Assessing Building Control Performance Using Physics-Based Simulation Models and Deep Generative Networks

Ankush Chakrabarty[†], Luigi Vanfretti, Wei-Ting Tang, Joel A. Paulson, Sicheng Zhan, Scott A. Bortoff, Vedang Deshpande, Ye Wang, and Christopher R. Laughman

***Abstract*—Despite the increasing sophistication of building simulation models and digital twins, some components of the building energy system remain challenging to model using first-principles. For instance, internal heat loads generated by occupants in buildings and their usage of the building are often *assumed* rather than *learned*. Consequently, building control systems are designed based on average human behavior, and the assessment of the closed-loop system is often limited to a small set of handcrafted scenarios. In this paper, we propose the use of deep generative networks to complement the physics-based simulation platforms by learning time-series distributions from real occupancy and usage data. The learned distribution can subsequently be sampled to construct scenarios that can drive the building simulation model as a disturbance input. This capability enables the more systematic construction of a large set of scenarios for controller performance assessment. Due to the expense of the simulator and the unknown relationship between the disturbance inputs and performance, we provide a sample-efficient algorithm for extracting 'limiting scenarios', i.e., disturbance inputs that are most likely to result in the best and worst closed-loop performance. We demonstrate the potential of our proposed framework using Mitsubishi Electric's SUSTIE building data, on a building simulation benchmark model implemented in Modelica. We report that the generated scenarios preserve signal features observed in the true data while enabling the automatic identification of low-probability scenarios for controller evaluation, and our sampling method determines these limiting scenarios using only 300 simulations.**

I. Introduction

Physics-based building simulation tools are often used to size equipment during building design, and also to monitor building performance; for example, to predict annual energy consumption or carbon footprint. Recently, sophisticated building simulation models (sometimes referred to as 'digital twins') have been developed such as the Modelica Buildings Library [1] based on an acausal, equation-based paradigm, which allow for simulation of the coupled dynamic behavior of building envelopes and HVAC systems [2], [3]. These simulations are extremely valuable: they can enable systematic controller design [4], and provide a cheap and fast alternative to field experiments for evaluating closed-loop control performance [5].

The basic physics of heat transfer by conduction and radiation, and mass transport by advection, are well known and can be transcribed accurately in these modeling libraries. As such energy and mass transfer through walls and windows can be accurately simulated, together with the thermo-fluid physics of modern HVAC systems. However, this is only part of the heating and cooling load. Building occupants also produce and absorb latent, sensible and radiative heat, and their behavior can strongly affect the performance of an HVAC system as measured by energy consumption, human comfort, indoor air quality etc [6]. Typical building simulation platforms must *assume* this behavior a priori, or consider some 'nominal' behavior for convenience. For example, the modeler assumes an occupancy (number of people occupying a zone) and their activity level and schedule, represented as an input disturbance, in order to execute a simulation. Human behavior of this type is not practically amenable to physics-based, or more generally first-principles', modeling.

Owing to the proliferation of occupancy and thermal usage data in modern buildings [7], deep generative networks have proven effective at capturing the distribution of single-output operational building profiles, including energy consumption [8], thermal comfort [9], and occupancy profiles [10]. While most prior work has only assessed the intrinsic quality of the learned distributions or showed their usefulness as a data augmentation tool for forecasting models [11] and controllers [12], some recent work has also showcased interesting applications including controller validation [13]. These generative models can be used to generate a variety of building simulation scenarios, such as simulating nominal or corner-case occupancy patterns for a particular building, perhaps in order to identify opportunities for improvement, or building models of occupancy that can be used in building design, HVAC product development, or controller performance assessment.

In this paper, we use a recently proposed generative modeling architecture called RAFT-VG [14] (regularized adversarial fine-tuned VAE-GAN) that combines the benefits of variational autoencoders (VAEs) and generative adversarial networks (GANs) for modeling occupancy and internal heat loads from real data collected from Mitsubishi Electric's SUSTIE[*] building. A contribution of this work is in the

[†]Corresponding author. Phone: +1 (617) 758-6175. Email: achakrabarty@ieee.org.

LV is affiliated with Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute, Troy, NY, USA.

WTT and JAP are affiliated with the William G. Lowrie Department of Chemical and Biomolecular Engineering at Ohio State University, Columbus, OH, USA.

SZ is affiliated with the Department of The Built Environment at the National University of Singapore.

All other authors are affiliated with Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA.

[*]https://www.mitsubishielectric.com/en/about/rd/sustie/index.html

integration of this deep generative network into Modelica for closed-loop simulation. This network learns a distribution in a latent space that can be decoded to form 'scenarios': occupancy and heat load time-series, which in turn can be used to automatically execute closed-loop building simulations. Another major contribution of this work is to ascertain latent vectors which, when decoded, result in *limiting scenarios* that result in extremely good or bad closed-loop simulation performance: we do this using an information-theoretic Bayesian Algorithmic eXecution sampling method called InfoBAX [15]. Obtaining such limiting scenarios systematically, and without manual intervention, is critical to an efficient controller design pipeline.

II. Problem Statement and Proposed Solution

Many dynamic simulators or "digital twins" of building energy systems can be abstracted by a model of the form

$$x_{t+1},\, y_t = \mathcal{M}(x_t, u_t, d_t^{\text{ext}}, d_t^{\text{int}}), \tag{1}$$

where $x_t \in \mathbb{R}^{n_x}$ denotes a state of the building dynamics, $u_t \in \mathbb{R}^{n_u}$ denotes the control inputs to the system such as heating, ventilation, and cooling (HVAC), and d_t^{ext}, d_t^{int} are disturbance inputs to the system generated either externally to the system, or internally within the system, respectively. While making this distinction between the disturbance inputs is not always necessary, we do so in this paper because external disturbances such as ambient conditions are simulated directly from standard weather files, whereas generating internal disturbances such as heat loads and occupancy patterns are the subject of our study. The simulation model $\mathcal{M}$ is often based on first-principles (e.g., physics-based) that accurately reflects the thermal dynamics of the building zones. We treat $\mathcal{M}$ as a black-box.

