FedEvolve: Evolutionary Tabular Data Synthesis in Vertical Federated Learning Systems

Adethya Srinivasan, Han Wu

University of Southampton
as37g23@soton.ac.uk, H.Wu@soton.ac.uk

Abstract

Generating synthetic data in a privacy-preserving manner is a key challenge when real data is distributed across multiple parties. Vertical Federated Learning (VFL) addresses this by allowing parties to jointly learn from different features of the same individuals without sharing raw data. However, existing VFL frameworks introduce significant privacy risks, such as information leakage from shared gradients. We propose FedEvolve, a novel VFL framework for privacy-preserving synthetic tabular data generation using a co-evolutionary Variational Autoencoder (VAE) combined with a Latent Diffusion Model (LDM).

First, a co-evolutionary algorithm optimizes the VFL-VAE. This approach avoids federated backpropagation by concurrently evolving separate populations of client-side autoencoders and server-side fusion models. Second, an LDM is trained on its latent space to capture the joint data distribution across parties. Synthetic tabular records are generated by sampling from the LDM and decoding through the client-side decoders. We empirically show that FedEvolve is able to generate synthetic data that matches the distribution of the raw data.

Introduction

The demand for high-quality data in machine learning is often unmet in critical sectors (e.g., healthcare, finance) (Rahman 2025) where data is siloed due to strict privacy and data governance requirements (Madathil et al. 2025). In such settings, different organizations typically possess distinct subsets of features pertaining to the same set of individuals, yet are unable to exchange raw data. Vertical Federated Learning (VFL) (Rahman 2025) has emerged as a principled solution to this problem, enabling collaborative model training on vertically-partitioned data while preserving data confidentiality (Liu et al. 2024).

Generating synthetic data in VFL is a valuable but difficult task (Polato 2021; Zhao et al. 2025), as it requires learning a shared latent representation from disjoint features without violating privacy (Rashad et al. 2024). Existing gradient-based VAEs for this task (Rashad et al. 2024; Shankar et al. 2024; Polato 2021) suffer from communication bottlenecks and, more critically, privacy risks from potential inference attacks on shared gradients or embeddings (Chang and Zhu 2024).

To address these fundamental privacy and security limitations, we propose a paradigm shift: replacing gradient-based optimization with a co-evolutionary architecture to train a distributed VAE. Evolutionary algorithms (EAs) (Parsons 1998) are gradient-free, black-box optimizers, suited for distributed (Desell et al. 2010) or non-differentiable (Li et al. 2024) problems, and have a history of optimizing neural networks (Stanley and Miikkulainen 2002; Wu, Cao, and Qi 2025). To the best of our knowledge, our contributions are as follows:

1. **A Co-evolutionary VFL-VAE Training Framework:** A VFL system where client encoders/decoders and a server-side fusion model are co-evolved as distinct populations. This fully eliminates the need for gradient sharing, mitigating privacy risks.
2. **Integration of Memetic Local Search:** We enhance the co-evolution process with a gradient-free (1+1)-Evolutionary Strategy (Ryan 2003), which improves convergence by iteratively refining inddividuals through local search.
3. **An End-to-End Evolutionary VFL-LDM Framework:** The evolved VFL-VAE provides a shared latent space for a Latent Diffusion Model (LDM) (Rombach et al. 2022; de Goede, Cox, and Decouchant 2024; Shankar et al. 2024), creating a complete, evolutionary-trained pipeline for privacy-preserving synthetic tabular data generation.

Related Work

Generative Models in Vertical Federated Learning

Generative modeling in VFL typically adapts centralized, gradient-based models like VAEs or GANs (Zhao et al. 2025). These VFL-VAEs learn a shared latent space by securely aggregating intermediate computations (e.g., latent vectors, gradients) (Rashad et al. 2024; Polato 2021; Shankar et al. 2024). While effective, this reliance on shared computations creates two challenges: (1) **Communication Overhead** from high-dimensional data exchange (Ángel Morell et al. 2022), and (2) **Privacy & Security Risks** from potential membership inference or data reconstruction attacks on shared gradients or embeddings (Chang and Zhu

2024). These risks present significant data governance and ethical challenges. Our work directly addresses these issues by adopting a gradient-free paradigm, eliminating this entire attack surface.

