SplitVAEs: Decentralized scenario generation from siloed data for stochastic optimization problems

H M Mohaimanul Islam[1], Huynh Q. N. Vo[1], Paritosh Ramanan[1]

[1]School of Industrial Engineering and Management
Oklahoma State University, Stillwater, OK 74078 USA
Email: {h_m_mohaimanul.islam, lucius.vo, paritosh.ramanan}@okstate.edu

Abstract—**Stochastic optimization problems in large-scale multi-stakeholder networked systems (e.g., power grids and supply chains) rely on data-driven scenarios to encapsulate complex spatiotemporal interdependencies. However, centralized aggregation of stakeholder data is challenging due to the existence of data silos resulting from computational and logistical bottlenecks. In this paper, we present SplitVAE, a decentralized scenario generation framework that leverages variational autoencoders to generate high-quality scenarios without moving stakeholder data. With the help of experiments on distributed memory systems, we demonstrate the broad applicability of SplitVAEs in a variety of domain areas that are dominated by a large number of stakeholders. Furthermore, these experiments indicate that SplitVAEs can learn spatial and temporal interdependencies in large-scale networks to generate scenarios that match the joint historical distribution of stakeholder data in a decentralized manner. Lastly, the experiments demonstrate that SplitVAEs deliver robust performance compared to centralized, state-of-the-art benchmark methods while significantly reducing data transmission costs, leading to a scalable, privacy-enhancing alternative to scenario generation.**

Index Terms—**scenario generation, stochastic optimization, decentralized learning, variational autoencoders.**

I. INTRODUCTION

Stochastic optimization (SO) problems are essential components in large-scale, data-driven planning and decision frameworks for multi-stakeholder-led networked systems including but not limited to energy [1], supply chain [2], spare parts management [3], healthcare [4] and logistics [5]. Good quality solutions to SO problems fundamentally require high-fidelity data-driven scenarios that effectively encapsulate spatial and temporal interdependencies from myriad sources [6]. Consequently, the generation of high-fidelity scenarios is heavily contingent on the availability of high-quality datasets that capture these interdependencies from heterogeneous data sources. Conventional scenario generation methods necessitate centralized data aggregation to capture interdependencies to drive stochastic planning decisions. These centralized aggregation schemes, however, face several computational, and logistical challenges [7], [8] due to the presence of data silos across multiple stakeholders of a networked system. Therefore, in this study, we present SplitVAE, a novel decentralized

scenario generation framework leverages split-learning to train variational autoencoders (VAEs) to eliminate the demand for centralized data aggregation. SplitVAEs are compatible with pre-existing data silos, ensure scalable generation of high-fidelity scenarios and significantly reduce data transfer costs.

Conventionally, SO solution frameworks rely on the acquisition of local spatiotemporally correlated datasets from individual stakeholders. The mechanism of data aggregation is typically carried out by a control center or planning organization as represented in Figure 1. Such coordinating entities are typically found in multi-stakeholder networked systems, for instance, Independent System Operators (ISOs) in power systems and control towers in the supply chain. The aggregated dataset is then used to generate high-fidelity scenarios that can be used to solve an SO problem in a centralized fashion at the control center level. The solutions to these SO problems represent data-driven planning decisions that can be in turn used to determine an operational schedule for each stakeholder subject to constraints like reliability, safety and demand satisfaction. However, the entire data driven planning mechanism is challenged in cases where data silos are present across various stakeholders.

$$\min_{x} c^T x + E_{s \in \mathcal{S}}[\min_{y} q_s^T y_s]$$
$$\text{s.t.} \quad Ax = b$$
$$W_s y_s = h_s - T_s x, \forall s \in S$$
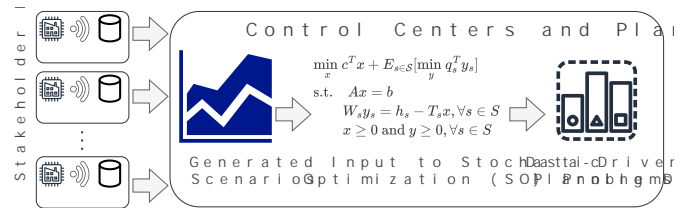$$x \geq 0 \text{ and } y \geq 0, \forall s \in S$$

Fig. 1: Scenario generation for stochastic optimization.

Therefore, the goal of SplitVAEs is to enable planning organizations and control centers to generate generate stochastic decisions without incurring the burden of data processing and aggregation from multiple system stakeholders. Such planning organizations typically have regulatory oversight on the operational aspects of these systems necessitating SO problems to be solved in a high-frequency fashion. Thus, our proposed SplitVAE framework relies on distributed memory based aggregation primitives to conduct training without the need to move local data. As a result, SplitVAEs by their very design are fully capable of functioning in the presence

of geographically distributed, decentralized data silos as is typically found in multi-stakeholder infrastructure systems.

As a result, SplitVAEs can potentially be extended to operate in hybrid edge-cloud environments based on containerized applications enabling significant reductions in data transmission. Additionally, it is capable of succinctly capturing spatiotemporal interdependencies in a decentralized manner despite the inherent non-linearities in the underlying data among multiple stakeholder subsystems while maintaining high scalability. SplitVAEs complement existing stochastic optimization frameworks by supplying high-fidelity scenarios generated from siloed datasets. SplitVAEs are not meant to be a replacement of stochastic optimization solution methodologies. Instead, they help generate spatiotemporally interdependent scenarios from siloed or geographically dispersed datasets, which can then be used as input for solving stochastic optimization problems. As a result, we can concisely summarize the contributions of our study as follows:

- We design a computationally efficient learning paradigm that combines edge-based autoencoders with a server driven variational autoencoder to jointly learn spatiotemporal interdependencies of siloed data.
- We develop a scalable algorithmic framework that decomposes global backpropagation steps into edge and server-based sub-problems, enabling the bi-directional flow of learning insights, eliminating the movement of data.
- We propose a distributed memory-based computational paradigm that can be seamlessly adopted to enable a scalable, real-world implementation of the SplitVAEs framework to generate scenarios for SO problems.
- We demonstrate the ability of SplitVAEs to handle heterogeneous datasets across diverse areas by comparing the generated scenarios with established benchmark methods.

The remaining parts of this study are organized in the following manner. Section II provides a literature review of prior work. Sections III and IV describe the architecture and the algorithmic design of our proposed framework, respectively. Section V provides a summary of benchmark methods as well as statistical evaluation metrics. Section VI analyzes the performance of the SplitVAEs framework across different application domains and diverse computational settings. Section VII serves as the concluding section of this study.

II. RELATED WORKS

There have been several scenario generation mechanisms that have been proposed including auto-regressive integrated moving average (ARIMA) methods [9], [10], Gaussian copula-based approaches [11], [12] and deep generative adversarial networks (GANs) [7], [13]. A majority of these studies, however, dealt with data centralization leading to computational and data privacy concerns [7]. From a computational standpoint, centralized methods such as ARIMA and copula-based approaches rely on the construction and factorization of covariance matrices [12], [14] aided by the accumulation of stakeholder data. Meanwhile, centralized GAN-based methods suffer from high instability and challenges in convergence [15]

during model training. As a result, scalability remains an issue with these methods, especially for large-scale networks and systems consisting of myriad sources of uncertainty.

Further, centralized methods depend on nonlinear modeling of relationships between predictor feature variables and the generated scenarios that are connected in space and time. The authors in [14] used wind speed data to construct scenarios of wind power generation via the power curve of wind turbines. The work in [16] leveraged the nonlinear relationship between temperature and power consumption using epi-spline functions to generate scenarios. These approaches assume homogenous sources of data. When integrating heterogeneous sources of data such as wind, PV farms, and alternative fuel sources like hydrogen, these approaches might become infeasible.

On the other hand, VAEs [17] serve as a potential alternative to methods that rely on non-linear modeling requirements. VAEs can help synthesize new, previously unseen scenarios by capturing the distribution of more concise, low-dimensional embeddings in a latent space. By sampling from the learned latent space distributions, VAEs can be used to generate scenarios that are spatially and temporally interdependent. Unlike GANs however, the VAEs provide a scalable, and decomposable implementation paradigm which is suited for data silos with heterogenous features as well.

