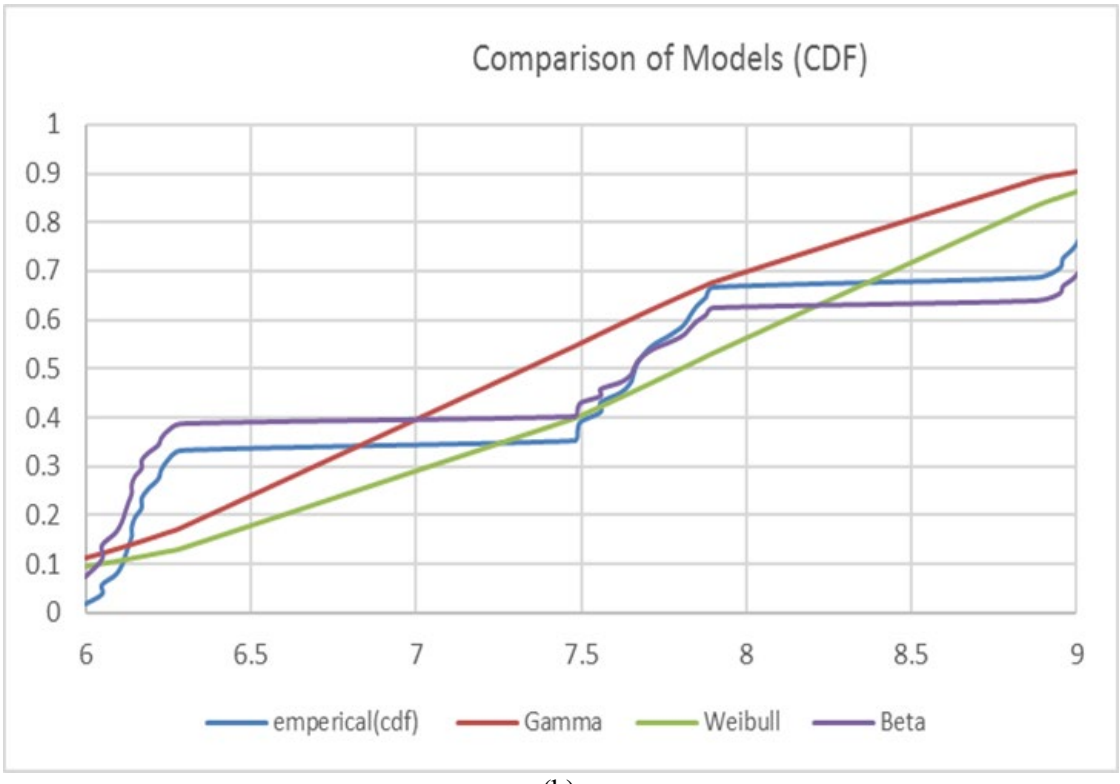
Motion Detection: 9.21, 8.88, 9.51, 9.26, 9.12, 9.01, 8.95, 9.11, 8.96, 9.03, 8.99, 9.02, 9.30, 9.22, 9.25, 9.14, 9. 17
Object Detection: 7.56, 7.50, 7.65, 7.90, 7.67, 7.66. 7.88, 7.77, 7.85, 7.62, 7.48, 7.69, 7.83, 7.56, 7.49, 7.72, 7.81
Object Tracking: 6.14, 6.0, 6.11, 6.17, 6.09, 6.19, 6.05, 6.13, 6.14, 6.12, 6.23, 6.15, 6.25, 6.29, 6.05, 6.17, 6.22

Analyzing the CDF plot provides insight into the fact that the empirical distribution of the dataset does not match the expected patterns of common distributions such as uniform, Poisson, or Exponential models (shown by Figure 1-a). The empirical distribution's shape suggests that different specific distributions, such as the Beta or Gamma distributions, can provide a better and more insightful presentation of the data's distribution (shown by Figure 1-b).

Another candidate is the family of normal distribution such as Weibull and Lognormal that, although not a good match, can show data trends. To complete the visual inspection, we used the Q-Q plots that visually examine the data's alignment with two distributions of Log-normal and Beta distributions in Figure 2. The theoretical line observed in the QQ plots of Figure 2 shows that deviations are most noticeable in



(a)



(b)

Figure 1: CDF comparison of (a) common and (b) candidates with empirical distribution.

the tails. The Beta distribution demonstrates the closest fit to the expected line, particularly in the middle quantiles, although it also shows some divergence at the extremes.

