Generative Machine Learning for Resource-Aware 5G and IoT Systems

Nico Piatkowski
Fraunhofer IAIS
nico.piatkowski@iais.fraunhofer.de

Johannes S. Mueller-Roemer
Fraunhofer IGD & TU Darmstadt
johannes.mueller-roemer@igd.fraunhofer.de

Peter Hasse
Fraunhofer FOKUS
peter.hasse@fokus.fraunhofer.de

Adam Bachorek
Fraunhofer IESE
adam.bachorek@iese.fraunhofer.de

Tim Werner
Fraunhofer IOSB-AST
tim.werner@iosb-ast.fraunhofer.de

Pascal Birnstill
Fraunhofer IOSB
pascal.birnstill@iosb.fraunhofer.de

Andreas Morgenstern
Fraunhofer IESE
andreas.morgenstern@iese.fraunhofer.de

Lutz Stobbe
Fraunhofer IZM
lutz.stobbe@izm.fraunhofer.de

***Abstract*—Extrapolations predict that the sheer number of Internet-of-Things (IoT) devices will exceed 40 billion in the next five years. Hand-crafting specialized energy models and monitoring sub-systems for each type of device is error prone, costly, and sometimes infeasible. In order to detect abnormal or faulty behavior as well as inefficient resource usage autonomously, it is of tremendous importance to endow upcoming IoT and 5G devices with sufficient intelligence to deduce an energy model from their own resource usage data. Such models can in-turn be applied to predict upcoming resource consumption and to detect system behavior that deviates from normal states. To this end, we investigate a special class of undirected probabilistic graphical model, the so-called integer Markov random fields (IntMRF). On the one hand, this model learns a full generative probability distribution over all possible states of the system—allowing us to predict system states and to measure the probability of observed states. On the other hand, IntMRFs are themselves designed to consume as less resources as possible—e.g., faithful modelling of systems with an exponentially large number of states, by using only 8-bit unsigned integer arithmetic and less than 16KB memory. We explain how IntMRFs can be applied to model the resource consumption and the system behavior of an IoT device and a 5G core network component, both under various workloads. Our results suggest, that the machine learning model can represent important characteristics of our two test systems and deliver reasonable predictions of the power consumption.**

***Index Terms*—generative model, probabilistic graphical model, internet of things, 5G core, energy model**

I. Introduction

The current generation of IoT appliances is cloud-centered: sensor data is collected at the edge and offloaded, preprocessed, and analyzed on large centralized cloud systems. The amount of data that is transferred every day between IoT nodes and the cloud as well as the resulting energy consumption are enormous. Offloading even simple computational tasks to the cloud is thus nonsustainable and should be prevented.

This work was funded by the Fraunhofer research program on Trusted Resource Aware Information and Communication Technology, TRAICT.

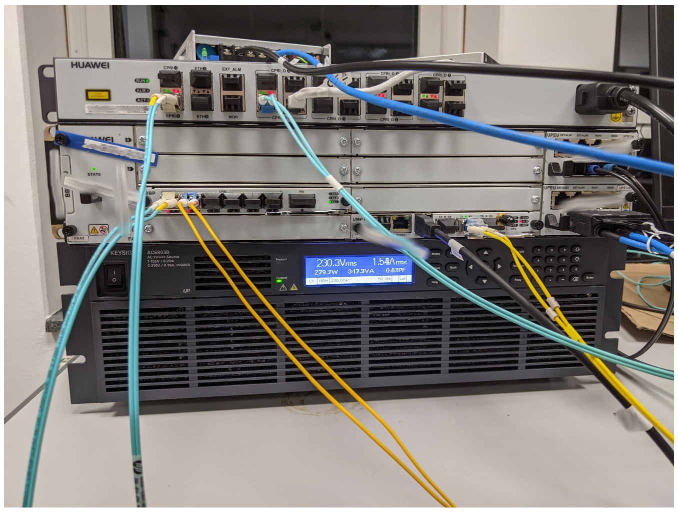

Fig. 1. Out custom measurement platform used for the evaluation of 5G core components. Huawei BBU5900 base band unit and Keysight AC6803B energy measurement device.