Additionally, challenges with respect to siloed data arise in several application domains. For example, visibility across stakeholder subsystems is a major hindrance in supply chain and logistics owing to heterogeneity and incompatibility of local data storage systems [18]. Similarly, in the medical field, data privacy is the primary reason for the existence of data silos [19]. Privacy obstacles to scenario generation could potentially stem from the sensitivity of underlying datasets from the commercial standpoint [20]. Such data sharing challenges are especially prevalent in the case of variable renewable energy (VRE) stakeholders [8], [21]. Overall, conventional methods of scenario generation face several limitations in terms of their applicability to heterogeneous data and requirements for privacy preservation from diverse stakeholders.

Split learning [22]–[24] presents a more generalized approach to handling siloed data sources in a decentralized, privacy-friendly fashion. Instead of relying on horizontal partitioning of data, split learning relies on model partitioning schemes that yield low-dimensional embeddings from heterogeneous data. Split learning paradigms leverage these embeddings to learn the interdependencies of the features contained in the siloed heterogeneous datasets without the need to move local data. As a result, split learning methods can potentially help resolve the data heterogeneity and data silo challenges that are inherent in scenario generation methods. Thus, our approach combines the split learning paradigm with the VAE to yield a computationally efficient, privacy-enhancing framework capable of delivering high-quality scenarios from decentralized and siloed datasets.
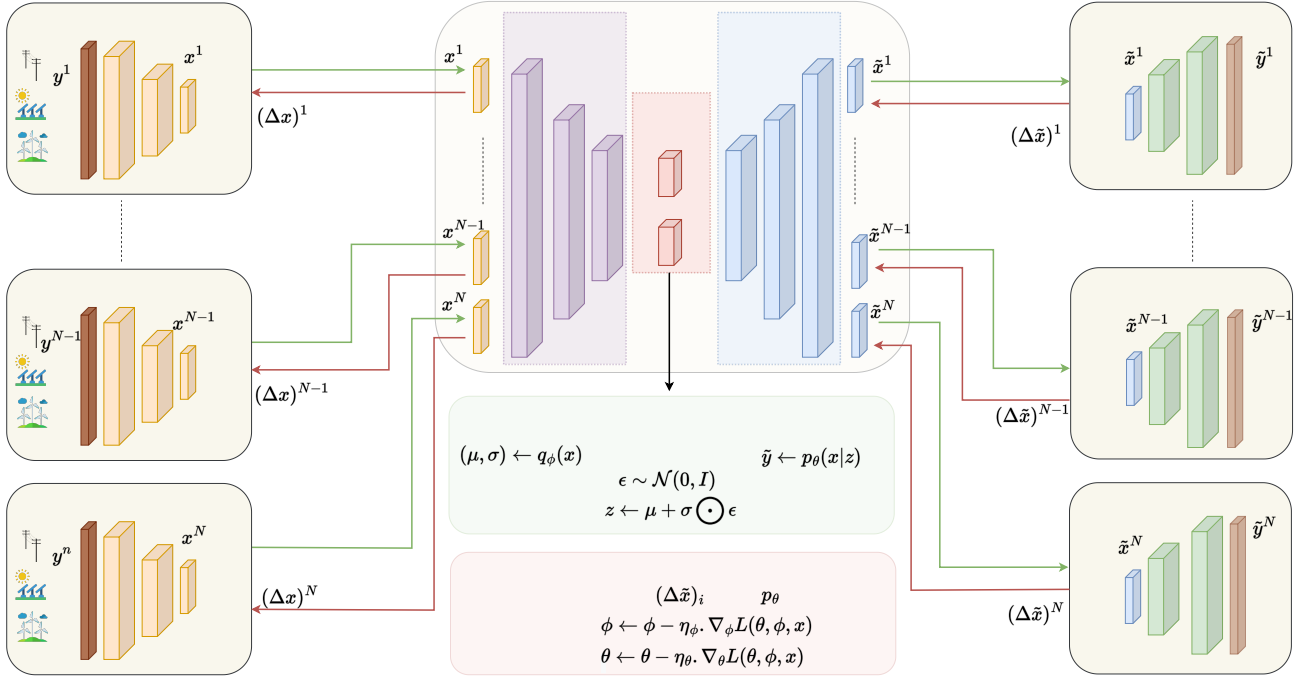
Fig. 2: The learning paradigm of SplitVAEs.

III. SplitVAEs Framework Design

We consider the SplitVAEs framework from the perspective of edge and server-level agents. The edge level represents heterogeneous system stakeholders that govern individual data silos comprising time series datasets. In our framework, the server entity houses the VAE and represents a planning organization, coordination agency, or a control center, while the edge entities consist of an autoencoder (AE) that is jointly trained in conjunction with the VAE at the server level. We begin our discussion by describing the conventional, centralized VAE (Central-VAE) formulation.

A. Centralized variational autoencoder (VAE) formulation

The Central-VAE model consumes data provided by edge-level AEs, denoted by $x \in \mathbb{R}^d$, to yield latent space embeddings, denoted by $z \in \mathbb{R}^s$, with the help of a probabilistic encoder, denoted by $q_\phi^{\mathrm{cvae}}(z|x)$. Specifically, $q_\phi^{\mathrm{cvae}}(z|x)$ initially predicts the mean and variance of the latent space distribution, denoted by (μ_s, σ_s), as represented in Equation (1). The mean and variance pair (μ_s, σ_s) are then utilized to generate latent space representations z with the help of sampling from the standard normal distribution as represented in Equation (2). The integration of a random sample, denoted by ϵ, is referred to as the *reparametrization trick* and is necessary to avoid computationally expensive simulation-driven latent space estimation [17]. Similarly, a probabilistic decoder, denoted by $p_\theta^{\mathrm{cvae}}(x|z)$, is applied so as to obtain a reconstruction, $\tilde{x} \in \mathbb{R}^d$, of the provided data x represented by Equation (3).

$$(\mu_s, \sigma_s) \leftarrow q_\phi^{\mathrm{cvae}}(x); \quad x \in \mathbb{R}^d \tag{1}$$

$$z \leftarrow \mu_s + \sigma_s \odot \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, I_s) \tag{2}$$

$$\tilde{x} \leftarrow p_\theta^{\mathrm{cvae}}(z); \quad z \in \mathbb{R}^s, \tilde{x} \in \mathbb{R}^d \tag{3}$$

Equations (1) to (3) are considered to be one forward pass of the VAE model. On the other hand, the backward pass of the VAE is carried out with the help of a total loss function represented by Equation (4). The total loss comprises of two errors: the reconstruction error, denoted by $L^{\mathrm{recon}}(\theta, x, z)$ and indicated by Equation (5); the Kullback-Leibler divergence, denoted by $L^{\mathrm{KL}}(\phi, \theta, x, z)$ and derived in (6), which measures the similarity of latent space distributions based on the probabilistic encoders and decoders.

$$L(\theta, \phi, x, z) = L^{\mathrm{recon}}(\theta, x, z) + L^{\mathrm{KL}}(\phi, \theta, x, z) \tag{4}$$

$$L^{\mathrm{recon}}(\theta, x, z) = -\mathbb{E}_{z \sim q_\phi^{\mathrm{cvae}}(z|x)}\Big(\log p_\theta^{\mathrm{cvae}}(x|z) \Big) \tag{5}$$

$$L^{\mathrm{KL}}(\phi, \theta, x, z) = D_{\mathrm{KL}}\Big(q_\phi^{\mathrm{cvae}}(z|x) \big|\big| p_\theta^{\mathrm{cvae}}(z) \Big) \tag{6}$$

Therefore, the implementation of the Central-VAE requires the transfer of edge-level stakeholder datasets to compute the reconstruction loss as well as generate latent space embeddings, which might be infeasible due to the reasons outlined in Section I. Therefore, to learn the spatiotemporal interdependencies given the decentralized, siloed nature of stakeholder data, the SplitVAEs framework incorporates a novel split training mechanism that leverages an edge-based AE component and a server-based VAE component.