The Log-normal distribution, in comparison, exhibits more significant departures from the line, especially in the lower quantiles. This suggests that it can somewhat accurately model the central portion of the data.

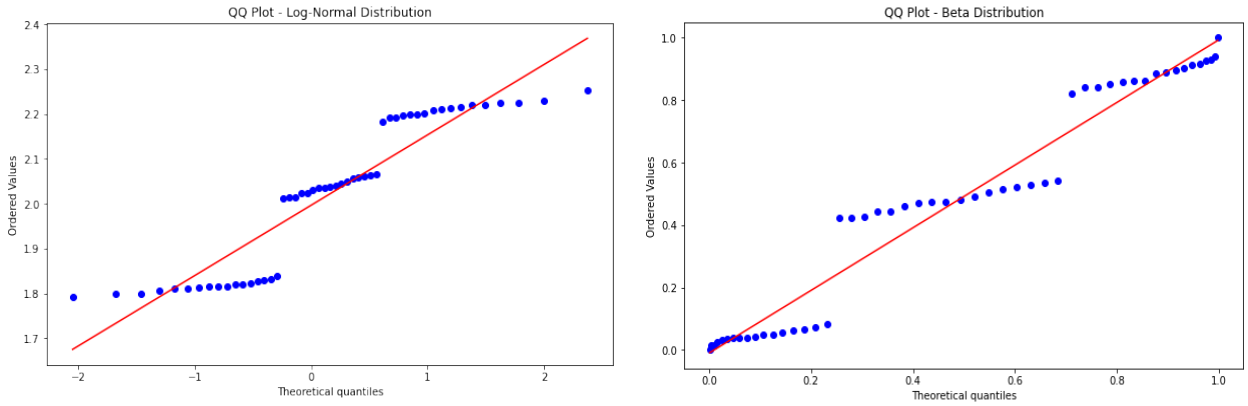


Figure 2: QQ plots for Log-normal distribution (left) and Beta distribution (right).

After visual observation, the Kolmogorov-Smirnov (K-S) test was used for the closest candidates with the specific p-values and K-S parameter to assess the strength of evidence for each distribution fit in Table 1: The results of the K-S test are illustrated in Table 1, and the hypothesis for statistical tests is explained below.

Table 1: Kolmogorov-Smirnov (K-S) test results for selected distributions.

Distribution	**K-S Statistics**	**P-Value**
Poisson	0.3576	2.4e-06
Exponential	0.4924	6.15e-12
Gamma	0.6939	9.00e-21
Beta	1.0	0
Normal	0.2007	0.028068

- **Null Hypothesis (H0)**: The distribution of video frame arrival rates follows the specified theoretical distribution (Poisson, Exponential, Gamma, or Uniform). This suggests that the observed data fits well with the specified distribution model, indicating no significant deviation from these theoretical distributions.
- **Alternative Hypothesis (H1)**: The distribution of video frame arrival rates does not follow the specified theoretical distribution (Poisson, Exponential, Gamma, or Uniform). This implies that the observed data significantly deviates from the specified distribution model, indicating that another distribution may better represent the data.

For each distribution model tested, the K-S test results will help us determine if we can accept the distribution as a suitable model for our data or if we need to consider alternative distribution models. If the p-value obtained from the K-S test is greater than the significance level (typically 0.05), in that case, we fail to reject the null hypothesis for that distribution, suggesting that the data does not significantly deviate from the specified distribution. Conversely, if the p-value is less than the significance level, we reject the null hypothesis, indicating that the data significantly deviates from the specified distribution model. Based on the p-values of the Kolmogorov-Smirnov (K-S) test (which are less than 0.05) shown in Table 1, we reject the null hypothesis, meaning none of the distributions fit the data.

