# VIGIL: VARIANCE-INFORMED LATENT GUIDANCE FOR PRIVATE TIME SERIES GENERATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Hosting time series diffusion models on third-party servers offers ubiquitous client access. But sharing raw time series with servers raises privacy concerns. Prior art enables clients to work around by sharing latent embeddings instead. But such methods are inapplicable to conditional tasks such as imputation and forecasting, since the constraints are specified in the client's data space, which is inaccessible to the server. We propose VIGIL, a novel framework that enables privacy-preserving time-series diffusion by conditioning entirely in the latent space, with no observable time-series values leaving the client's premises. VIGIL has three phases: (1) *latent anchor optimisation* finds a plausible latent reference consistent with the observed data; (2) the *anchor confidence map* identifies the anchor positions corresponding to observed and missing positions in the data space; and (3) uses the anchor and the confidence map to steer generation via *latent-guided sampling* towards the constraints induced by the observed values. Results on five datasets spanning window lengths from 64 to 168 time steps demonstrate that VIGIL matches the performance of approaches that directly operate on raw data. Despite guiding in the latent space, VIGIL even improves on its data-space counterpart's similarity scores by 1.3% on average across 20 evaluation settings. Code available at `https://anonymous.4open.science/r/Synthetic-Time-Series-Generation/`.

## 1 INTRODUCTION

Organisations outsource their generative time series models to third parties to save on time, resources, or for convenience. This has popularised the practice of hosting generative models *"as-a-service"* (Hesamifard et al., 2018; Hanzlik et al., 2021) (e.g. time series foundation models (Goswami et al., 2024)). However, clients share their raw data with the server for training and inference in this setup, exposing sensitive information (Gilad-Bachrach et al., 2016).

Recent works offer an alternative; they use *autoencoders* to map raw values from the data-space into a *latent* space, giving *information-theoretic* privacy under black-box settings (Shankar et al., 2024) or softer guarantees under relaxed assumptions (Maeng et al., 2023; Hwang et al., 2026). However, these approaches only support *unconditional* generation. Of higher practical value is *conditional* generation, where clients generate data subject to *constraints*. Examples include tasks such as *imputation* and *forecasting*, where conditions are represented using binary masks indicating the positions of missing and observed values (Tashiro et al., 2021; Alcaraz & Strodthoff, 2022; Yuan & Qiao, 2023; Kollovieh et al., 2023; Shankar et al., 2025). However, extending latent models to conditional generation requires *translating* the data-space constraints into their latent counterparts. This is non-trivial for time series conditions like imputation, which have partially observed inputs: standard autoencoders do not preserve the relative positions between values in the data-space and the corresponding latent representation (Bengio et al., 2013). Hence, the server cannot determine which latent positions encode the observed values. This tension between privacy and conditional generation is illustrated in Fig. 1, where data-space conditioning exposes the client's observations, while latent-space generation preserves privacy but does not directly support conditioning. Our goal is to bridge these two settings by enabling conditional generation entirely in the latent space.

We propose VIGIL (**V**ariance-**I**nformed **G**uidance **I**n **L**atent space), a novel *latent-guidance diffusion* framework for privacy-preserving time series imputation. We consider a general setting where



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

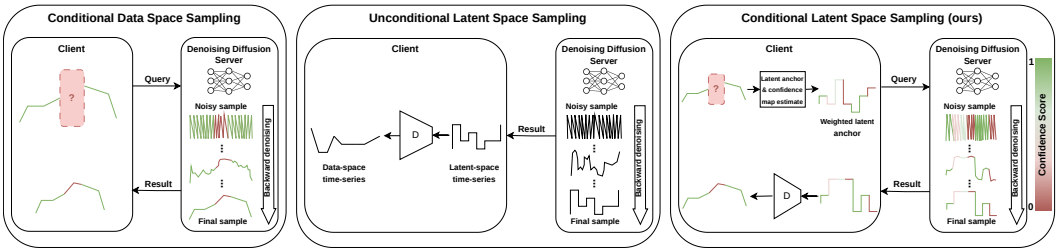


Figure 1: *Objective*: Outsource conditional time series generation without exposing the client's raw observations. *Left*: state-of-the-art data-space sampling exposes raw observations. *Middle*: latent-space sampling preserves anonymity, but is unconditional. *Right*: VIGIL enables conditional sampling and data anonymity using only locally constructed latent representations.

a client holds a pre-trained autoencoder, while a server hosts a latent diffusion model pre-trained on the autoencoder's embeddings. VIGIL enables conditional generation entirely in latent space and operates purely at *inference time*. The framework consists of three phases. First, *Latent Anchor Optimisation* uses the client's autoencoder to construct an *anchor*—a plausible latent representation consistent with the observed values in data space. Second, *Anchor Confidence Map Estimation* estimates the likelihood that each anchor position maps back to the observed values in data space, producing a map of latent-to-observation correspondence. The client communicates the anchor and confidence map to the server, where *Latent-Guided Sampling* uses *training-free guidance* (Kollovieh et al., 2023) to steer the pre-trained diffusion model towards the reliable anchor components. This enforces the client's observations without exposing the raw data or imputed values, while supporting arbitrary imputation granularities. We summarise our contributions as follows:

**Privacy-Preserving Imputation and Forecasting Framework.** We introduce VIGIL, a novel framework for time series imputation and forecasting using server-hosted diffusion models. VIGIL constrains generation purely in the latent space, keeping raw data within the client's premises.

**Variance-informed Latent Guidance.** We propose a *training-free*, inference-time guidance method composed of three phases: (i) *Latent Anchor Optimisation* produces observation-consistent latents via client-side optimisation; (ii) *Anchor Confidence Map Estimation* leverages variance-based estimation to translate arbitrary data-space masks into latent weight vectors; and (iii) *Latent-Guided Sampling*, which steers the diffusion model towards the constraint purely in the latent space.

We evaluate VIGIL on 5 benchmarks and show that it achieves imputation quality matching (or exceeding) centralised methods that observe the raw data directly. VIGIL also demonstrates robustness across arbitrary missingness scenarios and latent mask identification techniques.

## 2 Background and Related Work

**Diffusion models for time series**. Diffusion models synthesise data by reversing an iterative corruption process (Ho et al., 2020). Let $\mathbf{x}_0 \in \mathbb{R}^{T\times C}$ denote a sampled time window with $T$ timesteps and $C$ features. The *forward* (noising) process over $N_{\text{diff}}$ steps produces increasingly noisy samples $\{\mathbf{x}_t\}_{t=1}^{N_{\text{diff}}}$. The noise at each step is controlled by a schedule, expressed through the cumulative product $\bar{\alpha}_t = \prod_{j=1}^{t} \alpha_j$, where $\bar{\alpha}_t$ monotonically decreases from 1 toward 0. Under the standard Gaussian forward process (Ho et al., 2020), the marginal distribution at step $t$ is given by

$$p(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}\right). \tag{1}$$

A neural network learns the *reverse* process, progressively denoising $\mathbf{x}_{N_{\text{diff}}}$ to recover temporal sequences from the data distribution (Alcaraz & Strodthoff, 2022; Yuan & Qiao, 2023). The reverse process can additionally be *conditioned* on available information during generation, parametrised as $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{c})$, where $\mathbf{c}$ represents the conditioning information and $\theta$ parametrises the neural network. This conditional formulation is useful for time-series *imputation*, where the partially observed sequences provide valuable context to infer the missing values. There are two main strategies for conditional time-series generation: (i) learning the conditional distribution directly during *training*, or (ii) enforcing the conditions during *inference* via *guidance* (Kollovieh et al., 2023; Shankar et al., 2025). Details on conditioning under each strategy are given in Appendix C.

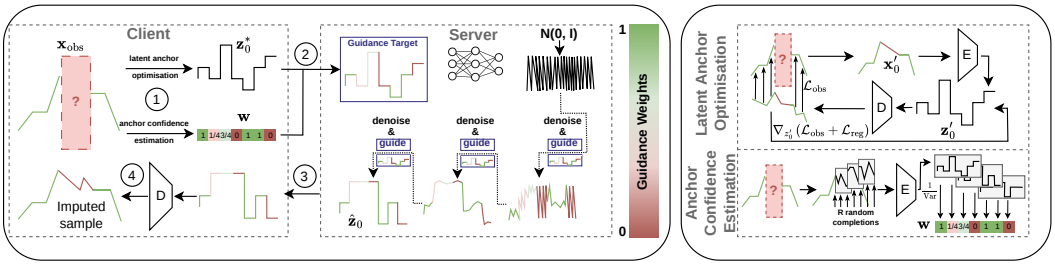


Figure 2: Overview of VIGIL. (1) The client optimises a latent anchor $\mathbf{z}_0^*$ and a confidence map $\mathbf{w}$ that assigns weights to its positions. On the right, the anchor is obtained by optimising a dummy $\mathbf{z}_0'$ to match the observations; the confidence map is estimated from the latent variance across randomised completions of $\mathbf{x}_{\text{obs}}$. (2) The server guides the reverse diffusion process toward the anchor weighted by the confidence map. (3) The final latent output is returned to the client. (4) The client decodes it locally to obtain the imputed time series.

**Latent diffusion models for privacy.** These approaches use an encoder to first map an input to a latent representation $\mathbf{z}_0 = E(\mathbf{x}_0)$, after which the diffusion process is performed over $\mathbf{z}_0$, which a decoder $D$ then maps back to the data space. This separation is useful for protecting privacy. The client keeps the encoder and decoder local, while the diffusion model trains on latent representations (at the server) without seeing the client's raw data (Shankar et al., 2024; Irureta et al., 2026).