B. Edge-level autoencoder (AE) formulation

Given the aforementioned reasons, at the edge-level, we implement an AE-based framework comprising deep neural network (DNN) encoder and decoder models. We denote the siloed time series data located at each edge location $n \in \mathcal{N}$ as y^n. The functional forms of the encoder and decoder components are denoted by $x^n \leftarrow F_{\theta_E^n}(y^n)$ and $\tilde{y}^n \leftarrow G_{\theta_D^n}(x^n)$, respectively. For the edge-based AE framework, let $x^n, \tilde{y}^n, \theta_E^n, \theta_D^n$ be the low-dimensional embedding, the reconstructed vector, and the model parameters of the encoder and decoder, respectively. Unlike conventional AE training, a unique aspect of the SplitVAEs training mechanism is the independent training of the encoder and decoder components to holistically capture spatial and temporal interdependencies among all the edge devices.

C. Server-level VAE formulation

At the server level, we implement a conventional VAE framework. However, unlike the Central-VAE, in each training epoch, the server model consumes the set of low dimensional embeddings, denoted by $\{x^1, \ldots, x^{N-1}, x^N\}$, supplied by the edge-based AEs at the various stakeholders. Let $q_\phi(z|x)$ be the probabilistic encoder, responsible for deriving the latent vector z from the input embeddings $\{x\}$ in terms of empirical mean and standard deviation - denoted by $\hat{\mu}$ and $\hat{\sigma}$, respectively. Additionally, we denote $z \leftarrow R_{\hat{\mu},\hat{\sigma}}(\epsilon)$ as the reparametrization function akin to Equation (2). The second core component is the latent vector z representing the compressed data in a lower dimension. The size of z becomes a crucial hyperparameter that necessitates careful tuning, in conjunction with the overall architecture of the entire network. The final component is the probabilistic decoder, denoted by $p_\theta(x|z)$, responsible for reconstructing a data point x from a given latent vector z.

D. Loss Functions

Here, we discuss the relevant formulations of the reconstruction error and the KL loss that are utilized by the SplitVAEs framework for training.

Reconstruction Loss: We employ the binary cross-entropy (BC) loss function at the n-th edge location for all $n \in \mathcal{N}$ to measure the mean reconstruction error between predicted values $\tilde{y}^n$ and observed values y^n as represented in Equation (7).

$$L_n^{\text{BC}} = -\frac{1}{B} \sum_{b=1}^{B} \left[\tilde{y}^b log(y^b) + (1 - \tilde{y}^b) log(1 - y^b) \right] \quad (7)$$

In Equation (7), let $B, \tilde{y}_b, y_b$ be the batch size, predicted, and target data for scenario generation, respectively. To preserve the simplicity of notations, we assume without loss of generality that the loss function L_n^{BC} is computed with respect to flattened representations of local tensor data and estimates both y^n and $\tilde{y}^n$ of dimension d^n. We note that the computation of L_n^{BC} can be carried out independently at each edge location since it relies purely on siloed data, represented by y^n.

KL Loss: We utilize the formulation given in Equation (8) to derive the relevant KL loss function as follows:

$$L^{\text{KL}} = -\frac{1}{2} \left[1 + 2\log(\hat{\sigma}) - \hat{\mu}^2 - \hat{\sigma} \right] \quad (8)$$

In Equation (8), the computed KL loss depends only on the latent space embeddings resulting from the reparametrization trick described in Section III-A. Consequently, the backpropagation of the error resulting from the KL loss can be initiated from the server level itself. Therefore, the decomposed reconstruction and KL error terms can be leveraged to effect a computationally efficient training mechanism of the SplitVAEs framework.

IV. SplitVAEs Algorithmic Framework

There are two crucial characteristics of the SplitVAE decomposition scheme that can be advantageous to its training. First, the KL loss, L^{KL}, is purely a factor of the latent space embeddings estimated by the server-based VAE framework. Second, the reconstruction error can be computed in a fully decentralized fashion by each of the edge locations locally without revealing the siloed datasets themselves. Thus, only the local reconstruction loss needs to be back-propagated from the edge-based AE decoder to the server-based VAE decoder. This step can be immediately followed by a backpropagation of the total loss, $L^{\text{recon}} + L^{\text{KL}}$, from the server-level VAE encoder to the edge-based AE encoders. As a result of our two-pronged backpropagation mechanism, we can enable the flow of insights back and forth between edge and server-level devices, leading to seamless learning of the overall spatiotemporal interdependencies among stakeholders.

A single training epoch in our proposed framework can be divided based on edge and server level algorithmic frameworks. In the following sections, we discuss the decomposition of the forward and backward passes of the server-level VAE as well as the encoders and decoders of edge-level AEs. Specifically, without loss of generality, we consider a set of $\mathcal{N}$ edge nodes wherein $|\mathcal{N}| = N$. To accomplish the training of SplitVAEs, we leverage several common distributed memory aggregation primitives such as the scattering and gathering operations, denoted by `DM.Scatter` and `DM.Gather`, respectively. Let `Tensor.Concat`, `Tensor.Split`, and `BACKPROP` functions indicate tensor concatenation, split and backpropagation primitives, respectively. We designate the server as the root process r, while the edge-level processes are denoted by n and are similar to the concept of ranks in distributed memory frameworks. We use k to denote iterative updates of the model parameters represented by $\theta_E^n, \theta_D^n, \phi, \theta$.

A. Edge-based Algorithmic Framework

One training epoch of the edge-based AE framework consists of the following computational steps.
AE Encoder Forward Pass: The computational steps associated with the forward pass of the AE encoder are represented by the function `EdgeEncFP`. In `EdgeEncFP`, we consider local siloed training data y_e^b that is passed through the AE

encoder $F_{(\theta_E^n)^k}$ to yield the low dimensional embedding $(x_e^b)^n$. We utilize a `DM.Gather` operation to aggregate the embeddings at the server level.

Algorithm 1 AE Encoder Forward Pass

---

function ENCEDGEFP(**root**:r, **agent**:n, **epoch**:e, **batch**:b)
  Acquire local data y_e^b
  Obtain $(x_e^b)^n \leftarrow F_{(\theta_E^n)^k}(y_e^b)$
  Execute `DM.Gather`(**root**:r, **data**:$(x_e^b)^n$)
end function

AE Decoder Forward Pass: The function `EdgeDecFP` represents the reconstruction of siloed edge data. We leverage a `DM.Scatter` operation to collect the predicted values of the low-dimensional embedding outcomes of the server-level training denoted by $(\tilde{x}_e^b)^n$. These predictions are passed through the AE decoder $G_{\theta_D^k}$ to generate reconstructions and the BCE loss through Equation (7).

Algorithm 2 AE Decoder Forward Pass

---

function EDGEDECFP(**root**:r, **agent**:n, **epoch**:e, **batch**:b)
  $(\tilde{x}_e^b)^n \leftarrow$ `DM.Scatter`(**root**:r, **data**:NULL)
  Obtain $\tilde{y}_e^b \leftarrow G_{\theta_D^k}(\tilde{x}_e^b)$
  Compute reconstruction loss $(L_{BCE})_e^b$ using (7)
end function

AE Decoder Backward Pass: As an immediate next step, our implementation leverages `EdgeDecBP` to back-propagate the errors based on the reconstruction losses. In `EdgeDecBP` $(g_{\theta_D})_e^b = \nabla_{\theta_D}(L_n^{\text{recon}})_e^b$ represents the gradient of $(L_n^{\text{recon}})_e^b$ concerning θ_D and $\Delta(\tilde{x})_e^b$ represents the error at the input layer of the edge-based decoder. Finally, we use a `DM.Gather` operation to collect the errors of the low-dimensional embeddings $(\tilde{x}_e^b)^n$ at the server.

Algorithm 3 AE Decoder Backward Pass

---

function EDGEDECBP(**root**:r, **agent**:n, **epoch**:e, **batch**:b)
  Obtain $(\Delta\tilde{x})_e^b$
  Obtain $(g_D)_e^b \leftarrow$ BACKPROP$\left(G_{(\theta_D^n)^k}, (L_n^{\text{BC}})_e^b\right)$
  Update $\theta_D^{k+1} \leftarrow \theta_D^k - \eta_D.(g_D)_e^b$
  `DM.Gather`(**root**:r, **data**:$(\Delta\tilde{x}_e^b)^n$)
end function

AE Encoder Backward Pass: The last step of the edge based training epoch is presented in `EdgeEncBP`. We leverage `DM.Scatter` operation that collects errors of low dimensional embeddings $(\Delta x_e^b)^n$ computed by the server. Consequently, the AE encoder back-propagates these errors through the local encoder $F_{(\theta_E^n)^k}$ to complete the training epoch.