Neuroevolution and Co-evolution

We replace backpropagation with neuroevolution, using evolutionary algorithms (EAs) to optimize neural network parameters (Stanley and Miikkulainen 2002). EAs are gradient-free black-box optimizers, robust in non-differentiable landscapes (Li et al. 2024). This approach has successfully evolved VAEs and other networks in centralized settings (Wu, Cao, and Qi 2025).

We employ a co-evolutionary framework, which decomposes a problem into interacting sub-components (Rosin and Belew 1997). This maps perfectly to VFL: client-side models and the server-side fusion model are treated as separate, co-evolving populations. To the best of our knowledge, we are the first to apply a co-evolutionary memetic algorithm to train a VFL-VAE, thus bypassing gradient exchange.

Latent Diffusion Models for Tabular Data

For synthesis, our framework uses Latent Diffusion Models (LDMs) (Rombach et al. 2022). LDMs are are effective for mixed-type (continuous/categorical) tabular data (Shankar et al. 2024) because they operate in a compressed, continuous latent space learned by a VAE (Rombach et al. 2022). This two-stage process simplifies the generative task. Our novelty is not the LDM, but the distributed, gradient-free co-evolutionary method used to train the VAE and obtain this latent space. Large foundation models (FMs) are increasingly in need of synthetic training data which are held in distributed data silos. Our approach provides a necessary paradigm for high-stakes, privacy-sensitive tabular data that is vertically-partitioned.

Methodology

The framework operates in two stages: (1) a distributed Vertical Federated VAE (VFL-VAE) learns a shared latent manifold, and (2) a Latent Diffusion Model (LDM) learns this manifold for data generation.

Framework Design

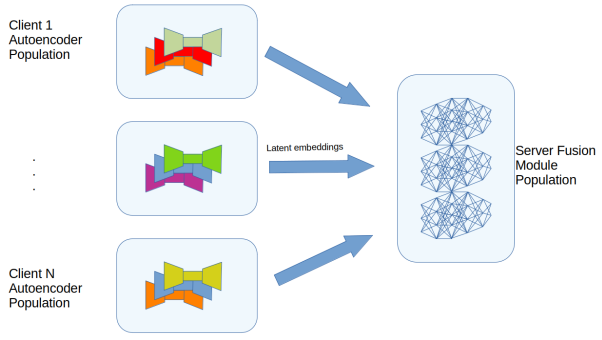

Figure 1: Architectural Setup of the FedEvolve System

Vertical Federated VAE Architecture The VFL-VAE consists of N client-side models and one server-side model.

Client-Side Models Each client $i \in \{1, ..., N\}$ has:

- **Encoder** (E_i): Maps local features $x_i \in \mathbb{R}^{d_i}$ to the parameters of a Gaussian distribution: mean $\mu_i \in \mathbb{R}^k$ and log-variance $\log(\sigma_i^2) \in \mathbb{R}^k$.
- **Decoder** (D_i): Reconstructs the original data $\hat{x}_i$ from a partition of the fused latent vector, $z_{\text{fused},i} \in \mathbb{R}^{f/N}$.

Server-Side Model The server coordinates representation fusion.

- **Server Bottleneck** (S): A fusion module. During a forward pass, each client i samples $z_i = \mu_i + \epsilon \odot \sigma_i$ (where $\epsilon \sim \mathcal{N}(0, I)$) and sends it to the server. The server concatenates these vectors into $[z_1, z_2, ..., z_N] \in \mathbb{R}^{N \times k}$ and passes them through S to produce a final, compressed representation $z_{\text{fused}} \in \mathbb{R}^f$. This z_{fused} is then partitioned and distributed back to the client decoders.

Objective Function as Fitness The VFL-VAE objective, minimized as a loss function, defines the evolutionary fitness. For a batch of B samples, the total loss L_{VFL} combines reconstruction and KLD terms.

The reconstruction loss L_{rec} is a composite of Mean Squared Error (MSE) for continuous features and Cross-Entropy (CE) for categorical features (using one-hot vectors and logit outputs). The total reconstruction loss is:

$$L_{\text{rec}} = \sum_{m=1}^{B} \sum_{i=1}^{N} \left(\sum_{j \in \text{cont}_i} L_{\text{MSE}}(x_{ij}^{(m)}, \hat{x}_{ij}^{(m)}) + \sum_{k \in \text{cat}_i} L_{\text{CE}}(x_{ik}^{(m)}, \hat{x}_{ik}^{(m)}) \right)$$

where $x^{(m)}$ is the m-th sample.