But extending this paradigm to conditional generation is non-trivial, since the conditions are defined in the data space while the diffusion model operates in the latent space. Related work on latent diffusion for image inpainting suggests two directions. The first exploits the spatial locality of autoencoders, assuming that positions that are close in the data space remain close in the latent representation (Corneanu et al., 2024). But this assumption does not generalise to encoders that do not preserve spatial or temporal locality. The second generalises across architectures—latent representations are decoded into the data space, where the constraint is evaluated, and the resulting gradients are backpropagated to steer the latent sample (Wang & Tartakovsky, 2026; He et al., 2024). But this process requires communication between the client and server at every guidance step, since the decoder resides on the client while the latents are at the server, making it prohibitively expensive.

## 3 VIGIL

We consider a client with a partially observed time series who wishes to impute its missing values using a (latent) diffusion model hosted by a server. The raw time series must remain local to the client. This setting imposes three requirements: *(i) privacy*, by sharing only latent representations with the server through a pre-trained autoencoder; *(ii) validity*, by ensuring that generated samples match the client's values at the observed positions; and *(iii) scalability*, by minimising the communication exchanges between the client and server.

**Threat model.** We assume a *semi-honest* server that follows the prescribed protocol but attempts to infer the client's raw data from any information shared with it (Hardy et al., 2017). We also assume that the client's autoencoder is a black box; its architecture and parameters are unknown to the server. Under these assumptions, prior work establishes information-theoretic privacy guarantees when sharing latent representations with the server (Shankar et al., 2024).

Fig. 2 is an overview of VIGIL. Our key idea is to convert the client's data-space observations into latent-space constraints that the server can enforce directly. First, the client performs *latent anchor optimisation* (Sec. 3.1) which constructs a plausible latent sample whose decoded representation matches the client's values at the observed positions. We refer to this latent representation as *anchor*. However, matching the observations does not guarantee that the entire anchor decodes to a realistic time series. Therefore, the client performs *anchor confidence map estimation* (Sec. 3.2) which estimates the reliability of each latent position with respect to the observed data, much like a latent analogue of the observation mask, indicating which latent values of the anchor represent the observations. Finally, the client shares the anchor and its confidence map with the server which combines them to perform *latent guidance* (Sec. 3.3). This enables guiding the diffusion process

toward the observed data regions while avoiding unreliable parts of the anchor. The guidance is *training-free* and operates entirely in the latent space, minimising communication with the client during generation while providing privacy guarantees.

### 3.1 Latent Anchor Optimisation (Client-side)

We first optimise a *latent anchor* whose decoded time series is consistent with the client's observed values. We design an iterative optimisation algorithm, starting with a *dummy* initialisation, and iteratively update it until its reconstruction matches the client's observed data.

Consider a time window of $T$ timesteps and $C$ features, $\mathbf{x}_0 \in \mathbb{R}^{T\times C}$, and let $(E, D)$ denote the client's pre-trained autoencoder, which maps the input to a latent representation $\mathbf{z}_0 = E(\mathbf{x}_0) \in \mathbb{R}^{L_z \times D_z}$, where $L_z \times D_z$ denotes the time series' shape in the latent embedding. The sequence can be decoded as $\hat{\mathbf{x}}_0 = D(\mathbf{z}_0)$. When $\mathbf{x}_0$ is only partially observed, we define the set of observed values as $\mathbf{x}_{\text{obs}} = \mathbf{m} \odot \mathbf{x}_0; \mathbf{m} \in \{0, 1\}^{T\times C}$, where $\mathbf{m}$ is a binary-valued observation mask indicating the positions of the observed values.

**Dummy initialisation via linear interpolation.** We denote the latent anchor as $\mathbf{z}_0$, whose decoded sequence matches the client's observed values, i.e., $\mathbf{x}_{\text{obs}} = \mathbf{m} \odot D(\mathbf{z}_0)$. As a reasonable starting point for $\mathbf{z}'_0$, we first fill the gaps in $\mathbf{x}_{\text{obs}}$ via *linear interpolation* to get a smooth, low-bias estimate of the missing entries. Specifically, for each feature $c$ and every missing timestep $s$ lying between its nearest observed timesteps $s_1 < s < s_2$ (i.e. $\mathbf{m}_{s_1,c} = \mathbf{m}_{s_2,c} = 1$ and $\mathbf{m}_{s,c} = 0$), we set

$$\mathbf{x}'_{0,s,c} = \mathbf{x}_{0,s_1,c} + (\mathbf{x}_{0,s_2,c} - \mathbf{x}_{0,s_1,c}) \, \frac{s - s_1}{s_2 - s_1},$$

with missing entries outside the observed interval boundaries set to the value of the nearest observed boundary position. The resulting $\mathbf{x}'_0$ is a complete sequence used to provide a starting point for anchor optimisation. Then, the client encodes $\mathbf{x}'_0$ to initialise the latent as $\mathbf{z}'_0 = E(\mathbf{x}'_0)$.

**Latent anchor optimisation.** The client aims to find the optimised latent anchor $\mathbf{z}^*_0$ whose decoded time series matches the client's observed values. To achieve this, we minimise an *observation loss* $\mathcal{L}_{\text{obs}}$ which measures the gap between the observations and the decoded latent anchor.

$$\mathcal{L}_{\text{obs}}(\mathbf{z}'_0, \mathbf{m}, \mathbf{x}_{\text{obs}}) = \frac{\|\mathbf{m} \odot (D(\mathbf{z}'_0) - \mathbf{x}_{\text{obs}})\|_F^2}{\sum_{s,c} \mathbf{m}_{s,c} + \varepsilon} + \mathcal{L}_{\text{reg}}(\mathbf{z}'_0, \mathbf{m}), \tag{2}$$

where $\varepsilon > 0$ ensures numerical stability. The client iteratively refines $\mathbf{z}'_0$ while keeping the autoencoder parameters frozen. At each iteration, $D(\mathbf{z}'_0)$ is used to evaluate the observation loss (Eq. 2), whose gradient $\nabla_{\mathbf{z}'_0} \mathcal{L}_{\text{obs}}$ is backpropagated through the frozen decoder to update $\mathbf{z}'_0$. The observation loss enforces consistency with the observed values.

$$\mathbf{z}^*_0 \leftarrow \arg\min_{\mathbf{z}'_0} [\mathcal{L}_{\text{obs}}(\mathbf{z}'_0, \mathbf{m}, \mathbf{x}_{\text{obs}})] \tag{3}$$

Eq. 2, uses a regularisation term $\mathcal{L}_{\text{reg}}$ to constrain the unobserved regions to remain plausible, preventing unrealistic variations during optimisation, which we detail in Appendix D.1.

Using $\mathbf{z}^*_0$ as the latent guidance target provides a practical approximation to the oracle anchor obtained by encoding the complete sequence. As shown in Appendix H.3, the optimised anchor is particularly beneficial in more challenging settings while remaining close to the oracle overall.

### 3.2 Anchor Confidence Map Estimation (Client-side)

The latent $\mathbf{z}^*_0$ (Sec. 3.1) provides an anchor for the observed values, but it should not be treated as a target for the entire latent representation. It is only *one* possible representation consistent with the client's observations. Multiple plausible anchor-completions can map to the same observations, not all of which are equally realistic. *How can we infer which positions of the anchor reliably correspond to the observed values in the data space?*

We answer this by measuring the *sensitivity* of each anchor position to perturbations of the missing entries, connecting our confidence measure to the general principle of variance-based sensitivity

analysis (Sobol', 2001; Kucherenko et al., 2012). We generate multiple completions by injecting random noise into the missing positions while keeping the observed values fixed, and encode each completion into the latent space. Since the observed values remain unchanged across completions, any variation in a latent position must be induced by the injected noise at the missing positions. A latent position that changes substantially is therefore sensitive to the missing entries and should not be used to enforce the observed constraint. Conversely, a latent position that remains stable across perturbations is less dependent on the missing entries and more reliably reflects the observed values. We therefore assign greater confidence to insensitive latent positions and use this information to guide diffusion towards the anchor.

The client completes the missing input values by injecting random Gaussian noise while keeping the observed values fixed. For each random trial $r \in [1, R]$, we define $\mathbf{x}_0^{(r)} = \mathbf{x}_{\text{obs}} + (1 - \mathbf{m}) \odot \boldsymbol{\xi}^{(r)}$, where $\boldsymbol{\xi}^{(r)} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$. We then encode each completion as $\mathbf{z}_0^{(r)} = E(\mathbf{x}_0^{(r)})$. For every position in the latent representation, we compute the element-wise sample variance as

$$\boldsymbol{\sigma}_{\ell,d} = \frac{1}{R-1} \sum_{r=1}^{R} \left( \mathbf{z}_{0,\ell,d}^{(r)} - \bar{\boldsymbol{\mu}}_{\ell,d} \right)^2, \qquad \bar{\boldsymbol{\mu}}_{\ell,d} = \frac{1}{R} \sum_{r=1}^{R} \mathbf{z}_{0,\ell,d}^{(r)}, \quad \forall (\ell, d) \in [L_z] \times [D_z].$$

Proposition 3.1 below formalises our intuition, showing that the variance of each latent position directly measures its sensitivity to the missing values. Since $\mathbf{x}_{\text{obs}}$ and $\mathbf{m}$ remain fixed across stochastic completions, any variation in $\mathbf{Z}_{\ell,d}$ is induced by changes to the missing entries. Equivalently, $\boldsymbol{\sigma}_{\ell,d}$ quantifies the disagreement of latent component $(\ell, d)$ across alternative completions. Thus, low variance identifies components that are insensitive to the unspecified values and can be assigned higher confidence. We provide the proof and further discussion in Appendix E.