B. Server-based Algorithmic Framework

At the server, a training epoch corresponding to the VAE can be described based on the following components.

VAE Forward Pass: The forward pass of the server VAE framework is represented by `VAEServerFP`. Specifically,

---

Algorithm 4 AE Encoder Backward Pass

---

function EDGEENCBP(**root**:r, **agent**:n, **epoch**:e, **batch**:b)
  $(\Delta x_e^b)^n \leftarrow$ `DM.Scatter`(**root**:r, **data**:NULL)
  $(\Delta y)_e^b, (g_E)_e^b \leftarrow$ BACKPROP$\left(F_{(\theta_E^n)^k}, \Delta x_e^b\right)$
  Update $(\theta_E^n)^{k+1} \leftarrow (\theta_E^n)^k - \eta_E.(g_E)_e^b$
end function

---

the server concatenates the low-dimensional embeddings obtained as a consequence of the `DM.Scatter` operation. The resulting concatenated embedding x_e^b is passed through the probabilistic encoder q_ϕ and is reparametrized with the standard normal distribution to yield the latent space representation z. Subsequently, we pass z through the probabilistic decoder to yield estimated low-dimensional embeddings, $\tilde{x}_e^b$, which are split into N components according to the mandated dimensions of each edge agent.

VAE Backward Pass: The backward pass of the VAE is represented by `VAEServerBP`. The backpropagated set of error values, denoted by $\left[(\Delta\tilde{x}_e^b)^1, (\Delta\tilde{x}_e^b)^2, \ldots, (\Delta\tilde{x}_e^b)^N\right]$ resulting from L_n^{recon} are acquired from each edge agent through `DM.Gather`. These error terms are concatenated and backpropagated through the probabilistic decoders to yield $(\Delta z)_e^b$ and $(g_\theta)_e^b$ representing latent space error terms and probabilistic decoder gradients, respectively. With a subsequent backpropagation of $(\Delta z)_e^b$ through the probabilistic encoder q_ϕ, we obtain $(\Delta x^{\text{BC}})_e^b$ and $(g_\phi^{\text{BC}})_e^b$ corresponding to the error terms at the input leaves of the VAE and the gradient values for the probabilistic encoder, respectively. More importantly, the errors and gradients accumulated so far on the VAE correspond to the reconstruction error terms observed at the edge level only.

Algorithm 5 VAE Forward Pass

---

function VAESERVERFP(**root**:r, **epoch**:e, **batch**:b)
  $\left[(x_e^b)^1, \ldots, (x_e^b)^N\right] \leftarrow$ `DM.Gather`(**root**:r, **data**:NULL)
  Acquire $x_e^b \leftarrow$ `Tensor.Concat`$\left[(x_e^b)^1, (x_e^b)^2, \ldots, (x_e^b)^N\right]$
  Obtain $(\hat{\mu}_e^b, \hat{\sigma}_e^b) \leftarrow q_\phi(x_e^b)$
  Create $z_e^b \leftarrow \hat{\mu}_e^b + \hat{\sigma}_e^b \odot \epsilon_e^b$, where $\epsilon_e^b \sim \mathcal{N}(0, I_s)$
  Estimate $\tilde{x}_e^b \leftarrow p_\theta(z_e^b)$
  $\left[(\tilde{x}_e^b)^1, (\tilde{x}_e^b)^2 \ldots (\tilde{x}_e^b)^N\right] \leftarrow$ `Tensor.Split`$\left[\tilde{x}_e^b\right]$
  `DM.Scatter`(**root**:r, **data**:$\left[(\tilde{x}_e^b)^1, \ldots, (\tilde{x}_e^b)^N\right]$)
  Compute KL loss $(L^{\text{KL}})_e^b$ using (8)
end function

However, to complete VAE training, we would also need to incorporate the KL loss. Since the KL loss is purely a function of the latent space embeddings z_e^b, we backpropagate $(L^{\text{KL}})_e^b$ computed during `VAEServerFP` and accumulate additional error values at the input leaves. The final set of accumulated errors $(\Delta x)_e^b$ are split and sent back to the corresponding edge agents using `DM.Scatter`.

C. Decentralized Training Algorithm

Combining the training steps of the edge and server-level models provides us with Algorithm 7 and 8 for VAE and AE, respectively. In these algorithms, we abstract the training mechanism using the set of functions detailed in Sections IV-A and IV-B for a specific set of batch and epoch sizes.

Algorithm 6 VAE Backward Pass

---

function VAESERVERBP(**root**:r, **epoch**:e, **batch**:b)

$\left[(\Delta\tilde{x}_e^b)^1, \ldots, (\tilde{x}_e^b)^N\right] \leftarrow$ DM.Gather(**root**:r, **data**:NULL)

$\Delta\tilde{x}_e^b \leftarrow$ Tensor.Concat$\left[(\Delta\tilde{x}_e^b)^1, \ldots, (\Delta\tilde{x}_e^b)^N\right]$

▷ Backpropagate BC loss:

$(\Delta z)_e^b, (g_\theta)_e^b \leftarrow$ BACKPROP$\left(p_\theta, \Delta\tilde{x}_e^b\right)$

$\left[(\Delta\hat{\mu})_e^b, (\Delta\hat{\sigma})_e^b\right]^{\text{BC}} \leftarrow$ BACKPROP$\left(R_{\hat{\mu},\hat{\sigma}}, (\Delta z)_e^b\right)$

$(\Delta x^{\text{BC}})_e^b, (g_\phi^{\text{BC}})_e^b \leftarrow$ BACKPROP$\left(q_\phi, \left[(\Delta\hat{\mu})_e^b, (\Delta\hat{\sigma})_e^b\right]^{\text{BC}}\right)$

▷ Backpropagate KL loss:

$[(\Delta\hat{\mu})_e^b, (\Delta\hat{\sigma})_e^b]^{\text{KL}} \leftarrow$ BACKPROP$\left(R_{\hat{\mu},\hat{\sigma}}, L_{\text{KL}_e}^b\right)$

$(\Delta x^{\text{KL}})_e^b, (g_\phi^{\text{KL}})_e^b \leftarrow$ BACKPROP$\left(q_\phi, \left[(\Delta\hat{\mu})_e^b, (\Delta\hat{\sigma})_e^b\right]^{\text{KL}}\right)$

▷ Update parameters

$\theta^{k+1} \leftarrow \theta^k - \eta_\theta \cdot (g_\theta)_e^b$

$\phi^{k+1} \leftarrow \phi^k - \eta_\phi \cdot \nabla_\phi((g_\phi^{\text{BC}})_e^b + (g_\phi^{\text{KL}})_e^b)$

▷ Accumulate errors at leaves

$(\Delta x)_e^b \leftarrow (\Delta x^{\text{KL}})_e^b + (\Delta x^{\text{BC}})_e^b$

▷ Scatter errors

$\left[(\Delta x_e^b)^1, \ldots, (\Delta x_e^b)^N\right] \leftarrow$ Tensor.Split$\left[\Delta x_e^b\right]$

DM.Scatter(**root**:r, **data**:$\left[(\Delta x_e^b)^1, (\Delta x_e^b)^2, \ldots, (\Delta x_e^b)^N\right]$)

end function

The sequence of computation and communication steps outlined in Algorithms 7 and 8 is represented in Figure 2. The algorithmic architecture mentioned in this section enables planning organizations, regulators, and control centers to generate high-quality scenarios by sampling from the latent distribution space.