The KLD term L_{KLD}, computed on the concatenated pre-fusion client vectors, regularizes the latent space towards a standard normal prior $\mathcal{N}(0, I)$:

$$L_{\text{KLD}} = -\frac{1}{2} \sum_{m=1}^{B} \sum_{i=1}^{N} \sum_{j=1}^{k} (1 + \log(\sigma_{ij}^{2(m)}) - \mu_{ij}^{2(m)} - \sigma_{ij}^{2(m)})$$

The total loss is $L_{\text{VFL}} = \frac{1}{B}(L_{\text{rec}} + \beta \cdot L_{\text{KLD}})$, with an annealed β. The evolutionary fitness is Fitness $= -L_{\text{VFL}}$.

Latent Diffusion Model for Data Synthesis Once the distributed VAE is trained, the best VAE components encode the entire training data into a centralized, normalized latent dataset. Data synthesis follows process from (Rombach et al. 2022; Shankar et al. 2024):

1. An LDM is trained on this normalized latent dataset.
2. The LDM generates new latent vectors from Gaussian noise.
3. These vectors are denormalized and passed through the fixed, evolved client decoders to synthesize new tabular records.

Co-evolutionary Memetic Algorithm

Each client model set $\{(E_i, D_i)\}$ in the initial population P_C is pre-trained locally as a standalone VAE on its own data partition x_i. This pre-training provides a warm start, allowing client encoders and decoders to learn a reasonable local representation before co-evolutionary search begins. The

client's take a partition of the latent embedding $z_i \in \mathbb{R}^k$ locally to $z_{part} \in \mathbb{R}^f$. This encourage the model to learn good local representations before focusing on the more difficult problem of cooperative fusion and reconstruction. The VFL-VAE is not trained with gradient descent. Instead, we use a co-evolutionary memetic algorithm (Wu, Cao, and Qi 2025; Kato et al. 2025) to optimize the distributed components in a gradient-free manner, avoiding the security risks and overhead of gradient exchange (Ángel Morell et al. 2022; Chang and Zhu 2024). This approach is defined by the following components (see Algorithm 1 in Appendix for further details):

- **Decoupled Populations:** We maintain two distinct populations: P_C for client-side model sets ($\{(E_1, D_1), ..., (E_N, D_N)\}$) and P_S for server-side fusion models (S).
- **Gradient-Free Fitness Evaluation:** Fitness is the negative validation loss ($-L_{\text{VFL}}$), a scalar value that eliminates gradient-based attack surfaces. An individual's fitness is its average performance when collaborating with a 'committee' of the top-K individuals from the opposing population, promoting generalist solutions. For a client set C_i and server committee S^*:

$$F(C_i) = \frac{1}{|S^*|} \sum_{S_j \in S^*} -\mathcal{L}_{VAE}(C_i, S_j)$$

- **Genetic Operators:** We use tournament selection and elitism. Crossover and mutation are specialized:
  - **Crossover:** Server children are created by averaging parent weights. Client set children use **module-level uniform crossover**, inheriting each (E_i, D_i) pair from a random parent.
  - **Mutation:** Weights w are perturbed via additive Gaussian noise: $w' = w + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma_m^2)$. The mutation strength σ_m follows a cosine decay schedule, and weights are clipped to $[-5.0, 5.0]$.
- **Memetic Local Search:** To accelerate convergence, each new child undergoes a (1+1)-Evolution Strategy. The individual is repeatedly mutated with a small, fixed strength, and the mutation is kept only if it improves or matches the current fitness.

Experimental Results

We conduct experiments to validate the effectiveness of our proposed co-evolutionary VFL framework. We aim to answer two primary questions: (1) Does the gradient-free co-evolutionary algorithm successfully converge to a stable VFL-VAE? (2) Does the full framework generate synthetic data that retains the statistical properties and utility of the original data?

Experimental Setup

Dataset: We present preliminary results on the **Higgs dataset** (Whiteson 2014), a standard benchmark in machine learning. For our VFL simulation, we horizontally partition the 28 features into two sets, assigning 14 features to Client 1 and 14 features to Client 2.