**Proposition 3.1** (Latent sensitivity under stochastic completion)**.** *Fix $\mathbf{x}_{\text{obs}}$ and $\mathbf{m}$, and let $Q$ denote the reference distribution used to sample the unspecified entries. Define $\mathbf{Z} = E(\mathbf{x}_{\text{obs}} + (1 - \mathbf{m}) \odot \boldsymbol{\xi})$, where $\boldsymbol{\xi} \sim Q$ and $\mathbb{E}[\mathbf{Z}_{\ell,d}^2] < \infty$ for all $(\ell, d)$. For $R$ independent stochastic completions, the empirical variance $\boldsymbol{\sigma}_{\ell,d}$ satisfies $\mathbb{E}[\boldsymbol{\sigma}_{\ell,d}] = \text{Var}_Q(\mathbf{Z}_{\ell,d})$, i.e., $\boldsymbol{\sigma}_{\ell,d}$ is an unbiased estimator of the variance induced by varying the unspecified entries under $Q$. Moreover,*

$$\boldsymbol{\sigma}_{\ell,d} = \frac{1}{R(R-1)} \sum_{r<s} \left( \mathbf{z}_{0,\ell,d}^{(r)} - \mathbf{z}_{0,\ell,d}^{(s)} \right)^2.$$

In our implementation, we use $Q = \mathcal{N}(\mathbf{0}, \mathbf{I})$ as a standardised reference distribution for probing the encoder, rather than as an assumption on the true missing-data distribution.

**Anchor confidence mapping.** The client uses the above variance-based sensitivity analysis to assign a confidence map of weights $\mathbf{w}$ to each position in the latent anchor—specifically, we use the normalised inverse variance to assign larger confidence values to more reliable latent positions.

$$\mathbf{w}_{\ell,d} = \delta + (1 - \delta) \frac{(\boldsymbol{\sigma}_{\ell,d} + \varepsilon)^{-1}}{\max_{\ell',d'} (\boldsymbol{\sigma}_{\ell',d'} + \varepsilon)^{-1}} \tag{4}$$

Here $\delta$ is a small confidence floor and $\varepsilon > 0$ ensures numerical stability. To emphasise the most reliable latent positions, we retain a fraction of confidence scores and attenuate the remainder:

$$\mathbf{w}_{\ell,d} \leftarrow \begin{cases} \mathbf{w}_{\ell,d}, & \mathbf{w}_{\ell,d} \geq \kappa, \\ \gamma \mathbf{w}_{\ell,d}, & \mathbf{w}_{\ell,d} < \kappa. \end{cases}$$

where $\kappa$ is the $(1 - p)$-th quantile and $\gamma \ll 1$ strongly downweights uncertain latent positions.

We provide the algorithm of the anchor confidence mapping in Appendix D.3. We validate this strategy against three alternative confidence map constructions in Sec. 4.3, and show that variance-based selection achieves the best overall performance across datasets and missingness scenarios.

### 3.3 Latent-Guided Diffusion Sampling (Server-side)

**Training-free guidance.** Using the optimised anchor $\mathbf{z}_0^*$ and confidence map $\mathbf{w}$, the server needs to guide the reverse diffusion process. Algorithm 1 shows how this guidance is performed at each reverse step during sampling. Starting from a fully noisy latent, the model first predicts the noise and uses it to estimate the corresponding clean latent (line 4):

$$\hat{\mathbf{z}}_{0|t} = \frac{\mathbf{z}_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(\mathbf{z}_t, t)}{\sqrt{\bar{\alpha}_t}} \tag{5}$$

with $\epsilon_\theta$ being the neural network estimate of the noise at step $t$. We then guide the clean latent (lines 8- 12) towards $\mathbf{z}_0^*$ according to $\mathbf{w}$ as described below.

**Algorithm 1** Latent-Guided Reverse Diffusion

**Require:** $\epsilon_\theta$, $\mathbf{z}_0^*$, $\mathbf{w}$, $N_{\text{diff}}$, $\eta_0$, $K_0$, $\varepsilon$
**Ensure:** $\hat{\mathbf{z}}_0$
1: $\mathbf{z}_{N_{\text{diff}}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = N_{\text{diff}}$ to 1 **do**
3: $\quad\hat{\boldsymbol{\epsilon}}_t \leftarrow \epsilon_\theta(\mathbf{z}_t, t)$
4: $\quad$Compute $\hat{\mathbf{z}}_{0|t}$ using Eq. 5
5: $\quad\tau_t \leftarrow (t-1)/(N_{\text{diff}}-1)$
6: $\quad\eta(t) \leftarrow \max(\eta_0(1-\tau_t), 0.1\eta_0)$
7: $\quad K(t) \leftarrow \max(2K_0 - \lfloor 2\tau_t K_0 \rfloor, 1)$
8: $\quad$**for** $k = 0$ to $K(t) - 1$ **do**
9: $\quad\quad$Compute $\mathcal{L}_{\text{guide}}$ using Eq. 6
10: $\quad\quad\mathbf{g} \leftarrow \nabla_{\hat{\mathbf{z}}_{0|t}} \mathcal{L}_{\text{guide}}$
11: $\quad\quad\hat{\mathbf{z}}_{0|t} \leftarrow \hat{\mathbf{z}}_{0|t} - \eta(t)\,\frac{\mathbf{g}}{\|\mathbf{g}\|_F + \varepsilon}$
12: $\quad$**end for**
13: $\quad\mathbf{z}_{t-1} \leftarrow \text{ReverseStep}(\hat{\mathbf{z}}_{0|t}, \hat{\boldsymbol{\epsilon}}_t, t)$ (Eq. 9)
14: **end for**
15: **return** $\hat{\mathbf{z}}_0$

**Dynamic guidance-strength.** For guidance, we follow an adapted version of He et al. (2024), who, at each reverse-diffusion step, correct the clean-latent estimate $\hat{\mathbf{z}}_{0|t}$ by taking a gradient step on the guidance objective before using the corrected estimate in the reverse-diffusion update. We deviate from that approach in two aspects. First, the original method requires repeated decoding into data space; we eliminate this by computing the guidance loss entirely in the latent space. Since this provides a less direct constraint on the observations, we compensate by performing multiple guidance corrections at each reverse diffusion step. Second, we adapt the guidance strength $\eta(t)$ (line 6) and the number of corrections $K(t)$ (line 7) throughout sampling.

In particular, early in the denoising process, the clean-latent estimate is still highly uncertain due to the large amount of diffusion noise; applying strong guidance at this stage may distort the denoising trajectory, leading to inadequate results. We therefore use weaker and fewer refinement steps early in sampling, and progressively increase both as the clean-latent estimate becomes more reliable.

At each denoising step, the server iteratively corrects the predicted clean latent estimate for $K(t)$ iterations (lines 10, 11) by taking normalised gradient steps on the latent guidance objective given in Eq. 6. After these refinement iterations, the resulting clean-latent estimate is used to perform a single standard unconditional reverse-diffusion step (line 13) using the corrected clean estimate and predicted noise, as described in Appendix C.3. Repeating this predict–guide–reverse procedure yields the final guided latent $\hat{\mathbf{z}}_0$.

Despite the multiple refinement iterations, these updates are inexpensive relative to denoising because they require neither an additional denoiser evaluation nor decoding to the data space. Appendix H.4 reinforces this by showing that the complete method is considerably faster than the other conditional inference baselines which operate in the more expensive data space.

**Confidence-weighted latent guidance objective.** In our setting, the server uses the confidence map $\mathbf{w}$ to suppress positions that depend strongly on the non-observed entries, while guidance toward $\mathbf{z}_0^*$ enforces consistency with the observations. The objective of Eq. 6 penalises deviations from this optimised anchor $\mathbf{z}_0^*$ in proportion to the confidence assigned to each latent component. Components with high confidence are therefore pulled more strongly toward $\mathbf{z}*_0$, while low-confidence components are constrained less strongly, allowing the diffusion prior to determine their values. The denominator converts the weighted sum into a weighted average, making the loss value comparable across confidence maps with different total weight. Finally, $\varepsilon > 0$ ensures numerical stability.

$$\mathcal{L}_{\text{guide}}\left(\hat{\mathbf{z}}_{0|t}, \mathbf{z}_0^*, \mathbf{w}\right) = \frac{\sum_{\ell=1}^{L_z}\sum_{d=1}^{D_z} \mathbf{w}_{\ell,d}\left(\hat{\mathbf{z}}_{0|t,\ell,d} - \mathbf{z}^*_{0,\ell,d}\right)^2}{(\sum_{\ell=1}^{L_z}\sum_{d=1}^{D_z} \mathbf{w}_{\ell,d}) + \varepsilon} \tag{6}$$

We validate the effectiveness of this guidance in Sec. 4.3, where guided sampling substantially outperforms the no-guidance variant. Appendix H.2 further compares against applying guidance

by repeatedly decoding intermediate latents on the client and returning corrections to the server, showing that our communication-free sampling remains competitive without this expensive iterative client–server interaction.

# 4 EXPERIMENTS

## 4.1 EXPERIMENTAL SETUP

**Datasets.** We evaluate VIGIL on five multivariate time-series datasets spanning diverse application domains: Appliances Energy Prediction (Candanedo, 2017), Beijing Air Quality (Chen, 2017), the Tennessee Eastman Process (TEP) (Rieth et al., 2017), Human Activity Recognition (HAR) (Reyes-Ortiz et al., 2013), and Metro Interstate Traffic Volume (Hogue, 2019). All datasets are chronologically split into training and test sets, standardised using training statistics, and segmented into overlapping sliding windows. To evaluate robustness across different temporal contexts, we use window lengths of 168 for Appliances, Air Quality, and Metro, 128 for HAR, and 64 for TEP. Further preprocessing details are provided in Appendix D.5. Following the **missingness scenarios** of Alcaraz & Strodthoff (2022), we consider random missingness, channel-wise contiguous blocks, blackout intervals shared across channels, and forecasting, where the final portion of the sequence is missing. Our primary **metric** is the continuous ranked probability score (CRPS), which evaluates the quality of the full predictive distribution and accounts for both accuracy and uncertainty; lower values are better. We additionally report mean squared error (MSE) in Appendix H.1.