Algorithm 7 Training Mechanism of Server-based VAE

---

for b=0,1,2,…**BatchSize do**
  for i=0,1,2,…**MaxEpochs do**
    VAESERVERFP(root:r,epoch:e,batch:b)
    VAESERVERBP(root:r,epoch:e,batch:b)
  end for
end for

From an implementation and deployment perspective, we envision a scenario where the server can generate a latent space sample to generate the low-dimensional embeddings for different edge locations by using a pre-trained VAE. The authority at the server level can now request the scenarios from

---

Algorithm 8 Training Mechanism of Edge-based Autoencoder

---

for b=0,1,2,…**BatchSize do**
  for i=0,1,2,…**MaxEpochs do**
    EDGEENCFP(root:r,agent:n,epoch:e,batch:b)
    EDGEDECFP(root:r,agent:n,epoch:e,batch:b)
    EDGEDECBP(root:r,agent:n,epoch:e,batch:b)
    EDGEENCBP(root:r,agent:n,epoch:e,batch:b)
  end for
end for

---

each edge location corresponding to the provided latent space estimate. As a result, the server-based entity can effectively gather a sufficient number of scenarios so as to facilitate the solution of the SO problem.

V. BENCHMARK METHODS

In this section, we discuss the relevant benchmarking strategies for SplitVAEs by outlining the evaluation methods as well as the evaluation metrics employed in this study.

A. Evaluation Methods

We compare SplitVAEs with centralized scenario generation methods, including the Gaussian copula [25], [26] and the Centralized-VAE. The Gaussian copula models the joint probability distribution function (PDF) to capture the interdependent structures, specifically the spatiotemporal interdependencies within the data. Meanwhile, in SplitVAEs and Centralized-VAE, these interdependencies are modeled within the latent space z when training the probabilistic encoder $q_\phi(z|x)$ and decoder $p_\theta(x|z)$ pair, as mentioned in Section III. Further, given that the datasets and their corresponding scenarios are multidimensional in nature, we employ t-SNE graphs [27] to visualize kernel density distributions of the observed and generated data, which are transformed into one-dimensional embeddings.

B. Evaluation Metrics

In addition to visual analyses, we employ multiple quantitative evaluation metrics to evaluate the quality of the generated scenarios. For each metric, lower values indicate a better model performance.

Let $X \in \mathbb{R}^{m_1 \times d}$ be the real data and $Y \in \mathbb{R}^{m_2 \times d}$ be the generated scenarios, each comprising of d features with m_1, m_2 number of observations respectively. Thus, X and Y can be expressed as: $X = \{x_1, x_2, ..., x_{m_1}\}$, where $x_i \in \mathbb{R}^d$ for all $1 \le i \le m_1$; $Y = \{y_1, y_2, ..., y_{m_2}\}$, where $y_i \in \mathbb{R}^d$ for all $1 \le i \le m_2$. Hence, we can derive each metric as follows:

Fréchet inception distance (FID): The FID score is a widely accepted, state-of-the-art metric employed to evaluate deep generative models (DGMs) [28], [29]. The FID measures the Wasserstein distance between two PDFs $f_X(x)$ and $f_Y(y)$. In our study, the observed dataset and the generated scenarios, have finite values pertaining to their respective mean and covariance matrices denoted by μ_x, μ_y and Σ_x, Σ_y respectively.

Under such conditions, based on [30] the FID can be defined as follows:

$$d_F[X, Y] = ||\mu_x - \mu_y||_2^2 - \text{Tr}\left(\Sigma_x + \Sigma_y - 2\left(\Sigma_x \cdot \Sigma_y\right)^{\frac{1}{2}}\right) \quad (9)$$

Energy Score: The energy score (ES) measures the dissimilarity in probability density between two multivariate random variables [31], which are the observed data and generated scenarios in our study:

$$\epsilon_{m_1, m_2}(X, Y) = \frac{1}{m_1 m_2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} |x_i - y_j| - \frac{1}{2n_1^2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_1} |x_i - x_j| \quad (10)$$

Root-mean-square Error: The well-known root-mean-square error (RMSE) measures the quadratic mean of the differences between the observed values and generated ones:

$$\text{RMSE} = \sqrt{\frac{1}{kd} \sum_{i=1}^{k} \sum_{j=1}^{d} \left(x_{i,j} - y_{i,j}\right)^2} \quad (11)$$

where $k = \min(m_1, m_2)$.

Continuous ranked probability score: The CRPS [32] measures the squared distance between a single multivariate observed observation y and the median of $F(x)$ which is the empirical cumulative distribution of generated scenarios [33]. For m_2 number of observations, the CRPS is given as:

$$\text{CRPS}(F, \{y_1, ..., y_{m_2}\}) = \frac{1}{m_2} \sum_{i=1}^{m_2} \int_{-\infty}^{\infty} \left(F(x) - \mathbf{1}(x \geq y_i)\right)^2 dx \quad (12)$$

VI. EXPERIMENTAL RESULTS

Our experimental results are primarily geared towards establishing the computational efficiency of the SplitVAEs method while delivering scenarios of the same statistical quality as the underlying dataset. In this section, we demonstrate that Split-VAEs deliver scenarios possessing similar statistical qualities to the observed data, while showcasing their ability of handling heterogeneous datasets in a scalable fashion.

A. Hardware and Software Environment

All experiments were conducted using the Pete High Performance Computing infrastructure from Oklahoma State University (OSU). All experiments pertaining to large-scale distributed processing were performed using the OpenMPI library [34] coupled with Python bindings facilitated by mpi4py [35]. We utilize the PyTorch library [36] for constructing, training and evaluating the DNN models, and the inbuilt Ray Tune framework [37] for hyperparameter tuning. Specifically, we first tune the hyperparameters of the Centralized-VAE, then adapt these hyperparameters for different numbers of edge locations and desired latent dimensions in SplitVAEs.

B. Description of Key Datasets

To demonstrate the broad applicability of our proposed framework, we primarily evaluate the decentralized scenario generation capability of our proposed framework using the

USAID [38], ACES [39] and ACTIVSg datasets [40]. The US-AID dataset comprises of time series data of health commodity supplies for different countries. The ACES dataset comprises hourly carbon dioxide (CO_2) emissions from the combustion of fossil fuels at a 1km^2 resolution in the continental United States from 2012 to 2017. In the ACES dataset, we extract time series data for emissions from 25 refineries in the US Gulf Coast during Summer 2017. Lastly, the ACTIVSg dataset provides the Demand and Renewable generation data for a power transmission network with an overall footprint for a large section of the state of Texas. We focus on hourly demand of 1125 individual network buses and renewable production from 87 renewable generators respectively which are referred to as Demand and Renewable datasets respectively.

C. Applicability to a Diversity of Use Cases

Table I presents the mean and standard deviation of each evaluation metric for 100 iterations, with output dimension of edge level autoencoders set to 20. The experiments for each dataset comprises varying number of edge nodes ranging from 25 for ACES to 1125 distinct edge processes in the Demand dataset. From Table I, we can clearly observe that the performance of our proposed framework is nearly identical to those of centralized training schemes. Thus, we can expect SplitVAEs to generate high-fidelity scenarios for different application areas including but not limited to supply chains, carbon credit compliance planning and forecasting renewable energy production for power systems.

TABLE I: Means (with standard deviations measured in 10^{-3}) of various evaluation scores across four different datasets.

Dataset	Edge Nodes	Metric	Copula	Central-VAE	SplitVAEs
ACES	25	FID	2.095 (7)	2.255 (8)	2.397 (11)
		ES	0.122 (3)	0.124 (4)	0.136 (9)
		RMSE	0.201 (18)	0.218 (16)	0.304 (20)
		CRPS	0.209 (8)	0.2327 (14)	0.2649 (17)
USAID	85	FID	2.177 (7)	2.752 (7)	2.814 (6)
		ES	0.092 (1)	0.164 (1)	0.186 (1)
		RMSE	0.240 (1)	0.280 (1)	0.287 (2)
		CRPS	0.168 (1)	0.250 (2)	0.280 (10)
Renewable	87	FID	2.105 (9)	2.167 (8)	2.169 (4)
		ES	0.307 (6)	0.309 (4)	0.310 (6)
		RMSE	0.309 (5)	0.334 (10)	0.351 (14)
		CRPS	0.320 (6)	0.322 (4)	0.329 (8)
Demand	1125	FID	1.540 (9)	1.545 (16)	1.947 (18)
		ES	0.129 (9)	0.143 (17)	0.148 (18)
		RMSE	0.328 (8)	0.419 (15)	0.462 (19)
		CRPS	0.180 (3)	0.182 (5)	0.184 (11)