At the same time, recent advances in probabilistic machine learning deliver results in a quality that was previously unheard-of. Domains in which the output of machine learning can hardly be distinguished from real-world data include synthetic image generation and the generation textual natural language. However, the IoT and 5G community does not yet benefit from these methods due to their ludicrous resource requirements [1]. Learning a generative model of the system itself would yet allow us to improve its energy management without the need for hand-crafted rules.

We hence aim at the conception of probabilistic machine learning models for the energy consumption of IoT devices and 5G infrastructure hardware components. Once learned, such a model allows us to process various probabilistic queries. Typical applications include the prediction of a system's energy consumption based on its current usage behavior and the automatic detection of abnormal or malfunctioning behavior—core features of any resource aware and trustworthy system. Energy models can also be used in an early phase of design space exploration that is based on simulations [2]. For example, having a realistic energy model of the individual 5G components and coupling them in a holistic simulation model for a complete 5G mobile network with realistic user profiles would allow us to find the most power-efficient architectures for next-gen mobile networks. This requires to (1) acquiring appropriate training data, and (2) choosing an appropriate machine learning model. In the context of resource aware 5G systems, the resource consumption (in terms of computation, memory, and energy requirements) of the machine learning model shall be negligible compared to the primary function of the underlying system. It should be possible to run learning as well as inference in the system itself without a significant increase in the system's energy consumption. For this reason, integer Markov random fields (IntMRF) are considered, since they can approximate arbitrary probability mass functions while having only minimal hardware requirements, e.g. they have been applied successfully in ultra-low-power (ULP) micro controller units (MCU) [3].

Learning a probabilistic model is based on a data set of sensor measurements which reflect realizations of the underlying random variable of interest: In a 5G setting, this could include the configuration of a multiple input multiple output (MIMO) antenna array, current data rates, the system temperature, CPU utilization, and measurements of the current energy consumption.

In order to benefit from model predictions for controlling and optimizing energy consumption, we need to integrate energy consumption models into monitoring frameworks that are capable of intercepting and controlling computation and possibly communication processes. Such frameworks are subject of research in the area of cyber security under the term *usage control*. Usage control frameworks such as [4] allow us to specify constraints in so-called policies, i.e., logical rules. Actions intercepted by a monitoring component are evaluated against these policies, whereby the reasoning component can also draw on external sources of information and knowledge, such as requesting energy consumption predictions for observed actions with certain parameters or properties. For example, control measures like maintaining (increasing/decreasing) energy budgets for specific user, groups, purposes, or in reaction to specific events can be implemented based on such policy enforcement infrastructures, but also rescheduling of computation tasks so as to leverage an energy surplus in the network or temporarily cheap energy. In other words, energy consumption models can augment a usage control framework with energy consumption control capabilities so as to add a further dimension to control measures, which in particular supports resource-efficient computing.

In this work, we explain how to learn generative probabilistic machine learning models of the system behavior for IoT and 5G devices. To this end, we collected data from different systems and conducted a qualitative evaluation of generative machine learning models on these data sets. Measurements are taken from real 5G campus network hardware as well as state-of-the-art GPU accelerated IoT hardware as explained in Sec. III. Our results suggest, that the machine learning model can represent important characteristics of two test systems and deliver reasonable predictions of the power consumption.

II. Methodology

For sake of completeness, we provide an overview on learning and inference with integer Markov random fields [5] which belong to the broader class of exponential family models [3]. In what follows, $\boldsymbol{X}$ is a n-dimensional discrete random variable with state space $\mathcal{X}$. The central model equation is given by

$$\mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{X}=\boldsymbol{x}) = 2^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle - A(\boldsymbol{\theta})} \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{N}^d$ is a vector of model parameters, ϕ is a feature map into a reproducing kernel Hilbert space, and $A = \log_2 \sum_{\boldsymbol{x}\in\mathcal{X}} 2^{\langle \boldsymbol{\theta}, \phi(\boldsymbol{x})\rangle}$ is the partition function (also known as normalization constant). Due to discreteness of $\mathcal{X}$, ϕ is a d-dimensional binary vector with finite dimension d, and fully specified by the conditional independence structure G of $\boldsymbol{X}$. Note that this implies $\mathbb{P}_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{Q} \cap [0;1]$. Learning integer MRF consists of two step:

a) Step 1: Structure Learning.: Each and every multi dimensional random variable posses a conditional independence structure, representable by a graph $G=(V,E)$ with vertex set V and edge set E. Each vertex in V corresponds to one dimension of $\boldsymbol{X}$, and thus $|V|=n$. Whenever G is connected, all dimension of $\boldsymbol{X}$ are not mutually independent. Moreover, G tells us which variables need to be known to fully determine the distribution of any vertex v in V—namely the directly adjacent neighbors of v. An important quantity that underlies most structure learning algorithms is the multi-dimensional mutual information of a subset of variables $S \subseteq V$

$$\mathcal{I}(\boldsymbol{X}_S) = -\sum_{M\subseteq S} (-1)^{|M|} \mathcal{H}(\boldsymbol{X}_M)\ .$$

Here, sub-scripting a random variable with a vertex subset, e.g., $\boldsymbol{X}_S$, indicates the $|S|$-dimensional sub-vector of $\boldsymbol{X}$ that contains all dimensions in S in an arbitrary but fixed order. Moreover, $\mathcal{H}$ denotes the information entropy, defined via

$$\mathcal{H}(\boldsymbol{X}_M) = -\sum_{\boldsymbol{x}_M \in \mathcal{X}_M} \mathbb{P}(\boldsymbol{x}_M) \log_2 \mathbb{P}(\boldsymbol{x}_M)\ .$$

Generic procedures for estimating the conditional independence structure can be found in [6]–[8].

b) Step 2: Parameter Learning.: Once the conditional independence structure G has been discovered, the model parameters $\boldsymbol{\theta}$ are fitted to the training data via the regularized maximum-likelihood method. The learning problem stems from a constrained maximum-entropy principle. We aim at finding a probability distribution which is parametrized by integers and resembles the distribution of our training data points, while staying as close as possible to the uniform distribution. Hence, introducing as few assumptions on the data generating process as possible. The corresponding objective function is

$$\ell(\boldsymbol{\theta}) = A(\boldsymbol{\theta}) - \langle \boldsymbol{\theta}, \tilde{\boldsymbol{\mu}} \rangle + \lambda R(\boldsymbol{\theta}) \tag{2}$$

where $\tilde{\boldsymbol{\mu}} = {}^1\!/_N \sum_{\boldsymbol{x} \in \mathcal{D}} \phi(\boldsymbol{x})$ is the average feature space representation of the training data, $R(\boldsymbol{\theta}) = \sum_{i=1}^{d} 1 - |1 - 2(\lceil \boldsymbol{\theta}_i \rceil - \boldsymbol{\theta}_i)|$ is a regularizer that enforces the integrality constraint $\boldsymbol{\theta} \in \mathbb{N}^d$, and $\lambda > 0$ is the corresponding regularization weight. The regularizer $R(\boldsymbol{\theta})$ allows us to optimize ℓ as if $\boldsymbol{\theta}$ is real-valued while still generating an integral solution $\boldsymbol{\theta} \in \mathbb{N}^d$.

The gradient of the smooth part of ℓ (R is nonsmooth) is given by

$$\nabla \ell^S(\boldsymbol{\theta}) = \hat{\boldsymbol{\mu}} - \tilde{\boldsymbol{\mu}} \tag{3}$$

where $\hat{\boldsymbol{\mu}} = \sum_{\boldsymbol{x} \in \mathcal{X}} \mathbb{P}_{\boldsymbol{\theta}}(\boldsymbol{x}) \phi(\boldsymbol{x})$ is the expected feature space representation of the model (1). Equation (3) is precisely the only step where the training data enters the learning procedure. How to prepare training data from raw measurements (including the discretization) is discussed in Section III.

The objective function (2) is then minimized via an proximal first-order method [9]–[11]. As shown in [3], the whole training procedure can be implemented such that only integer arithmetic is required and that all intermediate results are integers—this is the main reason for why this model class is especially well-suited for resource constrained devices. The only requirement to achieve integrality of the training procedure is the integrality of the inference step (which is executed various times during training).