Evaluation Metrics: We evaluate the quality of the generated synthetic data using two standard approaches:

- **Statistical Fidelity:** We measure the similarity between the real and synthetic data distributions. This is done qualitatively using Kernel Density Estimation (KDE) plots and dimensionality reduction (PCA, t-SNE (Cai and Ma 2022), UMAP (McInnes, Healy, and Melville 2020)) projections. Quantitatively, we report the average 1-D **Wasserstein Distance** between the distributions of each feature.
- **Downstream Utility:** We use the 'Train on Synthetic, Test on Real' (TSTR) paradigm. We train a simple classifier (Logistic Regression) on the generated synthetic data and test its performance on a held-out set of real data. We compare this TSTR accuracy to the 'Train on Real, Test on Real' (TRTR) accuracy, which serves as the performance upper bound. A TSTR accuracy close to the TRTR accuracy indicates high data utility.

Implementation Details: For the co-evolutionary algorithm, we used population sizes of 20 for clients and 20 for the server, evolving for 1000 generations. The LDM was trained for 250 steps.

Qualitative Results and Training Analysis

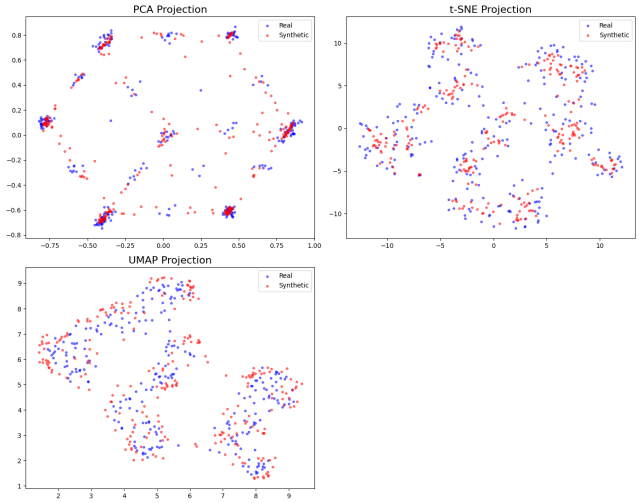

Figure 3: Projections of Real (Blue) and Synthetic (Red) Data using PCA, t-SNE, and UMAP. The significant overlap suggests the global structure is preserved.

Statistical Fidelity: Fig. 2 compares the distributions of real and synthetic data for a random subsample of features. The KDE plots show a high degree of visual similarity, indicating that the LDM successfully learned to sample from the VAE's latent manifold. Furthermore, Fig. 3 visualizes the PCA, t-SNE, and UMAP projections of both real (red) and synthetic

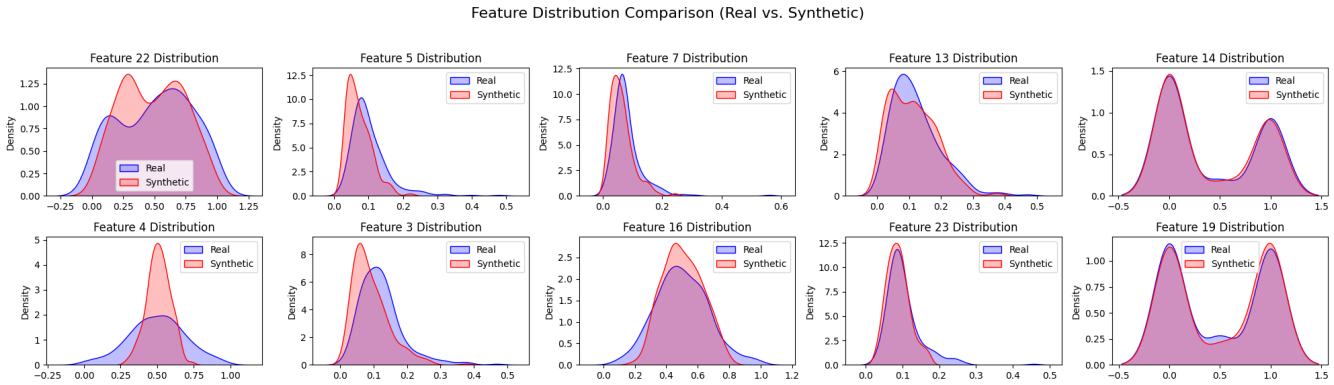

Figure 2: Feature Distribution Comparison between Real and Synthetic Data. (Blue: Real, Red: Synthetic). The synthetic distributions closely track the real data.

(blue) datasets. The strong overlap between the real and synthetic clusters suggests that our framework not only captures individual feature distributions but also preserves the multi-dimensional correlations and underlying data manifold.

Quantitative Analysis

We further quantify the performance using statistical and utility-based metrics.