Additionally, Figures 3, 4, and 5 present a comparison of the observed data with scenarios generated by three different methods across four case studies. Figure 3 illustrates density curves, estimated using a Gaussian-based kernel method, of one-dimensional embeddings constructed from generated scenarios and observed data by the t-SNE method [27]. Meanwhile, Figures 4 and 5 provides a comparison of centroids - computed by averaging the quantity of interest for all the nodes at each distinct time point - and their corresponding autocorrelation coefficients, respectively.
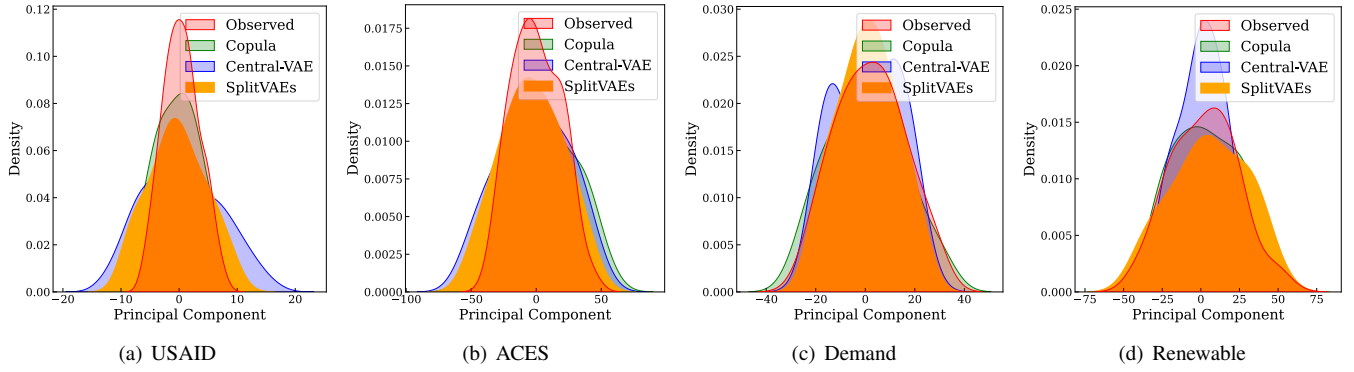
Fig. 3: Embedding distributions between observed data and generated scenarios using different methods across four cases.
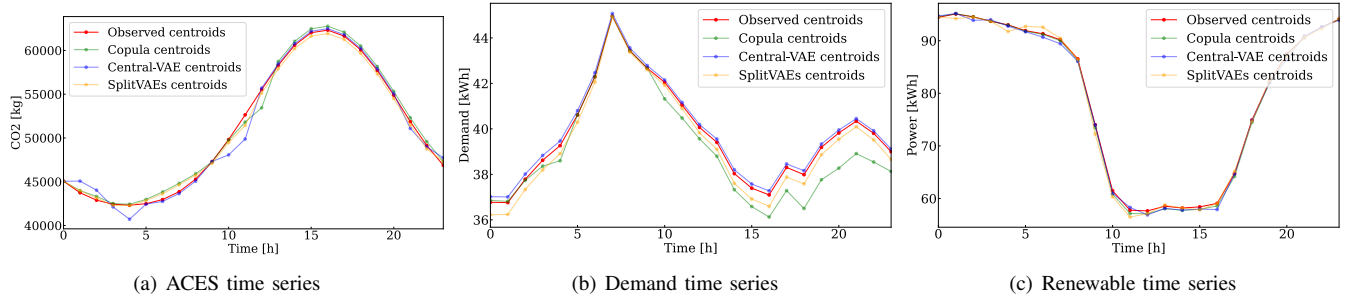

Fig. 4: Comparison of centroids between observed and generated data across four datasets.
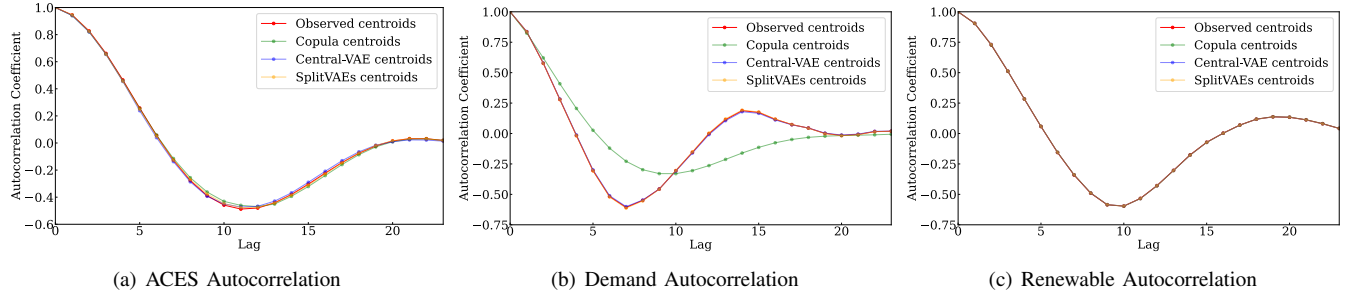

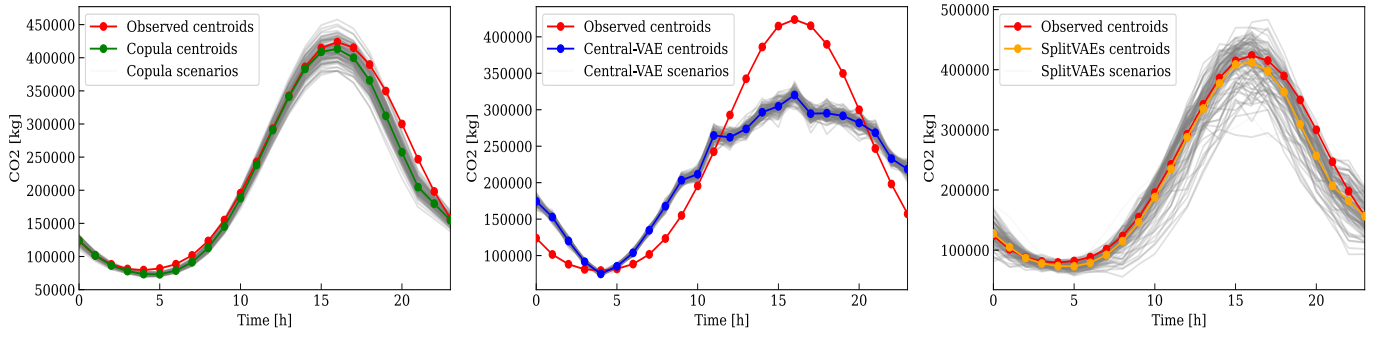Fig. 5: Comparison of autocorrelations between observed and generated data across four datasets.

From Figure 3, we can observe that all methods capture the underlying distributions well, with 95% of their principal components falling within the confidence range. Some minor misalignments, primarily result from feature compression in the latent dimension, do not significantly affect the overall alignment between the generated scenarios and the observed data. Further, Figures 4 and 5, show a close alignment between scenarios generated by SplitVAEs and state-of-the-art benchmark methods while capturing interdependencies present in original datasets both in terms of centralized and autocorrelations.

To illustrate a practical application, we present seasonal carbon emission scenarios from a refinery at Port Arthur, Texas, in Figure 6. As shown, the scenarios generated by the SplitVAEs closely align with the observed data and other benchmark methods. Specifically, within the perimeter of the industrial complex, the observe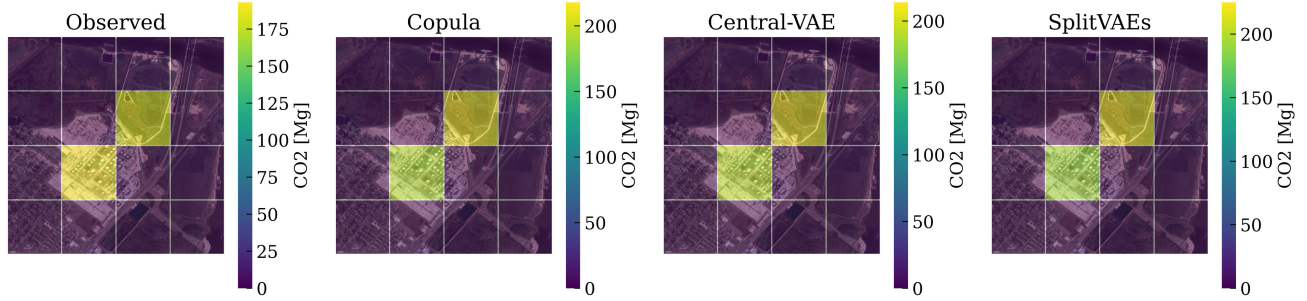d emission profile is approximately 180 to 190 Megagrams (Mg), which is accurately reflected in the scenarios generated by all three methods. Moreover, the emission patterns derived from the scenarios generated by the SplitVAEs follow the same trend as the observed data.