Some outputs $y_t \in \mathbb{R}^{n_y}$ of the simulation model are available for online measurement: these outputs may inform the controller for regulation/tracking, or for assessing the scalar performance output J of the closed-loop building simulation, usually by evaluating a performance function f. That is

$$J = f(\{y_t\}_{t=0}^{T_f}), \tag{2}$$

where T_f is the total time of closed-loop simulation over which the performance output is computed. Often, these closed-loop simulations are driven by control algorithms that have been designed for nominal scenarios, e.g. by selecting d_t^{int} and d_t^{ext} to be some nominal values based on domain experience or a simplifying statistic (such as the mean) of observed data. Even if such a controller based on a nominal scenario is effectively designed, it may not will perform equally well under a wide range of scenarios. For example, if the controller was designed assuming a zone has some nominal occupancy, there is no reason to expect it will perform adequately if the zone's occupancy is completely empty or filled to capacity.

Constructing scenarios with the specific objective of learning where the closed-loop control system will perform at its limits (extremely poor or extremely good performance) requires significant computational effort or intuition honed by significant domain expertise; neither strategy scales well if selecting candidate scenarios involve designing a large number of time-series signals and is not a systematic strategy across different building simulation models. Our **objective** in this paper is to provide a systematic framework for generating scenarios based on relevant building data, and to construct specific *limiting scenarios* which, when driving a high-fidelity simulator, is expected to result in the limits of closed-loop system performance. For any such method to be practical, we require it to be generalizable to any simulation model, and therefore, agnostic to the type of simulation model, as long as it can be represented by (1).

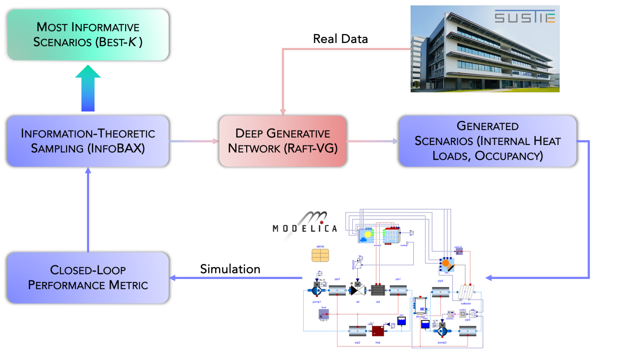
Most Informative Scenarios (Best-K)
Real Data
SUSTIE
Information-Theoretic Sampling (InfoBAX)
Deep Generative Network (RAFT-VG)
Generated Scenarios (Internal Heat Loads, Occupancy)
MODELICA
Closed-Loop Performance Metric
Simulation

Fig. 1. Overview of the proposed solution. The pink arrows denote infusion of real data via deep generative networks, whereas the blue arrows are operations involving high-fidelity simulations. Specific implementations used in this work are parenthesized.

To this end, we propose the framework pictured in Fig. 1. Assuming we have available a simulation model $\mathcal{M}$ and access to real scenario data $\mathcal{D} := \{d_t^{\text{int}}\}$, we propose the use of deep generative models to learn the underlying distribution that has generated this data. In this work, we consider a combination of a variational autoencoder (VAE) fine-tuned by a discriminator as in GANs; we refer to our architecture as RAFT-VG [14] and will describe this generative network in more detail in the next section. The RAFT-VG is trained to learn a distribution in its latent space that, when decoded, is 'close' (in some metric) to the true data distribution. Additionally, the generative model allows conditioning, that is, the distributions that are learned can be conditioned on relevant factors (e.g., workday/holiday, seasonality) that are expected to change the underlying data distribution. Therefore, sampling from this latent space and passing the latent sample through a trained decoder network results in synthetic d_t^{int} scenarios, such as internal heat loads and occupants in building zones, that can drive closed-loop building simulations. Such simulations can be used to ascertain a closed-loop performance metric. Effectively, the generative network and the physics-based simulator can be viewed as a black-box map from scenarios to performance metrics. Consequently, one can use efficient sampling methods for black-box functions such as Bayesian algorithm execution [15] to obtain informative scenarios by learning interesting regions in the latent space from a few simulations. In this paper, we use InfoBAX (a variant of BAX) to automatically learn K scenarios that result in the best- and worst-case constraint violation on room temperature.

In the following section, we delve deeper into the RAFT-VG architecture and the InfoBAX algorithm, and demonstrate how they can be combined to ascertain these 'limiting' scenarios, i.e., where the performance of the closed-loop system is at its limit.

III. Generation and Identification of Limiting Scenarios via Deep Generative Models

A. Generative modeling of scenarios with RAFT-VG

In this work, we adopt the generative modeling approach of [14], which applies regularized adversarial fine-tuning (RAFT) that combines the benefits of Variational AutoEncoders (VAE) [16], [17] and Generative Adversarial Networks (GAN) [18], [19]. This method is a two-stage approach, where we first train a conditional VAE in a conventional manner, and then we fine-tune the decoder with adversarial training against a discriminator, in a manner similar to Wasserstein GAN (WGAN) [20] with spectral normalization [21]. Additionally, the second-stage fine-tuning is regularized by limiting how much the decoder parameters may change, which aims to preserve consistency with the encoder. This approach combines the relative ease of training a VAE in the first stage, with the generative quality benefits of GAN-like adversarial fine-tuning in the second stage, while the regularization helps to preserves VAE consistency and improves the stability of fine-tuning. In the following, we briefly describe the steps of this method, while further details can be found in [14].

The first stage consists of training a conditional VAE that implicitly captures the distribution $p_\theta(w|s)$, where $w := \{d_t^{\mathsf{int}}\}_{t=0}^T$ is shorthand for the disturbance time series being modelled and s is a conditioning variable related to that time series. The conditional VAE provides a generative model specified by the distribution $p_\theta(w|s,z)$, where z is latent representation sampled from the latent prior distribution $p(z)$. Together, these implicitly (and intractably) determine the conditional model distribution,

$$p_\theta(w|s) = \int p_\theta(w|s,z)p(z)\,\mathrm{d}z.$$

The training objective for the VAE involves maximizing a variational lower bound of the expected log-likelihood, also known as the evidence lower bound (ELBO), given by

$$\mathsf{E}[\log p_\theta(w|s)] \geq \mathsf{E}\left[\log p_\theta(w|s,z) + \mathsf{KL}\big(q_\phi(z|w,s)\|p(z)\big)\right],$$

where $q_\phi(z|w,s)$ is a variational approximation of the (intractable) posterior $p_\theta(z|w,s) = p(z)p_\theta(w|s,z)/p_\theta(w|s)$.