A. *Probabilistic inference*

Given a structure G and some (not necessary optimal) parameter vector $\boldsymbol{\theta}$, an inference algorithm can be applied to compute conditional and unconditional marginal probabilities for arbitrary subset of V. This step is also required for the gradient computation during training, since it can be shown that $\hat{\boldsymbol{\mu}}$ from (3) consists of all marginal probabilities for all cliques (maximally connected vertex subsets) of G. In what follows, we assume that G is a tree structure to simplify notation. Various techniques exist which rely on simplifying assumptions w.r.t. the structure or the model parameters [12]. The inference algorithm presented here consists of sending so-called messages $m_{s \to t}$ from each vertex s to all of it's neighboring vertices t—we denote the set of s's neighbors by $\mathcal{N}(s)$.

$$m_{s \to t}(y) = \mathrm{bl} \sum_{x \in \mathcal{X}_s} 2^{\boldsymbol{\theta}_{st=xy} + \sum_{w \in \mathcal{N}(s) \setminus \{t\}} m_{w \to s}(x)} \tag{4}$$

Here, bl denotes the bit-length of an integer and $\boldsymbol{\theta}_{st=xy}$ is the model weight for having vertices s and t jointly in the states x and y, respectively. Since messages are defined recursively, they must be re-computed in an arbitrary order until convergence, which is guaranteed by assuming that G is a tree structure.

The numbers that appear as intermediate results during the computation of (4) can become quite large. However, the terms in the summation are integer powers of two, which implies that the result will contain at most $|\mathcal{X}_s|$ 1-bits. We can hence compute the bit-length of very large intermediate numbers with only $|\mathcal{X}_s|\omega$ bits, where ω is the word-size of the underlying compute architecture.

Finally, probabilistic inference is the main reason why we choose MRFs as our underlying machine model. Discriminative learning models (like support vector machines or feed forward neural networks) have been studied in the context of resource limitations [13]. However, they have a fixed input-output relation: given observed input variables $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, they predict their output variable(s) $\boldsymbol{y}$. In generative models (like MRFs), these roles are not fixed. They give us the freedom to solve energy modelling, anomaly detection, simulation, and other tasks with the very same model—without the need for learning a specialized model for each task separately.

TABLE I
PROBLEM SIZES OF THE FE SIMULATIONS THAT WERE RUN DURING DATA COLLECTION. NNZ DENOTES THE NUMBER OF NONZERO 3×3-BLOCKS IN THE SPARSE SYSTEM MATRIX.

i	Nodes	Elements	NNZ
0	2872	7749	29848
1	2872	7749	29848
2	1044195	5735785	15037425
3	144761	719084	1979697
4	210482	1071293	2914402
5	326218	1702793	4575636
6	548555	2935282	7792611
7	2872	7749	29848
8	69636	312441	899582
9	259486	1351435	3619154
10	66840	324561	899478

III. DATA

Data from two sources is collected and pre-processed for learning the IntMRF: an NVIDIA Jetson board and a Open5GCore [14] running on an Intel(R) Core(TM) i5-9500T in combination with an Huawei BBU5900 and indoor cells. Energy consumption of the Jetson board is measured with the internal sensors while energy consumption of the 5G platform is measured with the CPU built-in sensors via RAPL as well as external energy meters (AC6830B).

A. *IoT Device*

The IoT raw data is collected on an NVIDIA Jetson AGX Xavier Developer Kit [15]. The hardware platform contains an 8-core ARM v8.2 64-bit CPU with 8MB L2 and 4MB L3 Cache. In addition, the system is endowed with a 512-core Volta GPU, 32GB LPDDR4x main memory, and 32GB eMMC 5.1 flash storage for the operating system and additional applications.

Workload is generated via the finite element (FE) simulation RISTRA [16]. RISTRA is a highly optimized, GPU-accelerated FE simulation package that performs both system assembly [17] and solution [18] on the GPU, while making use of the 3×3-block structure of the system matrices [19].

Multiple series of measurements are conducted, where each measurement is composed of the following steps:

1) Select a POWER-MODE (1–6), setting the number of active cores and frequency ranges for the Jetson board.
2) Bring the system to an IDLE mode, i.e., a mode in which the system has no significant workload. We let the system idle for at least $120\,\mathrm{s}$ between simulation runs to ensure initial temperatures and clock rates are similar between measurements.
3) Begin recording system information via the NVIDIA `tegrastats` tool every $100\,\mathrm{ms}$. The sampling rate of $100\,\mathrm{ms}$ was chosen to minimize measurement overhead while providing a sufficient number of samples to resolve individual simulation modes (see below). The sensed values include indicators for the workload, like CPU utilization and memory consumption, as well as multiple temperature and energy measurements for various system components. Available sensors/measurements are: RAM currently in use, largest free block size (max. 4MB), number of free blocks available to allocator, swap memory in use, amount of swap memory cached, per core CPU utilization in %, per core CPU clock frequency in MHz, GPU usage in %, always-on temperature sensor located near SOC, GPU cluster center temperature sensor, external sensor near SoC center, temperature near computer vision cluster, CPU cluster center temperature sensor, weighted average of temperatures (for fan control), external board temperature sensor, GPU power draw, CPU power draw, power draw of misc. SoC clusters, DRAM controller power draw, and the power draw of other board components. We augment the provided measurements with a timestamp by piping the unbuffered output into a line timestamping tool.
4) After $\approx 10\,\mathrm{s}$ of recording, start RISTRA with configuration τ_i to generate workload on the device. The simulation software can be in different *modes* like, loading, simulation, or post-processing. In addition to the system sensors, the mode of the simulation is timestamped and logged whenever a mode transition occurs. An example is shown in Fig. 2. To simulate a varying load, after starting RISTRA, the system is assembled once and simulations are performed 10 times, with $10\,\mathrm{s}$ breaks in between, where RISTRA remains running, keeping the system matrix in memory.
5) Stop recording of system information $\approx 10\,\mathrm{s}$ after the workload has ended.

This has been carried out for 11 different configurations $\tau_0, \dots, \tau_{10}$ of the FE simulation. Each configuration covers a different model complexity. Thus, each configuration will incur a different workload. Most relevant for the workload are the problem sizes which are solved by the FE simulation—corresponding values can be found in Tab. I[1]. For each τ_i, measurement was repeated 11 times. A data sample is a point in time for which all sensor measurements are known. In total, 4060414, 37-dimensional raw data samples have been collected on the IoT device.

B. 5G Core Network Component

The logical architecture of our custom 5G measurement platform consists of diverse monitoring facilities and measurement instruments, the measurement system as well as the actual system under test (SUT). In the focus of the considered measurement campaign is the 5G core network based on the Open5GCore implementation configured for campus/edge mode of operation. The tested hardware is deployed on 4 SUTs which are run by LX2160A 16 Core ARM64 A72, AMD Ryzen 5 PRO 3400GE (8 Core x86/AMD64), Intel Core i5-9500T (6 Core x86/AMD64), and Intel Xeon D-2123IT (8 Core x86/AMD64), respectively[2]. This is complemented by a base band unit (BBU), remote radio head (RRH) and a radio hub (confer Fig. 1). External energy measurements are performed by an AC6803B from Keysight via the VISTA protocol whereas internal measurements of the energy consumption caused by CPU, RAM, and GPU are conducted via the PERF / RAPL API [20]. In order to measure the user equipment, RDTech UM25C are utilized. Measured data values are stored in an InfluxDB timeseries database. Multiple repetitions of experiment have been conducted which consist of:

1) User hardware (5G-UE) is attached to the core network
2) Four scenarios are executed: traffic is send for 60 seconds in upstream (`5G_UP_1`) and downstream (`5G_DOWN_1`) direction. 1024MB of data is transmitted in upstream (`5G_UP_2`) and downstream (`5G_DOWN_2`) direction.
3) Measurement data is collected and pushed to the database. Available sensors/measurements are: BBU power consumption, core network power consumption, 5G downstream (MBit/s), 5G upstream (MBits/s), RAPL power consumption (CPU cores), RAPL power consumption (package), RAPL power consumption (RAM), CPU usage system, and CPU usage user. An additional variable encodes the scenario with values in $\{0, \texttt{5G\_UP\_1}, \texttt{5G\_UP\_2}, \texttt{5G\_DOWN\_1}, \texttt{5G\_DOWN\_2}\}$, where 0 indicates an IDLE mode. All measurements are timestamped.

In total, 106668 raw data samples have been collected.

C. Pre-processing

Let us now explain how the raw data samples from the IoT device and the 5G system are pre-processed for learning a generative model of the underlying data generating processes.

While the setup of the IoT system delivered a valid value for all sensors every $100\,\mathrm{ms}$, the sensors of our 5G infrastructure component have different temporal resolutions. That is, the system and user CPU utilization are recorded every $300\,\mathrm{ms}$

[1]The mesh used in models 0, 1 and 7 is identical, with varying convergence threshold between 10^{-4} and 10^{-6}. The other models use 10^{-5} consistently.