4 PERFORMANCE VALIDATION OF QUEUING MODELS

Although we didn't find the best fit for the collected data, the easiest way to select the simulation models is to find the distributions for queuing models. For M/M/s model, its common distributions can be used but for G/G/s queue we need to select the distribution of arrival rate and service time. For arrival rate we considered both Beta and Gamma distributions as they are visually closer to the empirical distribution of collected data as shown in Figure 1. Beta distribution shows a better match to empirical data; however, we used Gamma distribution as it is more common in queuing modeling and is related to Poisson and Exponential distributions, so it can be used for arrival mode (i.e., both arrival rates and interarrival times). Gamma distribution has two parameters (a, b), a is shape and b is the scale parameter, sometimes shown by its inverse, which is called rate parameter. For showing λ arrival rate we used the mean value of the Gamma distribution with two parameters *a* and *b*, which is ab for the Gamma distribution when considering b as the scale parameter.

For service time, we use the Uniform distribution with the range (v_1, v_2) seconds for processing incoming frames from the IoT devices. So, we can calculate the mean service time as $\frac{v_2+v_1}{2}$ and $\mu = \frac{2}{v_2+v_1}$, where μ is the service rate and σ_s^2 is the variance of service time, which for Uniform distribution is:$\sigma_s^2 = \frac{(v_2-v_1)^2}{12}$. Therefore, considering identical *s* servers the stability of system can be shown as:

$$\rho = \frac{\lambda}{s\mu} = \frac{ab}{s\mu} = \frac{ab(v1+v2)}{2s} \quad (1)$$

There is no formula to calculate queuing delay in G/G/s queues. , the frame arrival mode acts similarly to Wq_{MMs} of M/M/s queue where Wq_{MMs} is a queuing delay for M/M/s . Wq_{GGs} queuing delay of G/G/s can be estimated by using the above constants and the following formula:

$$Wq_{GGs} = Wq_{MMs}\frac{C_a^2+C_s^2}{2} \quad (2)$$

Where, the C_a^2 constant for the arrival is calculated by the ratio of variance over the square of inter-arrival time and similarly the C_s^2 for service is calculated by dividing the variance of service time by the square of the mean of service time. All the calculations, including the estimation of Wq_{MMs} are done according to Medhi (2003) and Whitt (1983).

To validate the proposed model, the results used from the simulation of edge computing performed by the iFogSim simulator with the same data set discussed above and three edge servers with the same configuration as explained in (Abhari et al.2022). In that simulation, iFogSim was used for a security camera edge-computing application with the video input sizes mentioned in the previous section and its default configuration for three edge servers. It reported a latency of 5.12 milliseconds for this edge network, which can result from the addition of the latency of each device and waiting time in the network for each frame. The iFogSim simulation papers for the same application but different servers report a network delay close to zero for edge-only configuration and one camera (Fahimullah et al. 2023; Gupta et al. 2017).

Considering the specification of the real data shown in Figure 1, we measured the shape and scale parameters of the Gamma distribution as 38.97 and 0.19, respectively, which shown as Gamma (38.97, 0.19). We solved equations (1) and (2) above for the real arrival data that we collected and characterized. By using the same edge devices used in iFogSim simulation with a processing power of a minimum of 10 and a maximum of 17 fr/sec (equal to v1= 1/10 and v2=1/17 seconds), we calculated queuing delay or network latency. According to Equation (1) this system is stable (ρ<1) and with more than one server and the utilization factor ρ is decreased when increasing the number of servers. According to Equation (2) the queuing delay for Wq_{GGs}, for the same edge network used in the iFogSim simulator, calculated as 0.2 ms for three servers which is negligible and becomes very close to zero beyond four servers. We used Excel and manually calculated these equations 1 and 2, and the left graph of Figure 3 shows the network delay of G/G/s model simulation results. We calculated the network delay (i.e., queuing delay) of M/M/s model which has the available solution in the literature by using the web site Queuing Theory Calculator | Math

of Waiting (omnicalculator.com) to make cross-validation. The right graph of Figure 3 shows the M/M/s model generated the latency for three servers as 0.8 ms which is higher than G/G/s results. Thus, G/G/s queuing with Gamma and Unifrom distributions for arrival and service time is closer to zero, which we expect for edge computing AI-based video streaming applications where edge and IoT devices are close to each other. However, none of these models result in the network delay achieved by the iFogSim simulator, which adds the delay for fog devices (further than edge servers in a network).