**Baselines.** We compare against four representative diffusion-based time-series imputation methods: TSDiff (Kollovieh et al., 2023), Diffusion-TS (Yuan & Qiao, 2023), CSDI (Tashiro et al., 2021), and SSSD-S4 (Alcaraz & Strodthoff, 2022). TSDiff and Diffusion-TS are unconditional diffusion models that perform imputation at inference time, making them the closest comparison to our method, while CSDI and SSSD-S4 are trained explicitly for conditional imputation. We additionally consider two variants of our method that isolate the contributions of latent-space modelling and latent-space guidance. *Data Guidance* performs diffusion and observation guidance directly in the data space, providing a comparison for the effect of training the diffusion model in the latent space. *Spinning Decoder* is included as an ablation to assess the alternative of applying observation guidance through the decoder rather than directly in the latent space; its formulation and associated substantial computational and communication costs are discussed in Appendix H.2. Unlike our method, all baselines except Spinning Decoder operate centrally and perform diffusion directly in the original data space. Further details on the baselines are provided in Appendix D.6.

**Implementation and evaluation details.** We use a Transformer autoencoder to compress each multivariate time series into a latent representation of $L_z = 16$ tokens with embedding dimension $D_z = 128$, and train an unconditional TSDiff-based diffusion model directly in this latent space as our backbone. The diffusion backbone uses S4 layers augmented with Transformer-based feature mixing and is trained with 200 diffusion steps. We evaluate each method on 500 test sequences with $50\%$ missingness, generating 10 imputations per sequence and reporting results over five inference seeds. To keep the comparison consistent across datasets, we use a single set of hyperparameters for our method and do not tune the autoencoder architecture or guidance configuration separately for each dataset. Further architectural, optimisation, evaluation, metric, and hardware details are provided in Appendix D.4.

## 4.2 MAIN RESULTS

Fig. 3 provides example tasks showing that the generated trajectory remains well aligned with the ground truth in both the observed and missing regions, while additional qualitative examples across datasets and missingness settings are provided in Appendix H.5. Table 1 compares VIGIL against both inference-time and conditionally trained diffusion baselines. Our latent-space approach is competitive with the centralised data-space inference methods, achieving macro-averaged relative CRPS reductions of $1.3\%$, $26.2\%$, and $5.6\%$ over Data Guidance, TSDiff, and Diffusion-TS, respectively, across the 20 evaluation settings. The comparison with Data Guidance in particular shows that moving both the diffusion process and observation guidance to the latent space can provide substantial gains on some datasets. The main exception is TEP, which has substantially more features per

Table 1: Mean CRPS over five inference seeds under $50\%$ block, forecasting, random, and blackout missingness; standard deviations across seeds are reported in parentheses in units of $10^{-3}$. Bold and underlined values denote the best and second-best mean results, respectively, among the conditional inference methods; rankings are based on the unrounded means.

| | | **Conditional Inference** | | | | **Conditional Training** | |
|---|---|---|---|---|---|---|---|
| | | **Latent-Space Diffusion** | **Data-Space Diffusion** | | | **Data-Space Diffusion** | |
| **Dataset** | **Scenario** | **VIGIL (Ours)** | **Data Guidance (Ours)** | **TSDiff** | **Diff.-TS** | **CSDI** | **SSSD-S4** |
| | | Mean CRPS (SD $\times 10^{-3}$) | | | | | |
| Appliances | Block | 0.43 (0.64) | **0.34 (0.33)** | 0.55 (0.43) | 0.63 (0.04) | 0.22 (0.39) | 0.42 (0.28) |
| | Forecast | **0.51 (0.78)** | 0.53 (2.18) | 0.59 (0.67) | 0.70 (0.11) | 0.35 (0.90) | 0.69 (0.68) |
| | Random | 0.25 (0.48) | **0.16 (0.06)** | 0.51 (0.36) | 0.53 (0.01) | 0.02 (0.02) | 0.19 (0.04) |
| | Blackout | 0.48 (0.80) | **0.47 (2.20)** | 0.59 (0.85) | 0.67 (0.06) | 0.26 (0.65) | 0.63 (0.49) |
| HAR | Block | 0.68 (0.39) | 0.66 (2.19) | 0.81 (1.34) | **0.57 (1.09)** | 0.42 (1.00) | 0.38 (0.95) |
| | Forecast | 0.69 (1.61) | 0.89 (2.10) | 0.97 (1.39) | **0.68 (0.61)** | 0.56 (0.85) | 0.56 (0.88) |
| | Random | 0.32 (0.47) | **0.29 (0.64)** | 0.66 (1.15) | 0.42 (0.25) | 0.19 (0.34) | 0.25 (0.33) |
| | Blackout | 0.69 (1.37) | 0.82 (1.89) | 0.88 (2.73) | **0.65 (0.86)** | 0.53 (0.67) | 0.53 (0.65) |
| Metro | Block | **0.56 (1.15)** | 0.99 (2.86) | 1.05 (1.76) | 0.57 (0.66) | 0.42 (1.07) | 0.49 (2.19) |
| | Forecast | **0.58 (0.79)** | 1.11 (1.76) | 1.07 (5.25) | 0.67 (1.20) | 0.54 (3.46) | 0.58 (1.54) |
| | Random | 0.30 (0.98) | 0.57 (0.85) | 0.89 (2.44) | **0.28 (0.17)** | 0.13 (0.37) | 0.22 (0.27) |
| | Blackout | **0.53 (1.71)** | 1.03 (2.09) | 1.05 (1.43) | 0.60 (0.99) | 0.47 (1.73) | 0.54 (0.80) |
| TEP | Block | 0.76 (0.35) | **0.54 (0.57)** | 0.66 (0.59) | 0.63 (0.17) | 0.51 (0.40) | 0.67 (0.51) |
| | Forecast | 0.82 (1.08) | 0.77 (1.73) | 0.77 (0.82) | **0.75 (0.16)** | 0.70 (0.55) | 0.95 (0.93) |
| | Random | 0.74 (1.30) | **0.41 (0.29)** | 0.60 (0.43) | 0.56 (0.24) | 0.38 (0.20) | 0.84 (0.69) |
| | Blackout | 0.79 (0.88) | **0.71 (0.69)** | 0.74 (1.22) | 0.72 (0.38) | 0.65 (0.65) | 0.94 (0.58) |
| Air Quality | Block | 0.45 (0.67) | 0.48 (1.06) | 0.65 (1.18) | **0.44 (0.35)** | 0.29 (0.36) | 0.41 (1.04) |
| | Forecast | **0.52 (2.69)** | 0.62 (3.79) | 0.75 (1.86) | 0.57 (0.90) | 0.48 (0.97) | 0.57 (1.78) |
| | Random | 0.28 (0.51) | **0.27 (0.34)** | 0.58 (0.54) | 0.33 (0.07) | 0.13 (0.16) | 0.26 (0.24) |
| | Blackout | **0.50 (1.21)** | 0.59 (2.02) | 0.73 (2.43) | 0.53 (0.44) | 0.45 (2.29) | 0.56 (2.23) |

timestep than the other datasets and may therefore require a more demanding compression of cross-variable information into the fixed latent representation. We discuss this further in Appendix F.

The conditionally trained methods provide a stronger task-specific reference, since they are optimised explicitly for imputation using observed values as conditioning information. As expected, they generally achieve lower CRPS. CSDI is the stronger of the two, with VIGIL performing $105.8\%$ worse in macro-averaged relative CRPS. This aggregate gap is strongly influenced by the random-missingness settings, where CSDI attains particularly low CRPS, especially on Appliances. In contrast, the gap to SSSD-S4 is much smaller: VIGIL performs only $8.2\%$ worse, while matching or outperforming it in 9 out of the 20 settings. This is notable because SSSD-S4 is trained specifically for conditional imputation, whereas VIGIL uses an unconditional diffusion model and introduces conditioning only at inference time.

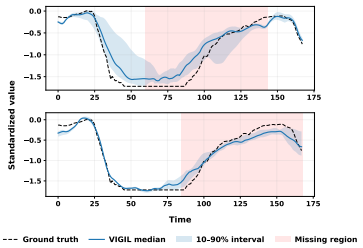


Figure 3: Qualitative VIGIL imputations on Appliances under $50\%$ block and forecasting missingness.

Overall, these results suggest that our latent-space inference can recover much of the performance of centralised data-space approaches without requiring dataset-specific tuning or expensive conditional diffusion training. Moreover, as shown in Table 8 of Appendix H.4, the diffusion cost remains essentially unchanged across datasets with different window lengths, with only minor variation in the lightweight client-side construction phase, whereas the data-space inference baselines exhibit substantially larger runtime variation. This suggests favourable scalability with respect to sequence length, as the dominant computational cost is decoupled from the original temporal resolution.