[2]Due to access restrictions while COVID19 lock down, measurements used in this paper could only be conducted on the Intel i5 platform.

```
3143178 RISTRA_STARTING
3143254 RISTRA_LOADING_MODEL
3143493 RISTRA_PREPARING_SOLVER
3144243 RISTRA_SIMULATING
3144397 RISTRA_POSTPROCESSING
3144400 RISTRA_IDLE
3154400 RISTRA_SIMULATING
3154548 RISTRA_POSTPROCESSING
...
3235721 RISTRA_SIMULATING
3235895 RISTRA_POSTPROCESSING
3235898 RISTRA_IDLE
3245958 RISTRA_STOPPED
```

Fig. 2. Timestamped mode sequence, generated by the RISTRA FE simulation.

while the other sensors are queried once per second. Moreover, measurements of RAPL, 5G network traffic, and BBU power are not synchronized. All measurements where hence aggregated by taking the maximum measured value of each sensor during one second. After aggregation, $N_{\mathrm{5G}} = 35521$, $n_{\mathrm{5G}} = 10$-dimensional samples are available for learning the energy model of the 5G system. Since no aggregation has been performed on the IoT data, all $N_{\mathrm{IoT}} = 4060414$, $n_{\mathrm{IoT}} = 37$-dimensional samples are available for learning the energy models of the IoT system.

Integer Markov random fields are discrete models—they have no native representation for numerical data. We thus employ an equal frequency binning with $k \in \{64, 128\}$ bins as follows: For each sensor, we sort all measured values and partition the full data range into k subsets, such that all subsets have almost[3] identical size. Within each bin $b_i, i \in \{1, 2, \dots, k\}$, we estimate the parameters μ_i and σ_i of a Gaussian distribution. This allows us to convert discrete states of the integer model back to the original numerical domain, e.g., via sampling from the corresponding Gaussian truncated at the bin-boundaries. The resulting discretized data sets are denoted by $\mathcal{D}^k_{\mathrm{IoT}}$ and $\mathcal{D}^k_{\mathrm{5G}}$ for $k \in \{64, 128\}$.

IV. Numerical Evaluation

We conduct the following numerical experiment to exemplify the capabilities of the proposed method. Integer Markov random fields are learned for both IoT and 5G scenarios, by minimizing Eq. (2) for 1000 iterations of the proximal gradient algorithm. The corresponding conditional independence structures are learned with the Chow-Liu algorithm [6]. For training, the data sets $\mathcal{D}^k_{\mathrm{IoT}}$ and $\mathcal{D}^k_{\mathrm{5G}}$ are used. We kept the last 1000 data points out of the learning procedure. After learning, we take those 1000 hold-out data points and use the probabilistic model to sample all sensor values, given the observed system mode. For the IoT system, the observed mode is given by the internal mode of the RISTRA simulation (cf. Section 2) together with its parameters τ_i. For the 5G setup, the system mode is defined by the upstream/downstream scenario as described in Section III-B. Excerpts of our results are visualized in Fig. 3. Therein, the first two plots (from left to right), show the energy consumption and the downstream 5G utilization of the 5G system over time. The third plot shows the power draw of the IoT system's CPU over time. Light grey points show actually measured values from the data set. Dark grey points show values which are sampled from the learned probabilistic model. The background color indicates the mode of the system—those modes are the only input to the probabilistic model for generating the samples from $\mathbb{P}_{\boldsymbol{\theta}}$. We see that the distribution of the 5G system samples matches the distribution of the ground truth data, which indicates that our model approximates the underlying data generating process well. About 5% of all samples of the energy consumption for the IoT model overshoot the true consumption, which might indicated the the learned Chow-Liu structure is not sufficient and that a higher-order model is required. Nevertheless, 95% of all samples are well-behaved.