The variational posterior $q_\phi(z|w,s)$ can be viewed as a probabilistic encoder that maps from the data features w to the latent representation z, while conditioned on s. We adopt the typical approach of parameterizing the encoder as a conditional Gaussian, $q_\phi(z|w,s) = \mathcal{N}(z;\mu_\phi(w,s),\Sigma_\phi(w,s))$, where the mean vector μ_ϕ and diagonal covariance matrix Σ_ϕ are parametric functions (i.e., realized by a neural network) of (w,s). With the latent prior set to the standard Gaussian distribution, i.e., $p(z) = \mathcal{N}(0,I)$, the Kullback-Liebler (KL) divergence term of the ELBO objective is tractable and differentiable [16]. Similarly, we also realize the decoder as a conditional Gaussian, i.e., $p_\theta(w|s,z) = \mathcal{N}\big(w;\hat{w}_\theta(z,s),\Sigma\big)$, where Σ is a trainable diagonal matrix, and $\hat{w}_\theta(z,s)$ is the decoder output. Thus, the first term of the ELBO is given by

$$\begin{aligned}\mathsf{E}[\log p_\theta(w|s,z)] = &-\frac{1}{2}\log(\mathsf{det}(\Sigma)) + c\\ &-\frac{1}{2}(w-\hat{w}_\theta(z,s))^\top\Sigma^{-1}(w-\hat{w}_\theta(z,s)),\end{aligned}$$

where $c = -\frac{d}{2}\log(2\pi)$ (with d as the dimensionality of w) is a constant that does not affect the optimization. The training objective for the VAE then becomes

$$\begin{aligned}\min_{\theta,\phi,\Sigma}\ &\frac{1}{2}\log(\mathsf{det}(\Sigma)) + \frac{1}{2}(w-\hat{w}_\theta(z,s))^\top\Sigma^{-1}(x-\hat{w}_\theta(z,s))\\ &-\mathsf{KL}\big(q_\phi(z|w,s)\|p(z)\big).\end{aligned} \tag{3}$$

In the second stage, we introduce an adversarial discriminator $D_\varphi : \mathcal{W}\times\mathcal{S}\to\mathbb{R}$ that aims to distinguish between actual data sample pairs (w,s) and synthetic pairs produced by the VAE generative model. This discriminator is applied in the manner of WGAN [20], with Lipschitz continuity enforced by spectral normalization [21], in order to improve the perceptual quality of the generative model. We also incorporate a regularization term that aims to constrain how much the fine-tuned decoder can vary, in order to maintain consistency with the frozen VAE encoder. The adversarial training loss of the second stage is given by

$$\min_\theta\max_\varphi \mathsf{E}\left[D_\varphi(w,s) - D_\varphi(\hat{w}_\theta(z,s),s)\right] + \lambda\|\theta-\theta_\star\|_2^2, \tag{4}$$

where $\lambda > 0$ controls the proximal regularization, $\theta_\star$ are decoder weights obtained by VAE training in the first phase with the latent representation z sampled from its prior, i.e., $z\sim p(z) = \mathcal{N}(0,I)$.

B. Information-theoretic identification of limiting scenarios

The simulated performance function (2), when combined with the building simulator (1) and the generative model in Section III, implicitly defines a black-box function $g:\mathcal{Z}\to\mathbb{R}$ that maps from the latent disturbance representation $z\in\mathcal{Z}$ to a potentially noisy performance outcome denoted by $J_z = g(z) + \mathcal{N}(0,\sigma^2)$. The set $\mathcal{Z}$ represents the set of allowable latent parameters, which we assume to be a compact subset of the set of all possible latent values concentrated where the VAE is accurate (within some standard deviation band around mean zero). The noise term can be used to capture other inputs that may vary within the simulator but are not modeled by the VAE.

We are interested in efficiently learning so-called "limiting scenarios" that lead to substantially different performance outcomes. We formulate this as a top-and-bottom-K estimation problem in which our goal is to simultaneously compute the top-K and bottom-K elements of a finite collection of latent values $Z\subseteq\mathcal{Z}$. The underlying idea is that we want to find scenarios that lead to a large spread in behavior (hence top and bottom), while also having the flexibility to observe the spread around these worst-case values when $K>1$. What makes this problem challenging is that we do not assume to have knowledge of the performance values $g_z := g(z)$ for most latent values $z \in Z$. If we had a large evaluation budget $L \geq |Z|$ and no noise in our evaluations, we could solve this problem with a simple algorithm $\mathcal{A}$ that (i) evaluates $g(z)$ for all $z \in Z$, (ii) sort the results $\{g_z\}_{z \in Z}$ in decreasing order, and (iii) return the first k and last k elements. However, we assume that our budget is much less than the number of points $L \ll |Z|$ since querying g_z requires one to run an expensive closed-loop building simulation. As such, we are motivated by the question: Can we identify the best L latent parameters $z_1, \ldots, z_L$ to sample such that we can accurately infer $\mathcal{K}^\star \subseteq Z$ (the optimal top-and-bottom-K elements)?

To answer this question, we look to the recently proposed Bayesian Algorithm Execution (BAX) framework [15]. BAX assumes uncertainty in the unknown function can be captured with some prior distribution $p(g)$, which for our purposes will be a Gaussian process (GP). The goal of BAX is to estimate the output of an algorithm $\mathcal{A}$, denoted by $O_{\mathcal{A}}(g) \in \mathcal{O}$, using some collection of noisy observations of the true function. Let $\mathcal{D}_\ell = \{(z_i, J_{z_i})\}_{i=1}^{\ell-1}$ denote the dataset of $\ell - 1$ function observations $J_{z_i} = g_{z_i} + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. The posterior distribution of the unknown function given this data is $p(g|\mathcal{D}_t)$. This distribution, when combined with algorithm $\mathcal{A}$ that returns output $O_{\mathcal{A}}$ (top-and-bottom-K elements), induces a posterior distribution over the algorithm output $p(O_{\mathcal{A}}|\mathcal{D}_\ell)$. Given some acquisition function $\alpha(z|\mathcal{D}_\ell)$ that reflects the potential benefit of querying g_z in the context of our knowledge of $O_{\mathcal{A}}$ given $\mathcal{D}_t$, the sequence of points $\{z_i\}_{i=1}^L$ can be designed using the following strategy

$$z_\ell \in \text{argmax}_{z \in \mathcal{Z}} \alpha(z|\mathcal{D}_\ell), \quad \mathcal{D}_{\ell+1} = \mathcal{D}_t \cup \{(z_\ell, J_{z_\ell})\}, \tag{5}$$

for all $\ell = 1, \ldots, L$.