### 4.3 ANCHOR CONFIDENCE MAPPING

Table 2 evaluates different confidence mapping mechanisms. Here, variance is our proposed strategy. Among the **baselines**, *interpolation* projects the observation mask directly into the latent resolution and, similarly, retains its top $p\%$ components. *Random guidance* activates the same proportion of components at uniformly sampled locations. Finally, *Full guidance* applies the anchor loss to ev-

Table 2: Mean CRPS over five seeds and four $50\%$ missingness scenarios; std. deviations are reported in parentheses in units of $10^{-3}$. Bold and underlined values denote the best and second-best mean results within each scenario, respectively; rankings are based on the unrounded means.

| Dataset | Scenario | Variance (Ours) | Interpolation | Random | Full Guidance | No Guidance |
|---|---|---|---|---|---|---|
| Appliances | Block | 0.43 (0.64) | **0.35 (0.14)** | 0.44 (1.92) | 0.38 (0.02) | 0.88 (3.82) |
| | Forecast | **0.51 (0.78)** | 0.51 (0.32) | 0.54 (0.94) | 0.63 (0.02) | 0.88 (3.33) |
| | Random | 0.25 (0.48) | 0.24 (0.17) | 0.37 (1.45) | **0.09 (0.01)** | 0.88 (3.64) |
| | Blackout | 0.48 (0.80) | **0.43 (0.38)** | 0.47 (1.26) | 0.50 (0.02) | 0.88 (4.18) |
| HAR | Block | **0.68 (0.39)** | 0.73 (0.39) | 0.71 (2.23) | 0.76 (0.02) | 0.93 (4.78) |
| | Forecast | **0.69 (1.61)** | 0.74 (0.59) | 0.83 (2.54) | 0.97 (0.02) | 0.93 (4.41) |
| | Random | 0.32 (0.47) | 0.34 (0.25) | 0.44 (1.04) | **0.30 (0.03)** | 0.93 (4.55) |
| | Blackout | **0.69 (1.37)** | 0.72 (1.17) | 0.78 (1.39) | 0.89 (0.02) | 0.93 (4.46) |
| Metro | Block | **0.56 (1.15)** | 0.68 (0.86) | 0.68 (1.61) | 0.73 (0.03) | 0.75 (2.06) |
| | Forecast | **0.58 (0.79)** | 0.67 (1.87) | 0.77 (2.33) | 0.84 (0.04) | 0.75 (2.32) |
| | Random | 0.30 (0.98) | 0.26 (0.45) | 0.28 (1.41) | **0.22 (0.02)** | 0.75 (2.64) |
| | Blackout | **0.53 (1.71)** | 0.61 (1.47) | 0.68 (0.42) | 0.73 (0.04) | 0.75 (2.67) |
| TEP | Block | **0.76 (0.35)** | 0.94 (0.20) | 0.91 (1.52) | 1.01 (0.05) | 0.84 (1.33) |
| | Forecast | **0.82 (1.08)** | 0.90 (0.31) | 1.12 (3.02) | 1.32 (0.04) | 0.85 (1.21) |
| | Random | **0.74 (1.30)** | 0.84 (0.22) | 0.89 (0.64) | 0.95 (0.05) | 0.84 (1.12) |
| | Blackout | **0.79 (0.88)** | 0.95 (0.28) | 1.08 (2.62) | 1.24 (0.02) | 0.84 (1.05) |
| Air Quality | Block | **0.45 (0.67)** | 0.54 (0.12) | 0.58 (1.27) | 0.63 (0.03) | 0.77 (2.49) |
| | Forecast | **0.52 (2.69)** | 0.73 (1.01) | 0.72 (3.38) | 0.82 (0.05) | 0.77 (1.29) |
| | Random | 0.28 (0.51) | 0.29 (0.40) | 0.41 (1.50) | **0.22 (0.04)** | 0.77 (2.36) |
| | Blackout | **0.50 (1.21)** | 0.60 (0.53) | 0.64 (3.91) | 0.72 (0.04) | 0.77 (1.84) |

ery latent component, whereas *no guidance* disables the anchor loss entirely. For the variance and interpolation strategies, the remaining components are attenuated by the $\gamma$ factor.

Variance-based selection performs best overall on four of the five datasets and outperforms random selection in 18 of the 20 settings, showing that its gains are not explained simply by restricting guidance to a subset of the latent space. Its advantage is most pronounced under structured (not random) missingness, where parts of the optimised anchor can be unreliable and selectively attenuating sensitive latent components prevents the diffusion trajectory from being over-constrained.

Exceptions arise under random missingness and on the Appliances dataset. Under random missingness, observations are distributed throughout the sequence, which may make the optimised anchor more reliable than under long contiguous gaps. Consistent with this interpretation, under random missingness, full guidance performs best on four datasets; when most anchor components are already informative, selective guidance may unnecessarily attenuate useful information. Appliances shows a similar tendency across several scenarios, where interpolation or full guidance can outperform variance-based selection. This may indicate unusually reliable anchors or a better correspondence between the projected observation mask and the latent representation, as explained in Appendix G, although the ablation alone cannot distinguish these effects. Notably, full guidance deteriorates on forecasting, where constructing the anchor requires extrapolation beyond the observed region, which is consistent with selective guidance being more valuable when anchor reliability is heterogeneous.

## 5 Conclusion

We tackle the problem of performing time series imputation and forecasting on third-party servers without the client having to share raw data with the server. We propose VIGIL, the first framework enabling privacy-preserving, conditional time series diffusion entirely in the latent space. Within VIGIL a client converts its data-space conditions, like the observed time series values, into a latent space where the server generates via diffusion. By performing diffusion in the latent space, VIGIL keeps the observed time series values from being directly exposed to the server, inheriting the privacy guarantees established in prior work. Experiments show that VIGIL achieves competitive imputation performance compared to centralised approaches that directly access the raw data, while enabling conditional generation entirely in the latent space. Future work could study how the choice of autoencoder affects privacy, utility, and conditioning reliability. It would also be valuable to explore alternative anchor optimisation strategies and extend the framework beyond time series to other tasks such as image inpainting (also see Appendix B).

## 6 AI USE STATEMENT

Generative AI tools were used to assist with writing and presentation, literature retrieval, software implementation, and verification of the work. For the manuscript, AI was used to improve grammar, concision, readability, and the presentation of author-developed mathematical statements and proofs; it was not used to derive the mathematical claims, proofs, or their essential arguments. AI was also used to critically review author-developed hypotheses, experimental methodology, baseline configurations, and result interpretations for possible weaknesses, inconsistencies, or overclaims. For implementation, AI assisted with writing and modifying code according to author-specified requirements; generated code was manually inspected and tested, with important changes additionally checked using a separate LLM. AI was used to help format author-provided content into LaTeX tables, equations, figures, and other artifacts, and to provide feedback on figure layout and presentation. Following the authors' primary literature review, AI was used to search for potentially missed related work, relevant baselines, and literature that might qualify or contradict manuscript claims. Finally, AI was used for limited brainstorming on framing and acronym ideas. It was not used to originate the research topic, research gap, conceptual framework, datasets, or experimental results. All AI-assisted content was reviewed by the authors, who take responsibility for the final work.

## 7 REPRODUCIBILITY STATEMENT

We provide the full source code and experiment scripts used in this work in the anonymous repository linked with the submission. To support reproducibility, Appendix D reports the model architectures, training and inference hyperparameters, guidance configuration, evaluation protocol, dataset preprocessing, baseline implementations, and hardware and software environment. All experiments use fixed random seeds: the evaluation subset and missingness masks are generated once using seed 42, while stochastic guidance construction and diffusion sampling are repeated using inference seeds 1–5, and we report the mean and standard deviation across these runs. We also specify the complete preprocessing procedure for each dataset and the implementation details required to reproduce all baselines and ablations. The experimental hardware and software environment, including GPU, CPU, operating system, PyTorch, CUDA, and cuDNN versions, are reported.

## REFERENCES

Juan Lopez Alcaraz and Nils Strodthoff. Diffusion-based Time Series Imputation and Forecasting with Structured State Space Models. *Transactions on Machine Learning Research*, December 2022. ISSN 2835-8856.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, August 2013. doi: 10.1109/TPAMI.2013.50.

Luis Candanedo. Appliances Energy Prediction. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5VC8G.

Song Chen. Beijing Multi-Site Air Quality. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5RK5G.

Ciprian Corneanu, Raghudeep Gadde, and Aleix M Martinez. Latentpaint: Image inpainting in latent space with diffusion models. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4322–4331, 2024. doi: 10.1109/WACV57701.2024.00428.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing. CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 201–210. PMLR, June 2016.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A Family of Open Time-series Foundation Models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 16115–16152. PMLR, July 2024.

Lucjan Hanzlik, Yang Zhang, Kathrin Grosse, Ahmed Salem, Maximilian Augustin, Michael Backes, and Mario Fritz. MLCapsule: Guarded Offline Deployment of Machine Learning as a Service. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3300–3309, 2021.

Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677*, 2017.

Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, WeiHsiang Liao, Yuki Mitsufuji, Zico Kolter, Ruslan Salakhutdinov, and Stefano Ermon. Manifold Preserving Guided Diffusion. *International Conference on Learning Representations*, 2024:44819–44850, May 2024.

Ehsan Hesamifard, Hassan Takabi, Mehdi Ghasemi, and Rebecca N. Wright. Privacy-preserving Machine Learning as a Service. *Proceedings on Privacy Enhancing Technologies*, 2018(3):123–142, June 2018. ISSN 2299-0984. doi: 10.1515/popets-2018-0024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

John Hogue. Metro Interstate Traffic Volume. UCI Machine Learning Repository, 2019. DOI: https://doi.org/10.24432/C5X60B.

Seokjin Hwang, Yuting Li, and Kiwan Maeng. Spectrum-Aware Bounds on Invertibility for Privacy-Enhancing Instance Encoding. arXiv preprint arXiv:2608.23382, August 2026. URL https://arxiv.org/abs/2608.23382.

Jon Irureta, Jon Imaz, Aizea Lojo, Javier Fernandez-Marques, Iñigo Perona, and Marco González. Towards active participant centric vertical federated learning: Some representations may be all you need. *Knowledge-Based Systems*, pp. 115334, 2026.

Marcel Kollovieh, Abdul Fatir Ansari, Michael Bohlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang (Bernie) Wang. Predict, Refine, Synthesize: Self-Guiding Diffusion Models for Probabilistic Time Series Forecasting. *Advances in Neural Information Processing Systems*, 36: 28341–28364, December 2023.

S. Kucherenko, S. Tarantola, and P. Annoni. Estimation of global sensitivity indices for models with dependent variables. *Computer Physics Communications*, 183(4):937–946, 2012. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2011.12.020. URL https://www.sciencedirect.com/science/article/pii/S0010465511004085.

Kiwan Maeng, Chuan Guo, Sanjay Kariyappa, and G. Edward Suh. Bounding the Invertibility of Privacy-preserving Instance Encoding using Fisher Information. In *Advances in Neural Information Processing Systems*, volume 36, pp. 51904–51925. Curran Associates, Inc., 2023. doi: 10.52202/075280-2261.