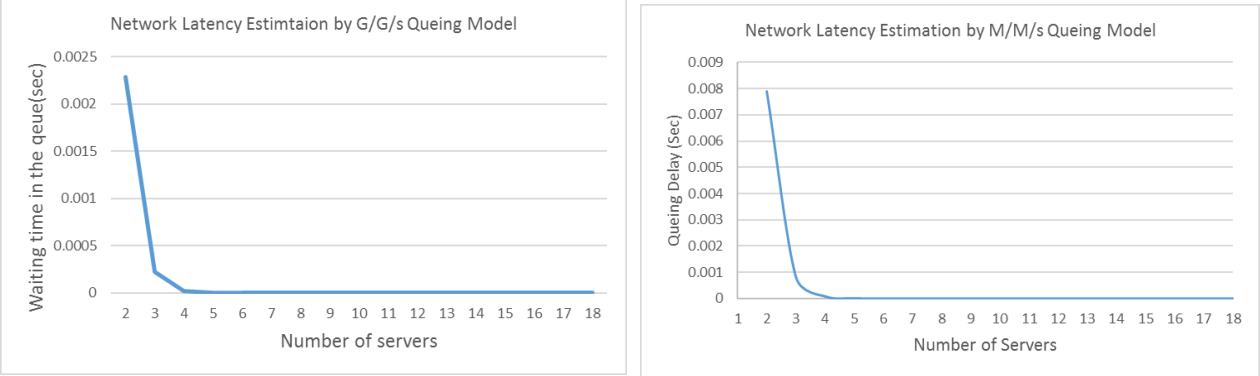


Figure 3: Queuing delay with increasing number of servers using G/G/s (left) and M/M/s (right) queuing models.

5 PROPOSED METHOD

As the general distribution cannot be found for collected data the final solution is to use an empirical distribution from the data. However, finding empirical distribution required collecting each application's data periodically and performing workload characterization. The novel idea presented in this paper is to use an artificial neural network (ANN) to automate workload characterization by estimating empirical distributions based on the periodically collected data from a real system. Automatic workload characterization includes finding the distribution models and their mathematical formulas provided by ANNs and Generative Adversarial Networks (GANs). These neural networks can learn from data to build a model and generate large amounts of data (i.e., synthetic data), which is considered a new era in Artificial Intelligence that made it possible to produce remarkably realistic and high-quality data in different formats, including text, photos, and videos. This feature not only makes synthetic data more realistic but also offers up a wide range of applications, from creating art that can be useful in many fields, including the gaming industry, to solving intricate data augmentation tasks in fields as varied as autonomous driving and medical (Goodfellow et al. 2014). In the rest of this section, we first discuss some problems with training GANs from the related works to provide background information. Then, we show the result of our work, which is focused on automatic data generation by estimating empirical distribution when the workloads of an edge-enabled application change.

Instability and mode collapse are two major training problems for classic GANs that can noticeably reduce their performance. Wasserstein GANs (WGANs) were developed as a solution to the classic GANs problems in which the method of measuring and optimizing distances across data distributions is reformulated, resulting in a more robust framework.

It has been demonstrated that applying a gradient penalty and the Wasserstein distance makes the training process more stable and results in a more reliable production of a variety of high-quality data. This development allows researchers and developers to fully utilize generative models while avoiding typical errors like mode collapse that will result in more dependable and adaptable solutions. In this work, we implemented the Wasserstein GAN with a gradient penalty.

There are two ways to enforce Lipschitz continuity: weight clipping and gradient penalty (Lui et al. 2020). In WGAN-GP, weight clipping is replaced by the gradient penalty, which addresses the drawbacks of the weight clipping method (Li et al. 2023). In this method, if the gradient norm deviates from the ideal

value of 1, the model will be penalized, and this procedure will result in a more stable and smooth training process (Arjovsky et al. 2017).

Gradient penalty fosters smooth gradients and helps ensure reliable convergence, whereas weight clipping may result in unexpected gradients and even cause training instability. WGAN-GP usually generates higher-quality synthetic images since the generator receives and is trained with more precise and reliable feedback. In the work done by Gulrajani et al. (2017), the authors proposed a method to improve the standard WGAN and enable stable training of a wide variety of GAN architectures. Their proposed method includes Gradient penalties implemented by computing the gradients of the critic function's output in relation to its inputs, which are a combination of generated and real data, and then adding a term to the loss function that penalizes the deviation of these gradients from the norm.