Information-based BAX (InfoBAX) defines $\alpha(z|\mathcal{D}_\ell)$ in an information-theoretic manner. In particular, it uses the mutual information (MI) between the random output $O_{\mathcal{A}}$ and unrevealed observation J_z given the current dataset $\mathcal{D}_\ell$; MI represents the expected information gain (EIG) about $O_{\mathcal{A}}$ upon observing J_z, conditioned on $\mathcal{D}_\ell$ [22]. EIG can be expressed mathematically as

$$\begin{aligned} \mathsf{EIG}_\ell(z) &= \mathsf{H}[O_{\mathcal{A}}|\mathcal{D}_\ell] - \mathbb{E}_{J_z|\mathcal{D}_\ell}[\mathsf{H}[O_{\mathcal{A}}|\mathcal{D}_\ell \cup \{(z, J_z)\}]], \\ &= \mathsf{H}[J_z|\mathcal{D}_\ell] - \mathbb{E}_{O_{\mathcal{A}}|\mathcal{D}_\ell}[\mathsf{H}[J_z|\mathcal{D}_\ell, O_{\mathcal{A}}]], \end{aligned} \tag{6}$$

where $\mathsf{H}[A] = \mathbb{E}_{p(A)}[-\log p(A)]$ denotes the differentiable entropy for any continuous random variable A. The second line above can be derived from the symmetry property of MI, often called the predictive entropy form based on previous work [23]. The main advantage of (6) is that the first term is the entropy of the posterior predictive distribution $p(J_z|\mathcal{D}_\ell)$, which has a closed-form expression for GP models

$$\mathsf{H}[J_z|\mathcal{D}_\ell] = 0.5 \log(2\pi e(v_\ell(z) + \sigma^2)), \tag{7}$$

where $v_\ell(z)$ denotes the predictive variance of the GP given $\mathcal{D}_\ell$. The second term, however, is difficult to compute since we do not have a closed-form expression for $p(J_z|\mathcal{D}_\ell, O_{\mathcal{A}})$. Thus, we rely on the method developed in [15, Section 3.3] to evaluate an approximate form of $\mathsf{EIG}_\ell(z)$ and use $\alpha(z|\mathcal{D}_\ell) = \mathsf{EIG}_\ell(z)$ in (5), along with a heuristic maximization procedure, to generate samples over our finite budget.

IV. Case Study: Generative Modeling and Scenario Learning (With Real Data) for Building Performance Simulations

A. Building Simulator

We developed a high-fidelity building emulator using the Modelica Buildings Library 7.0.0[†]. The emulators are based on the ASHRAE BESTEST model, which are benchmark models for building energy simulation with open-sourced information such as dimensions and constructions [24]. The building represents a single zone with a window on the south wall and a constant infiltration mass flow rate. There are two variations of construction, the light weighted Case 600 and the heavy weighted Case 900. The exterior walls and roof of Case 600 and 900 are respectively plaster board with fiberglass insulation and concrete block with foam insulation. The floor of case 600 is timber construction, and the floor of Case 900 is concrete slab. The external disturbances are characterized by the typical meteorological year data for San Francisco, USA.

The emulators include a simple fan coil unit (FCU) to condition the room that is regulated by a proportional-integral (PI) dual-setpoint controller to maintain the room temperature within the heating and cooling setpoints. When the FCU is activated, the supply fan runs at a constant speed to circulate the air through the heating and cooling coils. The heating and cooling setpoints are converted to the supply air temperature setpoint by the PI controller, and the coils are activated to reach the setpoint. The conditioned air is then supplied to the room, where the air is assumed to be well-mixed. As this study focuses on the control performance, the energy impact of FCU is simplified. The electric heating coil has a constant efficiency of 0.9, and the cooling coil operates at a constant coefficient of performance (COP) of 3.0.

The internal disturbances are defined to represent a small office, where the operating hours are assumed to be from 8 AM to 6 PM. With six workstations per 92.9 m^2 and a zone floor area of 48 m^2, three people are assumed to be in the room. The sensible heat gain density of office equipment is set to be 5 W/m^2 with the radiative heat gain comprising 30% of the total, which is a typical light office load. Similarly, the lighting heat gain is assumed to be 10 W/m^2 with a 50% radiative fraction. The occupant-related activity generates both sensible and latent gains, involved in room heat and moisture mass balance. They are assumed to be 73 W/person sensible with a 60% radiative fraction, and 45 W/person latent gain, corresponding to moderate office work. These heat gains are active during the occupied period and zero during unoccupied periods. The heating and cooling occupied and unoccupied setpoint temperatures are (18°C, 20°C) and (14°C, 28°C).

[†]https://simulationresearch.lbl.gov/modelica/

B. Data Collection

For the purpose of training generative models of internal disturbances, we use real experimental data collected from SUSTIE, which is a next-generation commercial office building located in Japan. The name SUSTIE combines the words "Sustainability" and "Energy" and the building is designed to research and demonstrate energy savings and workers' health and comfort. The four-story SUSTIE building has a total floor area of approximately 6456 m^2 which includes nine office spaces (experimental rooms) regularly used by around 260 office workers, an open-feel atrium area, a cafeteria and a gym. SUSTIE's building management system collects electrical energy consumption, meteorological, indoor environment conditions, occupancy, and equipment operational data to analyze and control energy consumption and comfort during building operations. The electrical energy consumption is measured separately for different types of equipment (air-conditioning, ventilation, lighting, hot water supply and elevators) and for each room. The occupancy, i.e., number of people in each room is counted by the access control system using card readers installed in each area. A dataset collected at SUSTIE over consecutive 20 months from January 2021 to August 2022 is used in this work for generative modeling.

Noting that time constants for many internal disturbances of a building are typically of the order of several minutes, we down-sampled the dataset with a sampling rate of 15 minutes. Since multi-zone SUSTIE is much more complex than a single-zone building simulator described in the previous section, we select a proportionate subset of signals from the SUSTIE dataset for our modeling efforts. In particular, we consider a set of 12 signals which include power consumption by HVAC equipment, ventilation, lighting systems and the occupancy in three different office spaces located on the same floor of SUSTIE. While this reduction in the size of dataset also makes the training process much more tractable, we emphasize that the scaled dataset still maintains characteristic day-to-day usage trends of a real commercial office building.