Jorge Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto, and Xavier Parra. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C54S4K.

Cory A. Rieth, Ben D. Amsel, Randy Tran, and Maia B. Cook. Additional Tennessee Eastman Process Simulation Data for Anomaly Detection Evaluation, 2017. URL https://doi.org/10.7910/DVN/6C3JR1.

Aditya Shankar, Hans Brouwer, Rihan Hai, and Lydia Chen. Silofuse: Cross-silo synthetic data generation with latent tabular diffusion models. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 110–123. IEEE, 2024.

Aditya Shankar, Lydia Chen, Arie van Deursen, and Rihan Hai. Wavestitch: Flexible and fast conditional time series generation with diffusion models. *Proc. ACM Manag. Data*, 3(6), December 2025. doi: 10.1145/3769842. URL https://doi.org/10.1145/3769842.

I. M. Sobol'. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and Computers in Simulation*, 55(1):271–280, 2001. ISSN 0378-4754. doi: https://doi.org/10.1016/S0378-4754(00)00270-6. URL https://www.sciencedirect.com/science/article/pii/S0378475400002706. The Second IMACS Seminar on Monte Carlo Methods.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional Score-based Diffusion Models for Probabilistic Time Series Imputation. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24804–24816. Curran Associates, Inc., 2021.

Yuanzhe Wang and Alexandre M Tartakovsky. Latent diffusion posterior sampling with surrogate likelihood guidance for pde inverse problems. *arXiv preprint arXiv:2606.26592*, 2026.

Xinyu Yuan and Yan Qiao. Diffusion-TS: Interpretable Diffusion for General Time Series Generation. In *The Twelfth International Conference on Learning Representations*, October 2023.

# A NOTATION

Table 3: Summary of the notation used.

| Symbol | Description |
|---|---|
| $\mathbf{x}_0 \in \mathbb{R}^{T\times C}$ | Complete time series |
| $T,\ C$ | Number of timesteps and features |
| $s,\ c$ | Time-series timestep and feature indices |
| $\mathbf{m} \in \{0,1\}^{T\times C}$ | Binary observation mask |
| $\mathbf{x}_{\text{obs}} = \mathbf{m} \odot \mathbf{x}_0$ | Partially observed time series |
| $E,\ D$ | Client encoder and decoder |
| $\mathbf{z}_0 = E(\mathbf{x}_0)$ | Latent representation of a complete sequence |
| $L_z,\ D_z$ | Number of latent tokens and embedding dimension |
| $\ell,\ d$ | Latent-token and embedding-dimension indices |
| $\mathbf{x}'_0$ | Interpolation-based completion of $\mathbf{x}_{\text{obs}}$ |
| $\mathbf{z}'_0$ | Latent variable being optimised |
| $\mathbf{z}^*_0$ | Final optimised latent anchor |
| $\mathcal{L}_{\text{obs}}$ | Observation-consistency objective |
| $\mathcal{L}_{\text{reg}}$ | Anchor-optimisation regularisation loss |
| $\lambda_{\text{smooth}}, \lambda_{\text{prox}}$ | Anchor-optimisation regularisation weights |
| $\mathbf{q}_{s,c}$ | Indicator that entries $s$ and $s+1$ are both missing |
| $R$ | Number of stochastic completions |
| $r$ | Stochastic-completion index |
| $Q$ | Reference distribution for stochastic completions |
| $\boldsymbol{\xi}^{(r)}$ | Perturbation used in stochastic completion $r$ |
| $\mathbf{z}_0^{(r)}$ | Latent representation of stochastic completion $r$ |
| $\boldsymbol{\sigma}_{\ell,d}$ | Empirical variance of latent component $(\ell, d)$ |
| $\mathbf{w} \in \mathbb{R}^{L_z\times D_z}$ | Anchor confidence map |
| $\delta$ | Confidence floor |
| $p$ | Fraction of highest-confidence components retained |
| $\kappa$ | $(1-p)$-quantile confidence threshold |
| $\gamma$ | Attenuation factor for lower-confidence components |
| $\varepsilon$ | Numerical stability constant |
| $t$ | Reverse-diffusion timestep |
| $\mathbf{z}_t$ | Noisy latent at reverse-diffusion step $t$ |
| $\hat{\mathbf{z}}_{0\|t}$ | Clean-latent estimate at reverse-diffusion step $t$ |
| $\hat{\boldsymbol{\epsilon}}_t$ | Predicted noise at reverse-diffusion step $t$ |
| $\epsilon_\theta$ | Noise-prediction network |
| $\bar{\alpha}_t$ | Cumulative diffusion noise-schedule coefficient |
| $\mathcal{L}_{\text{guide}}$ | Confidence-weighted latent guidance objective |
| $N_{\text{diff}}$ | Number of diffusion steps |
| $\eta(t),\ K(t)$ | Guidance strength and refinement count at step $t$ |
| $\eta_0,\ K_0$ | Base guidance strength and refinement count |

# B LIMITATIONS AND FUTURE WORK

**Privacy assumptions and latent representation.** VIGIL keeps the raw observations, observation mask, and final decoded outputs on the client, and communicates only latent representations to the server. Its privacy properties therefore inherit those of the underlying encoder. Prior work provides theoretical privacy guarantees under black-box access to the client-side encoder and decoder (Shankar et al., 2024), while weaker assumptions can still provide representation-level privacy depending on the properties of the encoder Maeng et al. (2023); Hwang et al. (2026). Our work focuses on enabling conditional generation under this latent-space setting rather than comparing different privacy-preserving encoders. An important direction for future work is to evaluate different autoencoder families jointly in terms of privacy, reconstruction utility, and suitability for latent conditioning.

**Autoencoder and hyperparameter configuration.** The effectiveness of latent conditioning depends on the geometry induced by the autoencoder. In this work, we use a Transformer autoencoder, whose attention-based representation can mix information across time and variables, making the correspondence between data-space observations and latent components non-trivial and providing a challenging setting for VIGIL. Encoders with stronger locality or more linear structure may admit a more direct mapping between data- and latent-space constraints, whereas more strongly entangled representations may require different confidence and guidance configurations. For a controlled comparison, we use a common architecture and a single set of VIGIL hyperparameters across datasets and missingness scenarios rather than tuning each configuration independently. Dataset- and scenario-specific tuning of the autoencoder, anchor regularisation, confidence thresholding, guidance strength, and refinement schedule could therefore provide additional gains. Studying how these choices interact with the degree of latent entanglement is a useful direction for future work.

**Anchor construction.** Our experiments show that our anchor optimisation strategy can provide meaningful improvements when the interpolation-based initialisation is a poor approximation of the desired latent target, while producing comparable performance when interpolation is already sufficient. This indicates that the usefulness of anchor optimisation is dataset- and scenario-dependent. Future work could investigate alternative initialisation strategies, regularisation objectives, optimisation procedures, learned anchor estimators, and adaptive stopping criteria to further improve the quality and robustness of the optimised latent anchor.

**Interpretation of variance-based confidence.** The variance used by VIGIL should be interpreted as a relative sensitivity measure under the chosen stochastic completion distribution, rather than as calibrated uncertainty under the unknown conditional distribution of the missing values. We use an i.i.d. Gaussian reference distribution because it provides a simple and standardised way to probe how strongly each latent component reacts to unspecified inputs without assuming a particular missing-data model. Other reference distributions, including structured or learned completion distributions, could be studied in future work to determine how the choice of perturbation distribution affects the resulting confidence map.

**Selective versus full guidance.** Selective guidance is particularly useful when the reliability of the optimised anchor varies across the latent representation, as occurs under structured missingness and extrapolative settings such as forecasting. However, it is not expected to dominate full guidance in every case. When observations are distributed throughout the sequence and the optimised anchor is reliable across most latent components, as can occur under random missingness, attenuating lower-ranked components may unnecessarily discard useful conditioning information. This behaviour is reflected in our ablations, where full guidance can outperform selective guidance in such settings. An adaptive mechanism that adjusts the amount of selective guidance according to the estimated reliability structure of the anchor, for example through a data-dependent retention fraction or a threshold derived from the empirical variance distribution, could therefore further improve robustness.

**Evaluation scope.** We evaluate VIGIL on five multivariate time-series datasets, four missingness patterns, and a $50\%$ missingness ratio. We choose $50\%$ as a challenging intermediate setting that avoids both near-complete observation and extremely sparse conditioning while allowing all methods to be compared under the same level of missingness. Nevertheless, a broader evaluation across different missingness ratios would provide a more complete characterisation of the method.

**Extension beyond time series.** Although this work focuses on time-series imputation, the proposed latent-guidance framework is naturally applicable to other partially observed generation tasks in which only part of the target is known, such as image inpainting. Extending VIGIL to these settings, and adapting its anchor construction, confidence estimation, and guidance mechanism to their domain-specific structure, is a promising direction for future work.

# C Additional Background

## C.1 Training-Time Conditioning

These approaches train the denoising network to approximate the conditional noise (or score) directly, by supplying the conditioning information $\mathbf{c}$ as an additional input throughout training, giving the following objective using mean-squared-error (MSE):

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t, \mathbf{c})\right\|^2\right], \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{7}$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, as in (1). For imputation, $\mathbf{c}$ is typically constructed from a binary mask $\mathbf{m} \in \{0,1\}^{T\times C}$ indicating which entries of $\mathbf{x}_0$ are observed, together with the observed values themselves, i.e. $\mathbf{c} = (\mathbf{m}, \mathbf{x}_{\text{obs}})$, where $\mathbf{x}_{\text{obs}} = \mathbf{m} \odot \mathbf{x}_0$. During training, $\mathbf{m}$ is sampled from some distribution over missingness patterns (e.g. randomly generated missing ratios or block-missing patterns), and the network learns to denoise the unobserved entries conditioned on the observed ones (Tashiro et al., 2021; Alcaraz & Strodthoff, 2022). A key limitation is that the network usually does not generalise to observation patterns or missing ratios that differ from the training distributions (Shankar et al., 2025).