TRTR Accuracy	0.6371
TSTR Accuracy	0.5184
TSTR Performance vs. TRTR	81.36%

Table 1: Downstream utility results (TSTR/TRTR) using a Logistic Regression classifier.

Downstream Utility: Table 1 presents the TSTR and TRTR results. The model trained on synthetic data achieves an accuracy of 0.5184 on the real test set, which is 81.36% of the performance of the model trained on real data (0.6371). This result indicates that the synthetic data retains downstream utility, making it a viable privacy-preserving substitute for the real data in this classification task.

Statistical Fidelity: We also provide the average 1-D Wasserstein distance per feature, which achieves 0.0315. This small quantitative distance confirms the qualitative findings from Fig. 2, validating that the learned feature distributions are similar to the real data.

In tandem, these quantitative and qualitative results indicate that our co-evolutionary VFL-VAE, combined with an LDM, is able to learn the underlying data distribution to generate synthetic samples.

Conclusion

We introduced FedEvolve, a novel framework for privacy-preserving synthetic data generation in Vertical Federated Learning. Our core contribution is the replacement of traditional federated backpropagation with a **co-evolutionary memetic algorithm** to optimize a distributed Variational Autoencoder (VFL-VAE). This **gradient-free** approach directly addresses a fundamental security flaw in VFL by eliminating the sharing of gradients, thereby mitigating critical privacy risks and communication bottlenecks.

Our preliminary results on the Higgs dataset demonstrate the viability of this paradigm. The co-evolutionary process successfully converges, and the resulting VAE's latent space allows an LDM to generate synthetic data. This generated data not only captures the some of the statistical structure of the original, siloed data (as shown by low Wasserstein distances) but also retains significant downstream utility, achieving **81.36% of the real-data performance** in a TSTR task.

This work represents a promising step towards responsible and private generative AI for decentralized data. However, several critical avenues for future research remain. A primary direction is a **formal privacy analysis**. Although we eliminate gradient sharing, the transfer of latent vectors to the server still presents a potential information leakage surface. Investigating the integration of differential privacy (DP) mechanisms is a crucial next step for providing formal privacy guarantees. Furthermore, future work must explore the **fairness and bias** implications of this generation process, ensuring that the resulting synthetic data does not amplify existing biases from siloed sources. Finally, we plan to focus on the **scalability and computational efficiency** of the co-evolutionary approach, as well as **extensive benchmarking** against state-of-the-art gradient-based VFL generative models to fully understand its trade-offs.

Appendix

Complete Co-evolutionary VFL-VAE Training Algorithm

Algorithm 1: Co-evolutionary VFL-VAE Training

Initialize: Client population P_C, Server population P_S, number of generations G_{max}, committee size K, elitism count E.
Pre-train all client models in P_C as standalone VAEs on their local data.
Evaluate fitness for all individuals in P_C and P_S using randomly chosen partner committees.
for generation $g = 1$ to G_{max} **do**
  Update KLD weight β using an annealing schedule.
  Adapt mutation strength σ_m using a cosine decay schedule.
  $\{$*Evolve Server Population*$\}$
  Select top-K client sets from P_C as $C_{committee}$.
  Initialize next server generation P'_S with top-E elites from P_S.
  while $|P'_S| < |P_S|$ **do**
    Select parents s_1, s_2 from P_S via tournament selection.
    $s_{child} \leftarrow$ Crossover(s_1, s_2).
    $s_{child} \leftarrow$ Mutate(s_{child}, σ_m).
    $s_{child} \leftarrow$ LocalSearch$(s_{child}, C_{committee})$. $\{$Memetic step ((1+1)-ES)$\}$
    Add s_{child} to P'_S.
  end while
  $P_S \leftarrow P'_S$.
  Evaluate fitness for all individuals in P_S using $C_{committee}$.
  $\{$*Evolve Client Population*$\}$
  Select top-K servers from P_S as $S_{committee}$.
  Initialize next client generation P'_C with top-E elites from P_C.
  while $|P'_C| < |P_C|$ **do**
    Select parents c_1, c_2 from P_C via tournament selection.
    $c_{child} \leftarrow$ Crossover(c_1, c_2).
    $c_{child} \leftarrow$ Mutate(c_{child}, σ_m).
    $c_{child} \leftarrow$ LocalSearch$(c_{child}, S_{committee})$. $\{$Memetic step ((1+1)-ES)$\}$
    Add c_{child} to P'_C.
  end while
  $P_C \leftarrow P'_C$.
  Evaluate fitness for all individuals in P_C using $S_{committee}$.
end for
Return: Best client set from P_C and best server from P_S.