C. RAFT-VG and InfoBAX implementation

In order to train the conditional RAFT-VG, we first scale the data by their median orders of magnitude i.e. temperature signals in °C are scaled down by 10, whereas power signals in W are scaled down by 1000 or 100 depending on their median orders of magnitude. Note that all the time-series to be generated in this work are non-negative. The data tensor is of size $600 \times 96 \times 12$, since the dataset is for 4 signals in 3 rooms collected every 15 min for 600 days. Two conditional inputs are considered: (i) the zone number, as the data is for 3 distinct zones in the SUSTIE building, and (ii) whether the scenario to be generated is a workday or a holiday, which greatly affects occupancy, and therefore, usage. For ease of conditional RAFT-VG training, the data tensor is reshaped to $1800 \times 96 \times 4$ so each sample is a scenario agnostic of the room, and the conditional inputs are of size 1800×2.

For the training itself, we split (randomly) the data into a training set with 1500 samples, and a hold-out validation set with the 300 remaining samples. The training loss is computed with the two losses (3) and (4), and for the validation loss, we compute the symmetric Jensen-Shannon[‡] divergence (JSD) between the validation data distribution and the decoder-generated distribution. The JSD is computed efficiently by approximating both by their mean and variance, and assuming independence and no cross-correlation. Network weights are saved when the total JSD over the validation data batches is lower than the JSD computed in all previous training iterations. The batch size is fixed at $n_{\mathsf{batch}} = 128$. We set $\beta_{\mathsf{VAE}} = 10^{-4}$ and the total number of VAE training iterations is 9000 with an Adam solver with learning rate 10^{-4} without scheduling. We allot 1000 iterations for GAN-based adversarial fine-tuning with learning rate 10^{-6} and $\lambda = 10^5$ to discourage finding solutions too far from the stage-1 decoder weights. The discriminator is optimized with an RMSProp solver and the decoder is updated with an Adam solver with momentum factors $(0.5, 0.999)$ instead of the default $(0.9, 0.999)$. For each outer loop decoder update in (4), the discriminator is updated 2 times, which is a common trick that helps Wasserstein-GAN training [20].

All three components of the RAFT-VG are conditioned, that is, conditioning inputs are passed through an embedding layer using `torch.nn.Embedding` with an embedding dimension of 4. Note that these layers are also learned during training. For the conditional encoder, the input 96×4 tensor is flattened and a batch has size $n_{\mathsf{batch}} \times 384$. This is passed through 5 encoder layers activated by `LeakyReLU` functions, with hidden dimension $(256, 256, 256, 128, 64)$ and ending with a latent distribution in $n_z = 8$ dimensions. Note that such a low latent dimension has been chosen specifically because we will search this latent domain for representative scenarios. The encoder has two outputs: the mean and log-variance of the encoding. These are passed through a standard reparameterization operation and sampled using an isotropic standard Gaussian distribution which is fed to a decoder.

Since the decoder is arguably the most important component of our architecture, as this is the component that generates the scenarios, we investigated various architectures and activation functions for its design. For instance, because our signals are always non-negative, we ensure that the final layer is passed through a non-negative activation function. However, we discovered that a ReLU function in the output does not work well in practice, because it leads to sudden drops to zero in the generated scenarios. To counter this, our final layer is passed through a smooth softmax function. We also discovered that using ELU functions as activation functions in the intermediate layers of the decoder improves the smoothness of the generated scenarios, as opposed to

[‡]Assuming both n-dimensional distributions are Gaussian, with means μ_P and μ_Q, and covariance matrices Σ_P and Σ_Q, the Jensen-Shannon Divergence between the Gaussian distributions P and Q can be expressed as: $\mathsf{JSD}(P \parallel Q) = \frac{1}{2}\mathsf{KL}(P \parallel M) + \frac{1}{2}\mathsf{KL}(Q \parallel M)$ where KL is the Kullback-Leibler divergence, which has the closed-form

$$2\mathsf{KL}(P \,\|\, Q) = \mathsf{Tr}(\Sigma_Q^{-1}\Sigma_P) + \|\mu_Q - \mu_P\|^2_{\Sigma_Q^{-1}} - n + \log\left(\frac{\det(\Sigma_Q)}{\det(\Sigma_P)}\right)$$

with M as the average distribution $\frac{1}{2}(P+Q)$.

ReLUs or LeakyReLUs. The decoder has 5 hidden layers, with sizes $(64, 64, 128, 128, 256)$ and an output of size 384 to match the encoder's input. The input to the decoder is an 8-dimensional latent along with the $4 + 4 = 8$ embedded conditional inputs.

The conditional WGAN has 4 layers, each of which are passed through the spectral norm (`torch.nn.utils.spectral_norm`) and subsequently activated by `LeakyReLU(0.01)` functions followed by `Dropout(0.1)` for regularization during stage-2 training of the RAFT-VG. The 384-size input to the discriminator is conditioned with the 8-dimensional conditional embedding, and the output is a scalar. The final layer of the WGAN does not have dropout or spectral norm regularization, and is a linear layer without activation.

To implement InfoBAX, we build upon the code from [25]. As mentioned previously, we select the algorithm $\mathcal{A}$ to solve the top-bottom-K problem where $K = 5$. The discrete set of latent parameters Z was generated by randomly sampling within a hypercube of $[-5, 5]^8$ in the 8-dimensional latent space. We generate $|Z| = 5000$ samples, while having a total budget of $T = 300$; that is, we have the ability to query less than 10% of the space. To construct the GP prior, we use a squared exponential (also known as radial basis function) kernel with automatic relevance determination (ARD), i.e., each dimension is individually scaled using a lengthscale hyperparameter. The lengthscale, output scale, and noise variance hyperparameters were estimated from the initial 100 data points using the standard maximum likelihood estimation (MLE) procedure outlined in [26]. These hyperparameters were then fixed throughout the rest of sequential sampling procedure for simplicity.

D. Modelica implementation

There are several aspects to consider when integrating Generative Models into building simulations, i.e. 1) Generative Model Implementation, 2) Generative Model integration with building models, and 3) simulation workflow automation. The goal of the software integration approach used here was to minimize all dependencies (only for simulation purposes) on external software tools other than the C compiler and the Modelica tool, in this case Dymola [27], [28], which requires a C compiler itself. Using the standardized external function feature from the Modelica language implies that if the integration of Generative Models with the simulation model is done solely using C, the only dependency would be that of the C compiler, which is anyway required by Dymola. Meanwhile, simulation workflow automation can be achieved by interfacing the simulator executable, which contains *both* the Generative Models and the building models, with a suitable scripting tool supported by Dymola. To achieve this goal, the three aspects considered at the beginning of this section were addressed in the software integration and implementation shown in Fig. 2.