## C.2 Inference-Time Conditioning via Guidance

An alternative is to train the diffusion model unconditionally using the same noise-prediction objective as in (7), but with the conditioning input $\mathbf{c}$ omitted. Instead, the conditions $\mathbf{c}$ are injected post-hoc when sampling, steering the reverse process using an auxiliary guidance term (Kollovieh et al., 2023). Formally, the unconditional clean-sample estimate is corrected using gradients of an inference-time loss $\mathcal{L}_{\text{inf}}(\cdot)$ that measures its fidelity to the given conditions:

$$\hat{\mathbf{x}}_{0|t}^{\text{guided}} = \hat{\mathbf{x}}_{0|t} - \lambda \nabla_{\hat{\mathbf{x}}_{0|t}} \mathcal{L}_{\text{inf}}\left(\hat{\mathbf{x}}_{0|t}, \mathbf{c}\right), \tag{8}$$

where $\lambda$ controls the guidance strength and $\hat{\mathbf{x}}_{0|t}$ denotes an estimate of the clean sample obtained from the current noisy state $\mathbf{x}_t$. Since the underlying diffusion model is trained independently of any particular conditioning pattern, guidance-based approaches can in principle be applied to arbitrary masks or missing ratios at test time without retraining.

## C.3 Reverse Diffusion Parameterisation

In Sec. 2, we describe the reverse diffusion process probabilistically through $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t, \mathbf{c})$. Here, we make explicit the noise-prediction parameterisation used for sampling. Given a noisy state $\mathbf{x}_t$, the denoising network predicts the noise $\epsilon_\theta(\mathbf{x}_t, t)$, from which the corresponding clean-sample estimate is obtained as

$$\hat{\mathbf{x}}_{0|t} = \frac{\mathbf{x}_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}}.$$

Using this clean-sample estimate, the reverse transition is

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{\mathbf{x}}_{0|t} + \sqrt{1-\bar{\alpha}_{t-1}-\sigma_t^2}\,\epsilon_\theta(\mathbf{x}_t, t) + \sigma_t \boldsymbol{\zeta}_t, \qquad \boldsymbol{\zeta}_t \sim \mathcal{N}(\mathbf{0}, I), \tag{9}$$

where

$$\sigma_t = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}}\sqrt{1-\frac{\bar{\alpha}_t}{\bar{\alpha}_{t-1}}}.$$

The same parameterisation applies in latent diffusion by replacing $\mathbf{x}_t$ and $\hat{\mathbf{x}}_{0|t}$ with their latent-space counterparts $\mathbf{z}_t$ and $\hat{\mathbf{z}}_{0|t}$.

# D Additional Implementation details

## D.1 Latent Anchor Optimisation Regularisation

The observation loss alone constrains the decoded anchor only at observed entries, leaving the missing regions unconstrained. To prevent the optimisation from producing implausible values in these regions, we regularise the optimised anchor using

$$\mathcal{L}_{\text{reg}}(\mathbf{z}'_0, \mathbf{m}) = \lambda_{\text{smooth}} \mathcal{L}_{\text{smooth}}(\mathbf{z}'_0, \mathbf{m}) + \frac{\lambda_{\text{prox}}}{L_z D_z} \left\| \mathbf{z}'_0 - \mathbf{z}'^{(0)}_0 \right\|_F^2, \tag{10}$$

where $\mathbf{z}'^{(0)}_0$ is the frozen initial encoded interpolation dummy. The two terms respectively encourage temporal smoothness within missing regions and proximity to the interpolation-based latent initialisation.

**Smoothness regularisation.** We discourage abrupt variation inside missing regions by penalising differences between consecutive reconstructed values that are both unobserved. Let $\mathbf{q}_{s,c} = (1 - \mathbf{m}_{s,c})(1 - \mathbf{m}_{s+1,c})$, which equals one only when consecutive entries $s$ and $s+1$ of feature $c$ are both missing. We define

$$\mathcal{L}_{\text{smooth}}(\mathbf{z}'_0, \mathbf{m}) = \frac{\sum_{s,c} \mathbf{q}_{s,c} \left( [D(\mathbf{z}'_0)]_{s+1,c} - [D(\mathbf{z}'_0)]_{s,c} \right)^2}{\sum_{s,c} \mathbf{q}_{s,c}}.$$

When no pair of consecutive missing entries exists, the loss is set to zero. This term encourages locally smooth reconstructions within gaps while leaving observed entries unaffected.

**Latent proximity regularisation.** The diffusion model is trained on latent representations produced by the encoder, whereas optimising the anchor using only data-space losses can move it away from the region represented by the interpolation-based initialisation. We therefore penalise deviation from the initial latent anchor $\mathbf{z}'^{(0)}_0$ through

$$\frac{\lambda_{\text{prox}}}{L_z D_z} \left\| \mathbf{z}'_0 - \mathbf{z}'^{(0)}_0 \right\|_F^2.$$

This term acts as a weak latent-space prior toward the encoded interpolation, discouraging excessive drift while still allowing the anchor to move when required to better satisfy the observed data.

## D.2 Latent Standardisation

For clarity, the main text omits the fixed standardisation applied to the autoencoder latents. Let $\widetilde{\mathbf{z}}$ denote a latent in the original autoencoder scale. Before latent optimisation or diffusion, it is standardised as

$$\mathbf{z} = \frac{\widetilde{\mathbf{z}} - \boldsymbol{\mu}_z}{\boldsymbol{\sigma}_z},$$

where $\boldsymbol{\mu}_z$ and $\boldsymbol{\sigma}_z$ are latent statistics computed from the training data.

In particular, for the interpolation-based initialisation, the autoencoder first produces

$$\widetilde{\mathbf{z}}'_0 = E(\mathbf{x}'_0),$$

which is then standardised to obtain the initialisation used in the main text,

$$\mathbf{z}'_0 = \frac{\widetilde{\mathbf{z}}'_0 - \boldsymbol{\mu}_z}{\boldsymbol{\sigma}_z}.$$

The latent anchor $\mathbf{z}'_0$ is optimised in this standardised space. Whenever it is decoded, it is first mapped back to the autoencoder scale,

$$\widetilde{\mathbf{z}}'_0 = \mathbf{z}'_0 \odot \boldsymbol{\sigma}_z + \boldsymbol{\mu}_z,$$

and the reconstruction is obtained as $D(\widetilde{\mathbf{z}}'_0)$. Accordingly, the notation $D(\mathbf{z}'_0)$ used in the main text is shorthand for this de-standardisation followed by decoding.

### D.3 Anchor Confidence Map Estimation Algorithm

**Algorithm 2** Variance-Based Anchor Confidence Map Estimation

**Require:** Partially observed sequence $\mathbf{x}_{\text{obs}}$, observation mask $\mathbf{m}$, encoder $E$, number of stochastic completions $R$, confidence floor $\delta$, retained fraction $p$, attenuation factor $\gamma$, numerical constant $\varepsilon$

**Ensure:** Anchor confidence map $\mathbf{w} \in \mathbb{R}^{L_z \times D_z}$

**for** $r = 1, \dots, R$ **do**
  Sample $\boldsymbol{\xi}^{(r)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_0^{(r)} \leftarrow \mathbf{x}_{\text{obs}} + (1 - \mathbf{m}) \odot \boldsymbol{\xi}^{(r)}$
  $\mathbf{z}_0^{(r)} \leftarrow E(\mathbf{x}_0^{(r)})$
**end for**
**for all** $(\ell, d) \in [L_z] \times [D_z]$ **do**
  $\bar{\boldsymbol{\mu}}_{\ell,d} \leftarrow \frac{1}{R} \sum_{r=1}^{R} \mathbf{z}_{0,\ell,d}^{(r)}$
  $\boldsymbol{\sigma}_{\ell,d} \leftarrow \frac{1}{R-1} \sum_{r=1}^{R} \left( \mathbf{z}_{0,\ell,d}^{(r)} - \bar{\boldsymbol{\mu}}_{\ell,d} \right)^2$
**end for**
$\mathbf{a}_{\ell,d} \leftarrow (\boldsymbol{\sigma}_{\ell,d} + \varepsilon)^{-1}$ for all $(\ell, d)$
$\mathbf{w}_{\ell,d} \leftarrow \delta + (1 - \delta) \frac{\mathbf{a}_{\ell,d}}{\max_{\ell',d'} \mathbf{a}_{\ell',d'}}$
$\kappa \leftarrow \text{Quantile}_{1-p}(\mathbf{w})$
**for all** $(\ell, d) \in [L_z] \times [D_z]$ **do**
  **if** $\mathbf{w}_{\ell,d} < \kappa$ **then**
    $\mathbf{w}_{\ell,d} \leftarrow \gamma \mathbf{w}_{\ell,d}$
  **end if**
**end for**
**return** $\mathbf{w}$

Algorithm 2 summarises the variance-based construction of the anchor confidence map. The client repeatedly fills only the unspecified entries with samples from the reference distribution while keeping the observations fixed, and encodes each completed sequence into the latent space. The element-wise variance across these latent representations measures how sensitive each latent component is to the unknown values. These variance estimates are converted into confidence weights, after which the top-$p$ components are retained and the remaining weights are attenuated.

### D.4 Hyperparameters, Architectures and Hardware

**Autoencoder.** We use a Transformer autoencoder to compress each multivariate time series into a latent representation of shape $16 \times 128$. The encoder projects the input channels to a 128-dimensional embedding space, adds sinusoidal positional encodings, and applies four Transformer encoder layers with eight attention heads, a feed-forward dimension of 256, GELU activations, and dropout $0.1$. Sixteen learned latent queries then cross-attend to the encoded sequence to produce the latent representation. The decoder uses positional sequence queries that cross-attend to the latent tokens, followed by four Transformer layers and a linear output projection.