D. Reduction in Data Movement

To evaluate the communication or transmission efficiency of SplitVAEs, we measure the total payload size handled by the system after a full cycle of our Algorithm 7 comprising one forward pass of embeddings followed by backward pass of gradients and errors. Figure 7 compares the original dataset size with the transmission overhead for various edge dimension choices. The results show a significant reduction in data size as latent dimensions decrease. For instance, in the Demand dataset, the transmission size drops from 1830 MB for the original data to 475 MB at latent size 20, 223 MB at size 16, and 154 MB at size 8. Data transmission reduction

(a) Comparison of temporal information (each gray line represents a single-day scenario of emission volume generated by a particular method).

(b) Comparison of spatial information (each grid represents a $1km^2$ resolution of mean emission volume computed over 92 days).

Fig. 6: A breakdown analysis of the spatial and temporal information extracted from generated scenarios at Port Arthur, Texas.
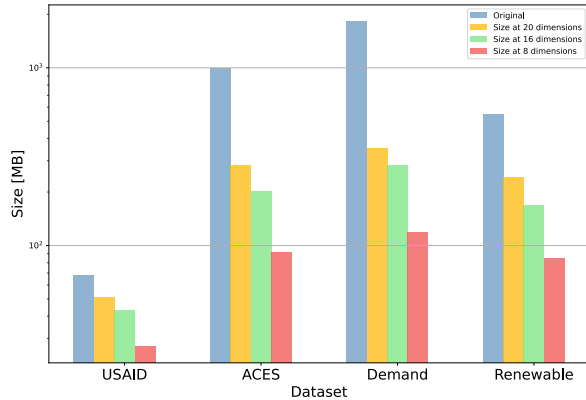

Fig. 7: Analysis of reduction in transmitted data size, measured in log-scale, across different latent dimensions.

factors at the smallest dimensions are 4.7, 2.3, 2.8, and 2.7 for the USAID, ACES, Demand, and Renewable datasets, respectively with respect to original data sizes. Combined with Figure 3, Figure 5, and Table I, these findings demonstrate that SplitVAEs deliver high-fidelity scenarios with up to nearly a 5 times reduction in data transmission costs.

E. Handling Dimensional Heterogeneity

In this case study, we evaluate the ability of SplitVAEs to handle heterogeneous data dimensions present across various stakeholders. Dimensional heterogeneity refers to differences in the size and structure of datasets across various edge devices in a decentralized network. For our experiments, we

decompose the Demand dataset into regions with varying numbers of power system buses, where each bus represents a unique temporal demand curve. This results in multiple region decomposition cases, with each region acting as a distinct edge device handling a different dimensional dataset. Our decomposition cases rely on graph-partitioning schemes [41] that attempt to decompose the power network graph by minimizing the total number of inter-regional transmission lines. As a result, we obtain three different decompositions pertaining to $n = 20$ regions, $n = 60$ regions, and $n = 120$ regions wherein, each region represents a distinct SplitVAE edge node and the dimension of local data at each node is non-homogeneous.

Figure 8 provides a visual representation of the data, depicted as means (solid circle) and standard deviations (whiskers), generated by the SplitVAEs model across the three different regional decompositions. The plots demonstrate that the model can adapt to varying local dataset dimensions, effectively capturing the underlying data distributions in all cases. Therefore, the results suggest that the SplitVAEs framework is robust to dimensional heterogeneity, delivers high performance across different regional decompositions. On the other hand, Figure 9 illustrates the global training loss for the SplitVAEs model across different regional decompositions of the Demand dataset. The x-axis represents the number of global training epochs, while the y-axis shows the total training loss, which includes both reconstruction loss and KL divergence loss. Each line in the graph corresponds to a different regional decomposition setting. As observed, all trends show initial

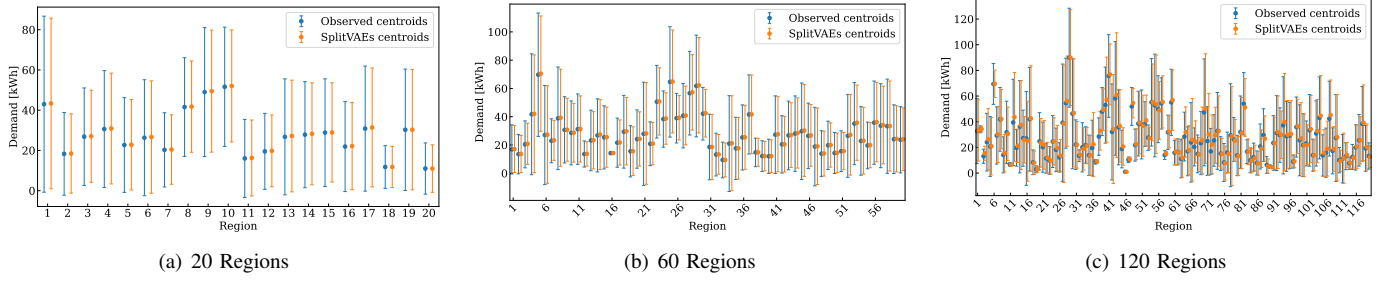(a) 20 Regions   (b) 60 Regions   (c) 120 Regions

Fig. 8: An analysis of scenarios generated using three different region decompositions, compared with the observed data.
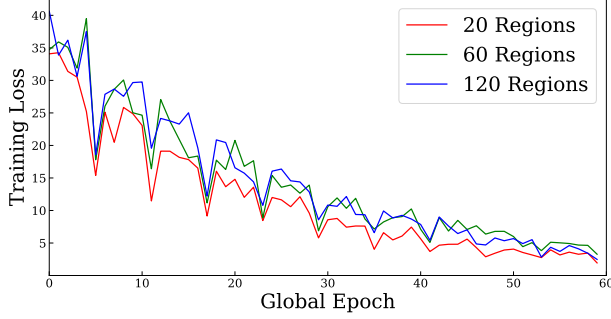

Fig. 9: Global training losses across various settings of region decompositions in the ACTIVSg-Demand dataset.

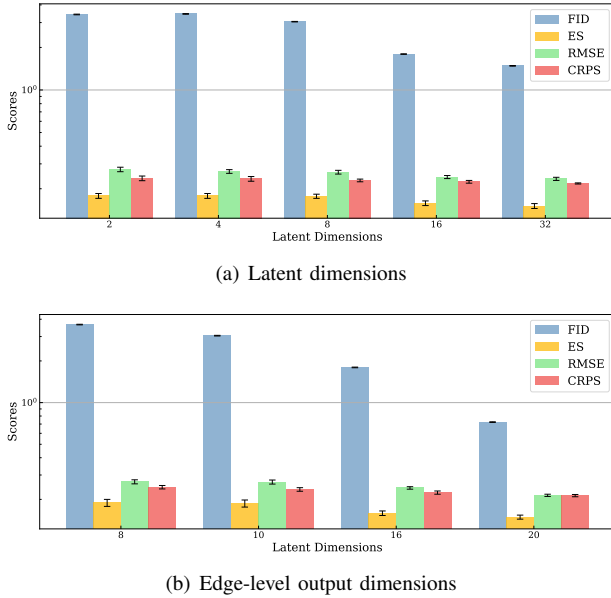losses that decrease over subsequent epochs, indicating the model's convergence.

(a) Latent dimensions

(b) Edge-level output dimensions

Fig. 10: Analysis of the SplitVAEs architecture's robustness on ACTIVSg Demand, presented in log-scale.