As Fig. 2 shows, the Generative Model architecture is re-implemented in C, where different C functions provide inputs to the Generative Model and request its output to populate the lookup table. The function itself is defined through the `external` function that provides the Generative Model output, i.e., the **`timeSeries`** output that will pass its data to `lookUpTableValues` in `CombiTimeTable`. The `CombiTimeTable` also performs sample-and-hold and periodic extrapolation. Finally, the outputs of the `CombiTimeTable` are mapped to `RealOutput` and `IntegerOutput` interfaces that are linked to the simulation model as shown in the right hand side of Fig. 2, the top left part of the area enclosed in an orange square. This model is then used within the full system model, as shown in the bottom right of Fig. 2. Finally, to facilitate the automation of the simulation workflow, that is, to execute multiple simulations automatically, the Python Interface for Dymola [28], an API for executing Dymola commands using a Python program, is used.

E. Results and Discussion

1) Results on Scenario Generation: We begin by evaluating the performance of the RAFT-VG architecture for scenario generation. To this end, we present occupancy and heat loads produced by lighting, ventilation, and office equipment in a zone for a 24-hour period, which is the time of interest T. This is shown in Fig. 3 for two latent distributions: $\mathcal{N}(0, I)$ and $\mathcal{N}(0, 5I)$. By raising the standard deviation of the latent distribution, we artificially encourage the sampling of scenarios that are far from the average behaviour, which is clear from the figure. The left column of subplots show the generated signals from the tighter latent distribution $\mathcal{N}(0, I)$ for the workday conditioning input in red, and the holiday conditioning input in blue. The nominal scenario is the average scenario taken over the entire SUSTIE dataset, and is represented in black. The mean of the true data for both conditionals are overlaid with 40 scenarios that have been sampled and decoded by the trained RAFT-VG decoder model. In fact, sampling from $\mathcal{N}(0, I)$ is so tight around the mean that the true data mean and the scenarios are almost indistinguishable. Conversely, for the looser distribution $\mathcal{N}(0, 5I)$, one can easily note from the right column of subplots that the scenarios generated are shifted significantly from the true data mean (dashed blue and dashed red lines), and result in scenarios that are different enough to induce some interesting closed-loop behavior. Note also that despite the larger $\sigma = 5$, the two conditional distributions (workday vs. holiday) do not intersect, i.e. the blue lines do not intersect with the red, and vice versa. We note that this is true even with a larger number of sampled scenarios, motivating us to use the truncated $\mathcal{N}(0, 5I)$ as the sampling distribution for InfoBAX.

2) Results on Latent Sampling with InfoBAX: To evaluate the performance of InfoBAX, we use the so-called Jaccard distance between the estimated top-and-bottom-K set $\hat{\mathcal{K}}_\ell$ given $\mathcal{D}_\ell$ and the true optimal set $\mathcal{K}^\star$, which is given by

$$\text{Jaccard distance}(\hat{\mathcal{K}}_\ell, \mathcal{K}^*) = 1 - \tfrac{|\hat{\mathcal{K}}_\ell \cap \mathcal{K}^*|}{|\hat{\mathcal{K}}_\ell \cup \mathcal{K}^*|}. \tag{8}$$

Since $\hat{\mathcal{K}}_\ell$ is a random variable corresponding to $p(O_\mathcal{A}|\mathcal{D}_\ell)$, we report the expected Jaccard distance at every BAX iter-

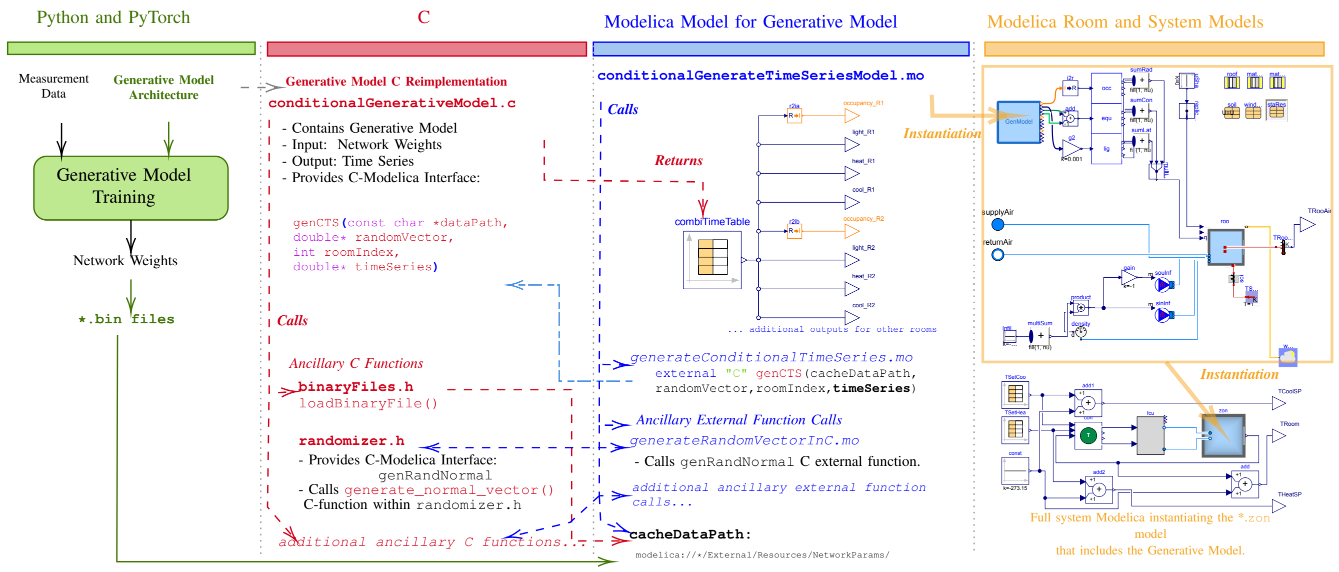
Python and PyTorch
C
Modelica Model for Generative Model
Modelica Room and System Models
Measurement Data
Generative Model Architecture
Generative Model Training
Network Weights
*.bin files
Generative Model C Reimplementation
conditionalGenerativeModel.c
- Contains Generative Model
- Input: Network Weights
- Output: Time Series
- Provides C-Modelica Interface:
genCTS(const char *dataPath, double* randomVector, int roomIndex, double* timeSeries)
Calls
Ancillary C Functions
binaryFiles.h
loadBinaryFile()
randomizer.h
- Provides C-Modelica Interface: genRandNormal
- Calls generate_normal_vector() C-function within randomizer.h
additional ancillary C functions...
conditionalGenerateTimeSeriesModel.mo
Calls
Returns
combiTimeTable
occupancy_R1
light_R1
heat_R1
cool_R1
occupancy_R2
light_R2
heat_R2
cool_R2
... additional outputs for other rooms
generateConditionalTimeSeries.mo
external "C" genCTS(cacheDataPath, randomVector,roomIndex,timeSeries)
Ancillary External Function Calls
generateRandomVectorInC.mo
- Calls genRandNormal C external function.
additional ancillary external function calls...
cacheDataPath:
modelica://*/External/Resources/NetworkParams/
Instantiation
GenModel
supplyAir
returnAir
TRooAir
Instantiation
TSetCoo
TSetHea
const
fcu
zon
TCoolSP
TRoom
THeatSP
Full system Modelica instantiating the *.zon model that includes the Generative Model.