To test the feasibility of using generative adversarial networks to produce empirical distribution, we have used a simple generator in WGAN with a standard architecture responsible for creating synthetic data that resembles real data. It starts with a noise vector as its input and goes through a series of linear layers with ReLU activation functions. These layers transform the input noise vector into a more complex structure that eventually turns into the desired output shape. In the last layer, Tanh is used as the activation function, which normalizes the output to the range [-1, 1].

Figure 4, below shows the histogram of the generated data. The critic assesses the realism of the data produced by the generator by assigning it a score that reflects the quality of the generated data. The critic uses a series of linear layers that contain a batch normalization layer and a leaky ReLU activation and outputs a single value that shows the quality of the data. We used these parameters to improve the quality of generated data and make it closer to empirical distribution.

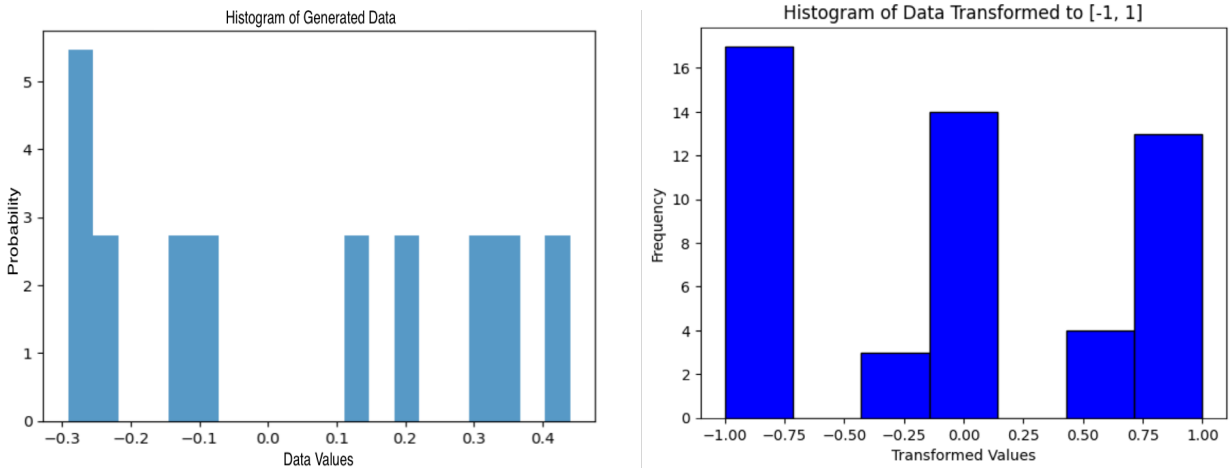


Figure 4: Histogram of empirical WGAN-generated data (left) and original data (right).

Figure 5 below shows the comparison of the distribution shape between the original and generated data. The limitation of our work, as Figure 5 shows, is that there is no perfect match between the empirical distribution and generated data by WGAN. The reason is the limited collected data, which is only 51 measured data in our edge-enabled video streaming applications. A large amount of collected real data is required, along with more advanced WGAN training for complex fog/cloud-based video streaming applications, which forms the future direction of this research.