It is, however, important to understand that due to the generative nature of our model, the *prediction error* and other metrics that we usually consult to assess the quality of a machine learning model are meaningless in the generative setup at hand. As mentioned before, the only information that we provide to the model is the mode of the system, e.g., `RISTRA_PREPARING_SOLVER` for the IoT system (cf. Fig. 2) or `5G_DOWN_2` for the 5G system. Since we employ our model to sample from $\mathbb{P}_{\boldsymbol{\theta}}(\ \cdot \mid \texttt{mode})$, a reasonable quantitative quality measure is the log-likelihood itself, i.e., Eq. (2) without the regularization term $\lambda R(\boldsymbol{\theta})$. The average (instance-wise) log-likelihood, is -22.405 for the 5G model and -18.655 for the IoT model.

V. Discussion

We presented a novel generative approach for modelling the resource consumption of IoT and 5G systems. The resulting model can be used to address various tasks like predicting the upcoming energy consumption of a system or detecting deviation from normal system behavior. Data has been collected on real 5G and IoT platforms under benchmark workloads. Numerical experiments verified, that models which are learned on this data can faithfully represent the distribution of power consumption and bandwidth utilization of a 5G system.

One way to apply our model is as part of an intelligent energy management, e.g., to decide if CPU or bandwidth has to be throttled to stay within a user specified energy budget. Another interesting application of our model is in the context of simulation.

The most widespread used approach in the simulation community is the manual creation of behavior models using mathematical-physical modeling in the sense of a white box model of the subsystems and processes involved. Often this is combined with some experimental steps where the input and output variables of a system are evaluated on a test-bench and the so-captured data is fed back into the white-box model to make it more accurate. This usually involves

[3]A perfect split in which all sets have equal size is only possible if N is a multiple of k.

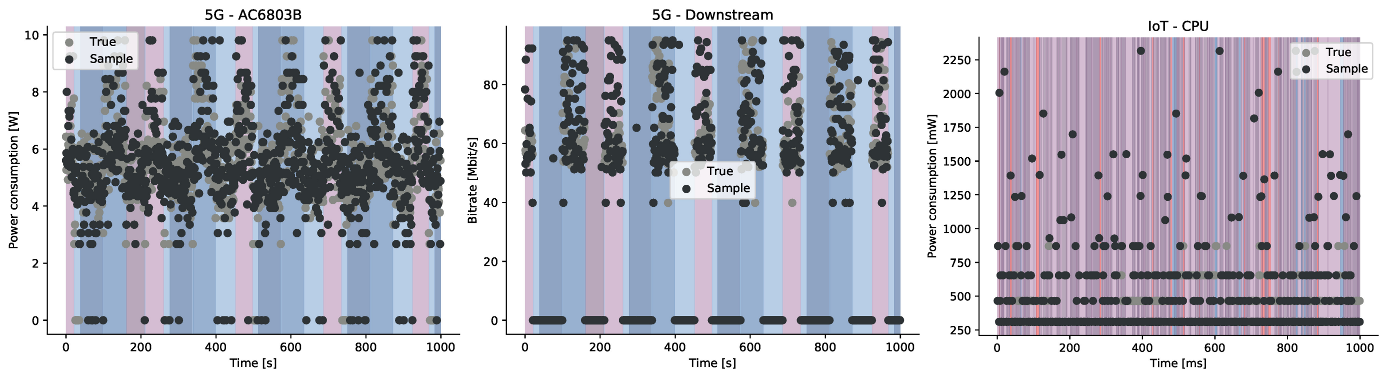

Fig. 3. Visual comparison of ground truth and samples from the machine learning model for the 5G core network component (left and mid), as well as the IoT system (right). Background colors indicate different modes of the system.

a lot of manual effort and while it often leads to highly accurate models, this comes with the price of low execution performance. For example, [21] discusses energy simulations of CPUs based on the Gem5 simulator. The Gem5 Simulator [22] is computer architecture simulator system that provides simulation models of ARM CPU cores, DRAMs, caches, I/O devices and more. While this approach is possible up to the level of individual devices like a single IoT Device, it is not possible to use this approach in a holistic system of systems simulation. Here, we could apply a *learned energy model* instead. Moreover, in a 5G-setting, a simulation down to the individual CPU cycles is not necessary. What we're instead aiming at are functional models of the individual entities that are coupled with the energy models that we proposed in this article. Finally, since we use lightweight machine learning models, our methods may run autonomously inside network components, e.g., smartphones or 5G base stations, to facilitate an intelligent energy management in the next generation of network devices.

References

[1] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Conference of the Association for Computational Linguistics*, 2019, pp. 3645–3650.