Fig. 2. Integration of RAFT-VG with Modelica.

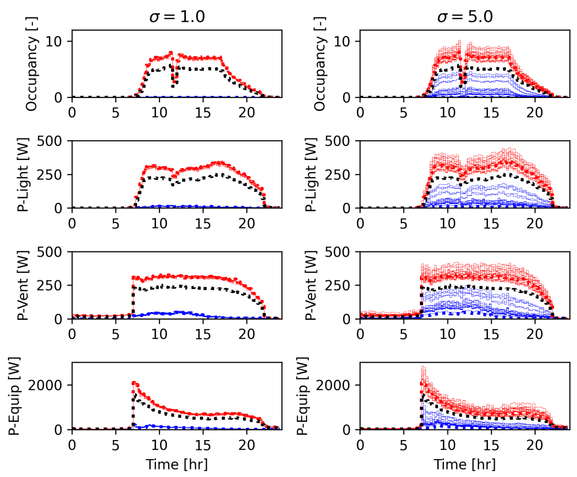

Fig. 3. Scenarios (occupancy and internal head load disturbances) for workday (red) and holiday (blue) for a single zone, generated by the RAFT-VG by sampling the latent distribution $\mathcal{N}(0, \sigma I)$ for two σ values. The nominal scenario is shown in black.

ation ℓ by averaging over posterior samples generated from $p(O_{\mathcal{A}}|\mathcal{D}_{\ell})$. Since the results depend on the random initial dataset, we further report confidence bounds around the Jaccard distance based on 10 complete InfoBAX replicates. The Jaccard distance for InfoBAX and random search over 300 total iterations is shown in Figure 4. We see a significant improvement in the Jaccard distance using InfoBAX with the median over the 10 replicates perfectly identifying $\mathcal{K}^{\star}$ (distance of 0) with less than 300 iterations whereas random search only arrives at a distance of ~ 0.7 in 300 iterations.

3) Results on simulations with limiting scenarios: With confidence that the InfoBAX algorithm is selecting 'good' limiting scenarios in a Jaccard-distance sense, we assess the performance of the closed-loop building simulation model driven by these limiting scenarios. We also compare against the nominal scenario (the average of the true data). The results of these simulations is shown in Fig. 5. The bottom

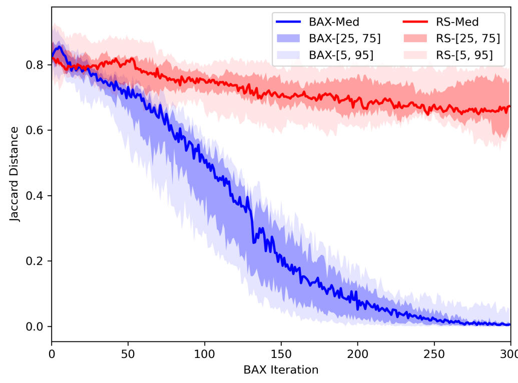

Fig. 4. Comparison of Jaccard distance between solutions obtained by InfoBAX versus random sampling in the latent space.

three subplots are the scenarios computed by InfoBAX, with the bottom-K in red, the nominal in black, and the top-K in blue. To summarize, the bottom-K samples all have in common a high occupancy throughout the workday, with significant use of lighting as well as equipment throughout the day, tapering off as the working hours come to a close. The top-K scenarios are the opposite, where few people use this zone, and even those who do are frugal in their use of lights and equipment. The ambient temperature and solar irradiance (direct horizontal, direct normal, global horizontal) are also shown as these are the external disturbances d_t^{ext} that strongly affect the zone's thermal profile. Note that the ambient temperature and solar radiation is fixed, and for a winter day with moderate sunshine and little cloud cover.

The top row of subplots are the zone temperature profiles throughout the 24-hour period considered, under PI regulation with free-floating allowed in the green bands; i.e. the PI controllers act only if the zone temperature exceeds the upper limit of the green band or falls below the lower limit. Two building models are considered, as described in Section IV-A. Building Model-1 has high air infiltration (0.41 ACH) and standard insulation (material heat capacity of 840 J/kgK and thermal conductivity of 0.04 W/mK), and Model-2 has low air infiltration (0.041 ACH) and improved insulation in the roof and walls (84 J/kgK, 0.01 W/mK). The zone temperature corresponding to the bottom-K, nominal, and top-K follow the same color theme as in the scenario subplots.

These simulations demonstrate the counter-intuitive effect that reducing the infiltration and increasing the insulation quality has on the building performance. As the occupant and solar heat gains are higher than the effect of the cold weather on the envelope, case (i) requires cooling from the HVAC system. This system hits its performance limit about halfway through the day, causing the room temperature to rise somewhat above the constraint from approximately noon to 6pm. In comparison, the reduced infiltration and improved insulation effectively mitigates much of the cooling provided by the low ambient temperatures, making the building prone to overheating. As a result, the cooling coil hits its maximum capacity limit much earlier in the day, so that the system is unable to manage the heat gains and the room temperature rises well above the desired setpoint for a large portion of the day. These undesired thermal behaviors caused by "improving" the building envelope are commonly observed, and this information would be valuable to properly size the HVAC systems to avoid such phenomena.