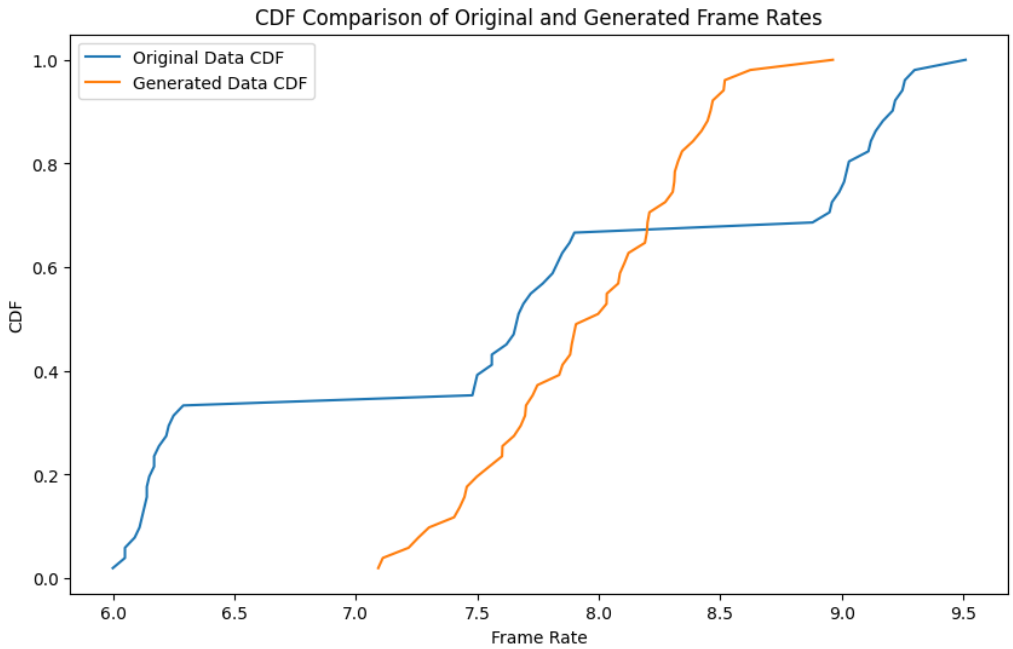


Figure 5: Comparison of CDF distributions of real data and WGAN-generated data.

6 CONCLUSION

In summary, the motivation of this work is to develop novel models that can be utilized in the simulation and evaluation of intelligent edge-enabled AI applications. The extensive use of AI techniques, such as Convolutional Neural Networks (CNN) in video analytics, frequently alters the workload pattern in edge-enabled applications. The models are needed to generate data automatically from these changing workloads.

In our recent experiments, we modelled the traffic patterns for different parts of a security camera IoT device application that was distributed to edge servers. We used G/G/s queuing systems with a general distribution of inter-arrival times and service times for *s* servers. In that application, network latency was negligible due to the proximity of IoT and edge devices. By comparing two models of G/G/s and traditional M/M/s queuing models, we found that using Gamma Distribution for arrival rate with the G/G/s model produces closer results to the previously simulated edge network and our expectation of the behaviour stability of the system. However, none of the distribution models were validated by simulation results because distributions cannot model real data accurately. The proposed solution in this work is to estimate the empirical distribution by using Generative Adversarial Networks called Wasserstein GANs (WGANs) to generate traffic patterns similar to the real data of DL-based video streaming applications running on top of edge/fog and cloud-based networks. In this work, we have developed a program to generate artificial data using WGANs to estimate the empirical distribution of the collected data for the tested application.

A potential direction for future work is to integrate the generative model-based workload characterization approach with popular edge computing simulators such as iFogSim or EdgeCloudSim. This would enable the evaluation of the impact of using synthetic workload data on the accuracy and fidelity of edge computing simulations. Future work could involve exploring more advanced generative models beyond WGANs. Models such as diffusion models or transformer-based architectures could be investigated to generate synthetic workload data. Comparing the performance and quality of data generated by different state-of-the-art generative models would be a valuable avenue for research. While this work focused on finding models for the simulation of edge-enable video streaming applications, the proposed approach of using generative models for workload characterization could be employed in real emerging applications such as the metaverse, digital twins, or blockchain-based systems. Evaluating the effectiveness of the methodology across diverse application domains would further demonstrate its generalizability.

ACKNOWLEDGEMENTS

I want to thank Sanaz Naseribonari, Kiana Kheiri and Laura Jamikeshova for contributing to this work.

REFERENCES

Abhari, A., D. Pudasaini. 2022. "Using Deep Learning for Simulation of Real time Video Streaming Applications". In *Proceedings of the 2022 Winter Simulation Conference (WSC),* 665-676.

Amadeo, M. G. Lia, A. Molinaro and G. Ruggeri. 2023. "In-network Placement of Reusable Computing Tasks in an SDN-based Network Edge". *IEEE Transactions on Mobile Computing* 23(2):1456-1471.

Arjovsky, M., S. Chintala, and L. Bottou. 2017. "Wasserstein Generative Adversarial Networks". *In Proceedings of the 34th International Conference on Machine Learning,* 70: 214–223.

Calheiros, R. N., R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya. 2010. "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms". *Software:Practice and Experience,* 41(1): 23–50.