[2] L. von Rueden, S. Mayer, R. Sifa, C. Bauckhage, and J. Garcke, "Combining machine learning and simulation to a hybrid modelling approach: Current and future directions," in *International Symposium on Intelligent Data Analysis*. Springer, 2020, pp. 548–560.

[3] N. Piatkowski, "Exponential families on resource-constrained systems," Dissertation, Fakultät für Informatik, TU Dortmund, 2018. [Online]. Available: http://hdl.handle.net/2003/36877

[4] A. Pretschner, M. Hilty, and D. Basin, "Distributed usage control," *Communications of the ACM*, vol. 49, no. 9, pp. 39–44, 2006.

[5] N. Piatkowski, S. Lee, and K. Morik, "Integer undirected graphical models for resource-constrained systems," *Neurocomputing*, vol. 173, Part 1, pp. 9–23, 2016.

[6] C. K. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, May 1968.

[7] E. Kovács and T. Szántai, *On the Approximation of a Discrete Multivariate Probability Distribution Using the New Concept of t-Cherry Junction Tree*. Springer Berlin Heidelberg, 2010, pp. 39–56.

[8] K. Rantanen, A. Hyttinen, and M. Järvisalo, "Learning chordal Markov networks via branch and bound," in *Advances in Neural Information Processing Systems*, I. Guyon, U. v. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 1847–1857. [Online]. Available: http://papers.nips.cc/paper/6781-learning-chordal-markov-networks-via-branch-and-bound.pdf

[9] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, no. 2, pp. 372–376, 1983.

[10] ——, "Smooth minimization of non-smooth functions," *Mathematical Programming*, vol. 103, no. 1, pp. 127–152, 2005.

[11] N. Parikh and S. Boyd, "Proximal algorithms," *Foundations and Trends in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[12] N. Piatkowski and K. Morik, "Stochastic discrete clenshaw-curtis quadrature," in *International Conference on Machine Learning (ICML)*, ser. JMLR Workshop and Conference Proceedings, M. Balcan and K. Q. Weinberger, Eds., vol. 48. JMLR.org, 2016, pp. 3000–3009. [Online]. Available: http://proceedings.mlr.press/v48/piatkowski16.html

[13] B. Sliwa, N. Piatkowski, and C. Wietfeld, "LIMITS: lightweight machine learning for iot systems with resource limitations," in *IEEE International Conference on Communications (ICC)*, 2020, pp. 1–7.

[14] Fraunhofer FOKUS, "Open5GCore – open5gcore – the next mobile core network testbed platform." [Online]. Available: https://www.open5gcore.org/

[15] NVIDIA Corporation, "Jetson AGX Xavier Developer Kit," 2018. [Online]. Available: https://developer.nvidia.com/embedded/buy/jetson-agx-xavier-devkit

[16] Fraunhofer IGD, "RISTRA – Rapid interactive structural analysis." [Online]. Available: https://www.igd.fraunhofer.de/en/projects/ristra-rapid-interactive-structural-analysis

[17] J. Mueller-Roemer and A. Stork, "GPU-based polynomial finite element matrix assembly for simplex meshes," *Comput. Graph. Forum*, vol. 37, no. 7, pp. 443–454, 2018.

[18] D. Weber, J. Bender, M. Schnoes, A. Stork, and D. W. Fellner, "Efficient GPU data structures and methods to solve sparse linear systems in dynamics applications," *Comput. Graph. Forum*, vol. 32, no. 1, pp. 16–26, 2013.

[19] J. S. Mueller-Roemer, A. Stork, and D. Fellner, "Analysis of schedule and layout tuning for sparse matrices with compound entries on GPUs," *Comput. Graph. Forum*, vol. 39, no. 6, pp. 133–143, 2020.

[20] Intel CORPORATION, "RAPL – running average power limit." [Online]. Available: https://01.org/blogs/2014/running-average-power-limit-–-rapl

[21] B. K. Reddy, M. J. Walker, D. Balsamo, S. Diestelhorst, B. M. Al-Hashimi, and G. V. Merrett, "Empirical CPU power modelling and estimation in the gem5 simulator," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[22] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," vol. 39, no. 2, p. 1–7, Aug. 2011. [Online]. Available: https://doi.org/10.1145/2024716.2024718