F. Architectural Robustness

In this case study, we aim to evaluate the robustness of SplitVAEs with varying dimensions of edge-level AE outputs

and server-level VAE latent embeddings across all four evaluation metrics. For each experiment, we consider 100 scenario generation instances employed to provide a statistically sound set of scores for evaluation. Our results are presented in Figure 10 with respect to the Demand dataset, where means and standard deviations are depicted as bars and whiskers, respectively. In Figure 10a, we observe that an increase in the VAE latent embedding size, from 2 to 32, greatly improves the performance of the SplitVAEs, as evidenced by lower scores across all metrics. Notably, the FID scores exhibit a significant improvement, indicating that models with larger latent dimensions are generating outcomes that closely resemble trends in the original dataset. Meanwhile, Figure 10b demonstrates that increasing AE output sizes in general leads to a higher quality of scenarios as expected. Overall, the trends presented in Figure 10 conclusively show that the performance of SplitVAEs remains robust to varying degree of latent dimensions across both server and edge models.

VII. CONCLUSION

In the paper we present SplitVAEs, a novel decentralized method to generate spatiotemporally interdependent scenarios from siloed data specifically geared toward solving SO problems. Using backpropagation, we effectively decompose the training across edge-level autoencoders and a server-level variational autoencoder and enable the bi-directional flow of insights among the edge and server models. We demonstrate the applicability of SplitVAEs specifically in the context of multi-stakeholder infrastructure systems by leveraging real-world datasets from various areas including supply chains, energy, and environmental science. We show that SplitVAEs generate scenarios that closely align with the real underlying data distributions by benchmarking with the state-of-the-art centralized methods and numerous statistical metrics. Using large scale, distributed memory driven experiments, our results indicate that SplitVAEs can significantly reduce data transmission, scale efficiently with growing number of edge locations and help break down data silos. Overall, our analysis demonstrates that SplitVAEs are a compelling alternative that enables scenario generation for SO problems in multi-stakeholder-led infrastructure systems without the need to transfer underlying datasets.

REFERENCES

[1] C. Zhao and Y. Guan, "Data-driven stochastic unit commitment for integrating wind generation," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 2587–2596, 2015.

[2] S. Chopra and P. Meindl, *Supply chain management. Strategy, planning & operation*. Springer, 2007.

[3] J. Shi, H. Rozas, M. Yildirim, and N. Gebraeel, "A stochastic programming model for jointly optimizing maintenance and spare parts inventory for iot applications," *IISE Transactions*, vol. 55, no. 4, pp. 419–431, 2023.

[4] R. M. Anderson, *Stochastic models and data driven simulations for healthcare operations*. PhD thesis, Massachusetts Institute of Technology, 2014.

[5] A. L. Erera, J. C. Morales, and M. Savelsbergh, "The vehicle routing problem with stochastic demand and duration constraints," *Transportation Science*, vol. 44, no. 4, pp. 474–492, 2010.

[6] A. Shapiro, D. Dentcheva, and A. Ruszczyński, *Lectures on Stochastic Programming: Modeling and Theory, Second Edition*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2014.

[7] Y. Li, J. Li, and Y. Wang, "Privacy-preserving spatiotemporal scenario generation of renewable energies: A federated deep generative learning approach," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2310–2320, 2021.

[8] C. Goncalves, R. J. Bessa, and P. Pinson, "Privacy-preserving distributed learning for renewable energy forecasting," *IEEE Transactions on Sustainable Energy*, vol. 12, no. 3, pp. 1777–1787, 2021.

[9] J. M. Morales, R. Minguez, and A. J. Conejo, "A methodology to generate statistically dependent wind speed scenarios," *Applied Energy*, vol. 87, no. 3, pp. 843–855, 2010.

[10] A. Papavasiliou, S. S. Oren, and R. P. O'Neill, "Reserve requirements for wind power integration: A scenario-based stochastic programming framework," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2197–2206, 2011.

[11] P. Pinson, H. Madsen, H. A. Nielsen, G. Papaefthymiou, and B. Klöckl, "From probabilistic forecasts to statistical scenarios of short-term wind power production," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 12, no. 1, pp. 51–62, 2009.

[12] T. Werho, J. Zhang, V. Vittal, Y. Chen, A. Thatte, and L. Zhao, "Scenario generation of wind farm power for real-time system operation," *arXiv preprint arXiv:2106.09105*, 2021.

[13] J. Liang and W. Tang, "Sequence generative adversarial networks for wind power scenario generation," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 1, pp. 110–118, 2019.

[14] A. Papavasiliou, S. S. Oren, and I. Aravena, "Stochastic modeling of multi-area wind power production," in *2015 48th Hawaii international conference on system sciences*, pp. 2616–2626, IEEE, 2015.

[15] D. Saxena and J. Cao, "Generative adversarial networks (gans) challenges, solutions, and future directions," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–42, 2021.

[16] I. Rios, R. J. Wets, and D. L. Woodruff, "Multi-period forecasting and scenario generation with limited data," *Computational Management Science*, vol. 12, no. 2, pp. 267–295, 2015.

[17] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, 2019.

[18] S. Somapa, M. Cools, and W. Dullaert, "Characterizing supply chain visibility–a literature review," *The International Journal of Logistics Management*, vol. 29, no. 1, pp. 308–339, 2018.

[19] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.

[20] Y. Li, R. Wang, Y. Li, M. Zhang, and C. Long, "Wind power forecasting considering data privacy protection: A federated deep reinforcement learning approach," *Applied Energy*, vol. 329, p. 120291, 2023.

[21] M. Jia, C. Shen, and Z. Wang, "A distributed probabilistic modeling algorithm for the aggregated power forecast error of multiple newly built wind farms," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 4, pp. 1857–1866, 2018.

[22] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.

[23] A. Singh, P. Vepakomma, O. Gupta, and R. Raskar, "Detailed comparison of communication efficiency of split learning and federated learning," *arXiv preprint arXiv:1909.09145*, 2019.

[24] M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, "Split learning for collaborative deep learning in healthcare," *arXiv preprint arXiv:1912.12115*, 2019.

[25] J. P. Restrepo, J. C. Rivera, H. Laniado, P. Osorio, and O. A. Becerra, "Nonparametric generation of synthetic data using copulas," *Electronics*, vol. 12, no. 7, 2023.

[26] R. Houssou, M.-C. Augustin, E. Rappos, V. Bonvin, and S. Robert-Nicoud, "Generation and simulation of synthetic datasets with copulas," 2022.

[27] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008.

[28] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *CoRR*, vol. abs/1812.04948, 2018.

[29] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," *CoRR*, vol. abs/1912.04958, 2019.

[30] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of Multivariate Analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[31] G. J. Székely and M. L. Rizzo, "Energy statistics: A class of statistics based on distances," *Journal of Statistical Planning and Inference*, vol. 143, no. 8, pp. 1249–1272, 2013.

[32] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management Science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[33] M. B. Bjerregård, J. K. Møller, and H. Madsen, "An introduction to multivariate probabilistic forecast evaluation," *Energy and AI*, vol. 4, p. 100058, June 2021.

[34] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, *et al.*, "Open mpi: Goals, concept, and design of a next generation mpi implementation," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface: 11th European PVM/MPI Users' Group Meeting Budapest, Hungary, September 19-22, 2004. Proceedings 11*, pp. 97–104, Springer, 2004.

[35] L. Dalcin and Y.-L. L. Fang, "mpi4py: Status update after 12 years of development," *Computing in Science & Engineering*, vol. 23, no. 4, pp. 47–54, 2021.

[36] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[37] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[38] "Usaid global health supply chain program - procurement and supply management (ghsc-psm) health commodity delivery," 2018.

[39] C. Gately and L. Hutyra, "Anthropogenic carbon emission system, 2012-2017, version 2. ornl daac, oak ridge, tennessee, usa," 2018.

[40] H. Li, J. H. Yeo, A. L. Bornsheuer, and T. J. Overbye, "The creation and validation of load time series for synthetic electric power systems," *IEEE Transactions on Power Systems*, vol. 36, no. 2, pp. 961–969, 2021.

[41] G. Karypis and V. Kumar, "Metis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices," 1997.