CloudSim: A Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services. 2021. Available: http://www.cloudbus.org/cloudsim/. Accessed June 2024.

CloudSim Plus Simulator. https://github.com/manoelcampos/cloudsim-plus. Accessed June 2024.

CloudSim Plus. https://cloudsimplus.org. Accessed June 2024.

Deng, Q., M. Goudarzi, and R. Buyya. 2021. "FogBus2: A Lightweight and Distributed Container-based Framework for Integration of IoT-enabled Systems with Edge and Cloud Computing". in *Proceedings. of the International Workshop on Big Data in Emergent Distributed Environments*, 1-8.

Fahimullah, M., G. Philippe, S. Ahvar, and M. Trocan. 2023. "Simulation Tools for Fog Computing: A Comparative Analysis*". Sensors* 23(7): 3492.

Fan, W., L. Gao, Y. Su, F. Wu and Y. Liu. 2023. "Joint DNN Partition and Resource Allocation for Task Offloading in Edge–Cloud-Assisted IoT Environments". *IEEE Internet of Things Journal* 10(12): 10146-10159.

Giorno, V., A. Nobile and E. Pirozzi. 2018. "A state-dependent queuing system with asymptotic logarithmic Distribution". *Journal of Mathematical Analysis and Applications*, 458: 949-966.

Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. 2014. "Generative Adversarial Networks*". In Advances in Neural Information Processing Systems*, 2672–2680.

Gulrajani, I., F. Ahmed, M. Arjovsky, V. Dumoulin and A. Courville. 2017. "Improved Training of Wasserstein GANs.", https://arxiv.org/abs/1704.00028.

Gupta, H., A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya. 2017. "IFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in the Internet of Things, Edge and Fog Computing". *Software:Practice and Experience* 47(9): 1275–1296.

Jha, D.N., K. Alwasel, A. Alshoshan, X. Huang, R. K. Naha, S. K. Battula, *et al.* 2020. "IoTSim-Edge: A Simulation Framework for Modeling the Behavior of Internet of Things and Edge Computing Environments". *Software: Practice and Experience,* 50(6):844-867.

Li, H., J. Qian, Y. Tian, A. Rakhlin and A. Jadbabaie, 2023. "Convex and Non-convex Optimization under Generalized Smoothness". *Advances in Neural Information Processing Systems 36*, https://arxiv.org/abs/2306.01264.

Liu, K. and G. Qiu, 2020, "Lipschitz Constrained GANs via Boundedness and Continuity". *Neural Computing and Applications*, *32:*.18271-18283.

Medhi, J. 2003. *Stochastic Models in Queuing Theory.* 2nd ed. Academic Press, Amsterdam.

Pu, L., J. Shi., X. Yuan, X. Chen, L. Jiao, T. Zhang, and J. Xu. 2023. "EMS: Erasure-Coded Multi-Source Streaming for UHD Videos Within Cloud Native 5G Networks". *IEEE Transactions on Mobile Computing* 23(2): 1472-1487.

Silva Filho M. C., R. L. Oliveira, C. C. Monteiro, P. R. M. Inácio and M. M. Freire. 2017. "CloudSim Plus: A Cloud Computing Simulation Framework Pursuing Software Engineering Principles for Improved Modularity, Extensibility and Correctness". *IFIP/IEEE Symposium on Integrated Network and Service Management,* 400-406.

Tyokighir, S., J. Mom, K. Ukhurebor and G. Igwue. 2024. "New Developments and Trends in 5G Technologies: Applications and Concepts". *Bulletin of Electrical Engineering and Informatics,* https://doi.org/10.11591/eei.v13i1.6032.

Whitt W. 1983. "The queuing network analyzer". *Bell System Technical Journal* 62(9): 2779-2815.

AUTHOR BIOGRAPHY

ABDOLREZA ABHARI is a professor in the Department of Computer Science at Toronto Metropolitan University and director of DSMP lab (http://dsmp.ryerson.ca). He holds a Ph.D. in Computer Science from Carleton University, Canada. His research interests include data science, network simulation, web social networks, AI and agent systems, and distributed systems. His email is aabhari@torontomu.ca, and his website is https://www.cs.ryerson.ca/~aabhari/.