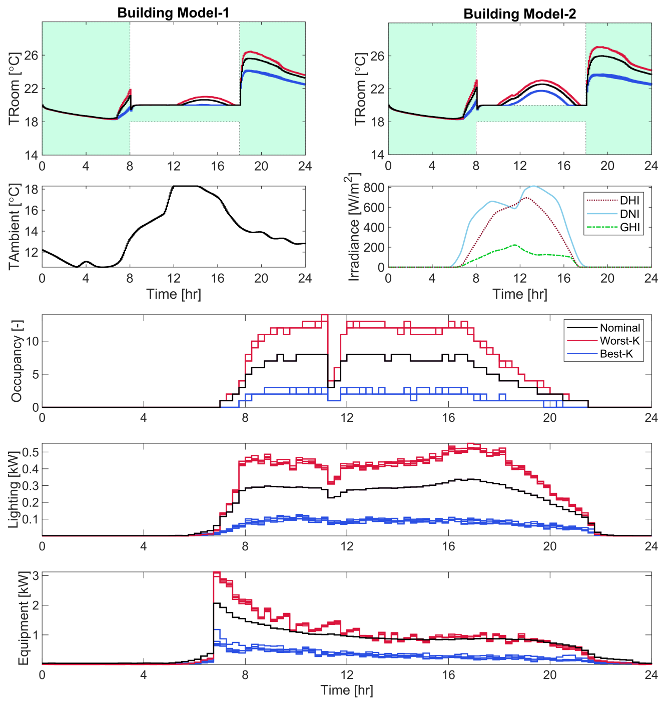

Fig. 5. Best-K (blue), worst-K (red), and nominal (black) closed-loop control performances in two separate building simulation models ($K = 5$).

References

[1] M. Wetter, W. Zuo, T. S. Nouidui *et al.*, "Modelica buildings library," *Journal of Building Performance Simulation*, vol. 7, no. 4, pp. 253–270, 2014.

[2] A. Chakrabarty, E. Maddalena, H. Qiao *et al.*, "Scalable bayesian optimization for model calibration: Case study on coupled building and HVAC dynamics," *Energy and Buildings*, vol. 253, p. 111460, 2021.

[3] S. Zhan, G. Wichern, C. Laughman *et al.*, "Calibrating building simulation models using multi-source datasets and meta-learned bayesian optimization," *Energy and Buildings*, vol. 270, p. 112278, 2022.

[4] M. Wetter, P. Ehrlich, A. Gautier *et al.*, "OpenBuildingControl: Digitizing the control delivery from building energy modeling to specification, implementation and formal verification," *Energy*, vol. 238, p. 121501, 2022.

[5] P. Stoffel, L. Maier, A. Kümpel *et al.*, "Evaluation of advanced control strategies for building energy systems," *Energy and Buildings*, vol. 280, p. 112709, 2023.

[6] A. Mirakhorli and B. Dong, "Occupancy behavior based model predictive control for building indoor climate—A critical review," *Energy and Buildings*, vol. 129, pp. 499–513, 2016.

[7] B. Dong, Y. Liu, W. Mu *et al.*, "A global building occupant behavior database," *Scientific data*, vol. 9, no. 1, p. 369, 2022.

[8] Y. Ye, M. Strong, Y. Lou *et al.*, "Evaluating performance of different generative adversarial networks for large-scale building power demand prediction," *Energy and Buildings*, vol. 269, p. 112247, 2022.

[9] H. P. Das, R. Tran, J. Singh *et al.*, "Conditional synthetic data generation for robust machine learning applications with limited pandemic data," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 11, 2022, pp. 11 792–11 800.

[10] Z. Chen and C. Jiang, "Building occupancy modeling using generative adversarial network," *Energy and Buildings*, vol. 174, pp. 372–379, 2018.

[11] C. Fan, M. Chen, R. Tang *et al.*, "A novel deep generative modeling-based data augmentation strategy for improving short-term building energy predictions," in *Building Simulation*, vol. 15. Tsinghua University Press, 2022, pp. 197–211.

[12] M. Fochesato, F. Khayatian, D. F. Lima *et al.*, "On the use of conditional timegan to enhance the robustness of a reinforcement learning agent in the building domain," in *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, 2022, pp. 208–217.

[13] F. Khayatian, Z. Nagy, and A. Bollinger, "Using generative adversarial networks to evaluate robustness of reinforcement learning agents against uncertainties," *Energy and Buildings*, vol. 251, p. 111334, 2021.

[14] A. Salatiello, Y. Wang, G. Wichern *et al.*, "Synthesizing building operation data with generative models: VAEs, GANs, or something in between?" in *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 125–133.

[15] W. Neiswanger, K. A. Wang, and S. Ermon, "Bayesian algorithm execution: Estimating computable properties of black-box functions using mutual information," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8005–8015.

[16] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[17] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.

[18] I. Goodfellow, J. Pouget-Abadie, M. Mirza *et al.*, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. [Online]. Available: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[19] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014. [Online]. Available: http://arxiv.org/abs/1411.1784

[20] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[21] T. Miyato, T. Kataoka, M. Koyama *et al.*, "Spectral normalization for generative adversarial networks," 2018.

[22] D. V. Lindley, "On a measure of the information provided by an experiment," *The Annals of Mathematical Statistics*, vol. 27, no. 4, pp. 986–1005, 1956.

[23] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[24] ASHRAE, "140: Standard method of test for the evaluation of building energy analysis computer program," ASHRAE, Tech. Rep., 2007.

[25] W. Neiswanger, K. Wang, and S. E., "Bayesian Algorithm Execution (BAX)," https://github.com/willieneis/bayesian-algorithm-execution, since 2021.

[26] C. K. Williams and C. E. Rasmussen, *Gaussian Processes For Machine Learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[27] D. Bruck, H. Elmqvist, H. Olsson *et al.*, "Dymola for Multi-Engineering Modeling and Simulation," in *2nd International Modelica Conference*, 2002, pp. 55–1 — 55–8. [Online]. Available: https://modelica.org/events/Conference2002/papers/p07_Brueck.pdf

[28] Dassault Systemes AB, *Dymola — Dynamic Modeling Laboratory — Full User Manual*, Lund, Sweden, September 2023.