References

Cai, T. T.; and Ma, R. 2022. Theoretical Foundations of t-SNE for Visualizing High-Dimensional Clustered Data. arXiv:2105.07536.

Chang, W.; and Zhu, T. 2024. Gradient-based defense methods for data leakage in vertical federated learning. *Computers Security*, 139: 103744.

de Goede, M.; Cox, B.; and Decouchant, J. 2024. Training Diffusion Models with Federated Learning. arXiv:2406.12575.

Desell, T.; Anderson, D. P.; Magdon-Ismail, M.; Newberg, H.; Szymanski, B. K.; and Varela, C. A. 2010. An analysis of massively distributed evolutionary algorithms. In *IEEE Congress on Evolutionary Computation*, 1–8.

Kato, A.; Kojima, K.; Nomura, M.; and Ono, I. 2025. A Memetic Algorithm based on Variational Autoencoder for Black-Box Discrete Optimization with Epistasis among Parameters. arXiv:2504.21338.

Li, P.; Hao, J.; Tang, H.; Fu, X.; Zheng, Y.; and Tang, K. 2024. Bridging Evolutionary Algorithms and Reinforcement Learning: A Comprehensive Survey on Hybrid Algorithms. arXiv:2401.11963.

Liu, Y.; Kang, Y.; Zou, T.; Pu, Y.; He, Y.; Ye, X.; Ouyang, Y.; Zhang, Y.-Q.; and Yang, Q. 2024. Vertical Federated Learning: Concepts, Advances, and Challenges. *IEEE Transactions on Knowledge and Data Engineering*, 36(7): 3615–3634.

Madathil, N. T.; Dankar, F. K.; Gergely, M.; Belkacem, A. N.; and Alrabaee, S. 2025. Revolutionizing healthcare data analytics with federated learning: A comprehensive survey of applications, systems, and future directions. *Computational and Structural Biotechnology Journal*, 28: 217–238.

McInnes, L.; Healy, J.; and Melville, J. 2020. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv:1802.03426.

Parsons, R. J. 1998. Chapter 9 - Evolutionary approaches to computational biology. In Salzberg, S. L.; Searls, D. B.; and Kasif, S., eds., *Computational Methods in Molecular Biology*, volume 32 of *New Comprehensive Biochemistry*, 165–186. Elsevier.

Polato, M. 2021. Federated Variational Autoencoder for Collaborative Filtering. In *2021 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Rahman, R. 2025. Federated Learning: A Survey on Privacy-Preserving Collaborative Intelligence. arXiv:2504.17703.

Rashad, M.; Zhao, Z.; Decouchant, J.; and Chen, L. Y. 2024. TabVFL: Improving Latent Representation in Vertical Federated Learning. arXiv:2404.17990.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv:2112.10752.

Rosin, C. D.; and Belew, R. K. 1997. New methods for competitive coevolution. *Evol. Comput.*, 5(1): 1–29.

Ryan, C. 2003. Evolutionary Algorithms and Metaheuristics. In Meyers, R. A., ed., *Encyclopedia of Physical Science and Technology (Third Edition)*, 673–685. New York: Academic Press, third edition edition. ISBN 978-0-12-227410-7.

Shankar, A.; Brouwer, H.; Hai, R.; and Chen, L. 2024. SiloFuse: Cross-silo Synthetic Data Generation with Latent Tabular Diffusion Models. arXiv:2404.03299.

Stanley, K. O.; and Miikkulainen, R. 2002. Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, 10(2): 99–127.

Whiteson, D. 2014. HIGGS. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5V312.

Wu, Z.; Cao, L.; and Qi, L. 2025. eVAE: Evolutionary Variational Autoencoder. *IEEE Transactions on Neural Networks and Learning Systems*, 36(2): 3288–3299.

Zhao, Z.; Wu, H.; Van Moorsel, A.; and Chen, L. Y. 2025. Gtv: Generating tabular data via vertical federated learning. In *2025 55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 33–46. IEEE.

Ángel Morell, J.; Dahi, Z. A.; Chicano, F.; Luque, G.; and Alba, E. 2022. Optimising Communication Overhead in Federated Learning Using NSGA-II. arXiv:2204.02183.