The autoencoder is trained for 50 epochs with batch size 32 using AdamW with learning rate $10^{-4}$ and weight decay $10^{-4}$. We optimise a Smooth L1 reconstruction loss with $\beta = 0.5$ and clip the gradient norm to $1.0$.

**Latent diffusion model.** We train a separate unconditional latent diffusion model for each dataset using the TSDiff framework (Kollovieh et al., 2023). The model operates directly on the autoencoder latent representations, with forecasting-specific lag features and temporal covariates removed. To capture dependencies across latent features, the S4-based residual backbone is augmented with Transformer-based feature mixing.

The denoising backbone uses hidden dimension 768, diffusion-step embedding dimension 256, 16 residual blocks, and dropout $0.03$. The forward diffusion process uses 200 diffusion steps with a linear noise schedule. Models are trained for 300 epochs with batch size 64 and learning rate $3 \times 10^{-5}$.

**Anchor optimisation and latent guidance.** Anchor optimisation is initialised by linearly interpolating the observed values and encoding the completed sequence with the autoencoder. Starting from this initialisation, the latent anchor is optimised for 200 steps using Adam with learning rate $10^{-2}$, while the parameters of the pre-trained autoencoder remain fixed. We use $\lambda_{\text{prox}} = 10^{-3}, \lambda_{\text{smooth}} = 10^{-2}$, Optimisation is performed in the standardised latent space described in Appendix D.2.

For variance-based confidence estimation, we generate $R = 100$ stochastic completions and use a confidence floor of $\delta = 0.005$. The top $p = 20\%$ of latent components are retained at their original confidence values, while the remaining components are attenuated by $\gamma = 10^{-3}$. Latent guidance uses base guidance strength $\eta_0 = 1$ and base refinement count $K_0 = 100$, with the time-dependent schedules defined in Sec. 3.3. These hyperparameters are fixed across datasets and missingness scenarios.

**Evaluation protocol.** All experiments use 500 evaluation sequences with a missing ratio of $50\%$. For each sequence, we generate 10 independent imputations and aggregate results over five inference seeds. The evaluation subset and missingness masks are generated using a fixed seed of 42, while stochastic guidance construction and diffusion sampling vary across inference seeds $1, \ldots, 5$.

Following Alcaraz & Strodthoff (2022), we evaluate four missingness scenarios. In the *random* setting, randomly selected time steps are masked across all channels. In the *block* setting, each channel contains an independently positioned contiguous missing interval. In the *blackout* setting, the same contiguous interval is missing across all channels. In the *forecasting* setting, the final contiguous portion of each sequence is masked.

**Evaluation metrics.** For each input, the model produces 10 imputations. We use their median as the deterministic prediction when computing MSE on the masked entries. Full MSE results are reported in Appendix H.1.

For probabilistic evaluation, we follow the CRPS implementation of Tashiro et al. (2021). Empirical quantiles are computed from the 10 imputations at levels $0.05, 0.10, \ldots, 0.95$. For each quantile, we compute the corresponding quantile loss over the masked entries. These losses are averaged and normalised by the sum of the absolute ground-truth values over the missing region. Lower CRPS indicates a more accurate and better calibrated predictive distribution.

**Hardware and software.** All experiments are run on a workstation equipped with an NVIDIA GeForce RTX 4090 GPU with 24 GB of memory and a 13th Gen Intel Core i7-13700 CPU. The system runs Ubuntu 22.04.4 LTS. Experiments use PyTorch 2.0.1 with CUDA 11.7 and cuDNN 8.5.

### D.5 Dataset Preprocessing

For all datasets, preprocessing is performed before constructing the diffusion inputs. Whenever an official train–test partition is unavailable, we use a chronological 80/20 split rather than a random split. This ensures that all training observations precede the test observations, prevents future measurements from leaking into training, and reflects the intended forecasting and temporal-imputation setting. Continuous features are standardised independently using the mean and standard deviation of the training partition, and the same statistics are subsequently applied to the test partition. The resulting sequences are represented using single-precision floating-point values. We adjust the window stride across datasets to obtain a comparable number of training samples, reducing excessive redundancy and computational cost for larger datasets.

**Appliances Energy Prediction.** Observations are ordered by timestamp and divided chronologically into training and test partitions. Missing values are filled within each partition using forward filling followed by backward filling, preserving temporal continuity while ensuring that every input window is complete. The synthetic variables `rv1` and `rv2` are discarded because they are random variables without meaningful physical or temporal interpretation. We do not include explicit calendar features such as hour of day or day of week, so that performance reflects the model's ability to learn temporal structure directly from the measured variables rather than relying on externally provided time indicators. The normalised series is segmented into overlapping windows of length 168 with stride 1, which maximises the number of available training examples while remaining computationally manageable for this dataset.

**Beijing Air Quality.** We use measurements from the Aotizhongxin monitoring station only, yielding a single continuous multivariate time series with a consistent set of sensor variables. We retain eleven pollutant and meteorological variables and fill missing measurements within each partition using forward and backward filling, which exploits the temporal continuity of environmental observations. We exclude explicit hour-of-day and month-of-year features so that the model must infer daily and seasonal patterns from the environmental measurements themselves and so that evaluation remains focused on the observed sensor channels. Windows of length 168 are extracted with stride 2, reducing the associated training cost while retaining substantial temporal overlap.

**Human Activity Recognition** We use the official training and test partitions and retain the nine provided inertial signals corresponding to body acceleration, body gyroscope measurements, and total acceleration along the three spatial axes. The official split is preserved because it follows the dataset's established evaluation protocol and avoids constructing a potentially incompatible partition. Each feature is standardised using statistics calculated over all time steps and sequences in the training partition. The dataset already consists of fixed-length windows of 128 time steps, so no additional window extraction or overlap is introduced.

**Tennessee Eastman Process.** We use the official fault-free training and testing files. The fault identifier, simulation-run identifier, and sample index are removed because they are metadata rather than process measurements. Rows containing incomplete measurements are discarded instead of being interpolated, avoiding the introduction of synthetic process values into a controlled industrial simulation. The simulation-run identifiers are retained separately during preprocessing so that windows can be constructed independently within each run; this prevents a window from combining the end of one simulation with the beginning of another. After standardisation using the training statistics, values are clipped to the interval $[-5, 5]$ to limit the influence of extreme standardised observations and improve the numerical stability of autoencoder and diffusion training. We construct windows of length 64 using an adaptive training stride chosen to produce approximately 30,000 windows. This limits computational cost while maintaining coverage across all simulation runs.

**Metro Interstate Traffic Volume.** We retain traffic volume, temperature, hourly rainfall, hourly snowfall, and cloud coverage. Rows containing missing or non-numeric measurements are removed so that the model is trained only on complete observed traffic and weather records, without introducing assumptions through interpolation. Because no official temporal partition is provided, the observations are divided chronologically using an 80/20 split. Standardised values are clipped to $[-5, 5]$ to reduce the effect of extreme weather measurements and stabilise model optimisation. The series is segmented into windows of length 168, and the stride is selected adaptively to produce approximately 20,000 training windows. The same stride is used for the test partition, providing comparable overlap and temporal coverage in both partitions.

### D.6 Baselines Implementation

For a consistent comparison, all methods are evaluated on the same preprocessed test sequences and exactly the same missingness masks, with 10 imputations generated per sequence and results reported over five inference seeds. The unconditional methods are trained once per dataset and reused across all missingness scenarios, whereas the conditional baselines follow their respective task-specific training procedures as detailed below.

**TSDiff.** We train TSDiff unconditionally in the original data space using the same diffusion architecture and optimisation setup as our backbone, with the difference that diffusion is performed in data-space. At inference, we use TSDiff's quantile self-guidance (TSDiff-Q) with DDPM sampling, corresponding to the stronger guidance variant reported by the original method. We set the guidance scale to $0.5$ based on preliminary experiments, as larger values led to numerical instability.

**Diffusion-TS.** We train Diffusion-TS unconditionally in the original data space, using a single model per dataset across all missingness scenarios. We use four encoder and four decoder layers with hidden dimension 128 and eight attention heads, with 500 diffusion steps, a cosine noise schedule, and an $L_1$ training objective. Training uses a batch size of 64, a base learning rate of $10^{-4}$, gradient accumulation over two batches, and an exponential moving average with decay $0.995$. Models are

trained for 9,900 steps. At inference, we use the conditional restoration procedure with 250 sampling steps; the same restoration procedure is applied to all four tasks, with the corresponding evaluation mask defining the observed and missing portions of each sequence.

**CSDI.** We retain the standard CSDI model configuration rather than resising it to match our diffusion backbone, as it already provides a strong task-specific conditional-imputation baseline. The model uses four diffusion layers with 64 hidden channels, eight attention heads, 128-dimensional diffusion and temporal embeddings, a 16-dimensional feature embedding, and 50 diffusion steps with a quadratic noise schedule. We train for 200 epochs with batch size 16 and a learning rate of $10^{-3}$. Following CSDI's self-supervised training procedure, conditioning masks are generated randomly during training, with the masking proportion independently sampled for each sequence. Following CSDI's generic random target-selection strategy for arbitrary imputation masks, a single model is trained per dataset and subsequently evaluated under all four missingness scenarios.

**SSSD-S4.** We use SSSD-S4 with 16 residual layers, 128 residual and skip channels, and bidirectional S4 layers with state dimension 64, selecting this configuration to maintain a broadly comparable model capacity to our diffusion backbone. We train for 50,000 iterations with Adam at a learning rate of $2 \times 10^{-4}$ and batch size 64, using 200 diffusion steps with a linear noise schedule from $10^{-4}$ to $0.02$. The training objective is evaluated only over artificially masked entries. As SSSD-S4 is trained conditionally for imputation, we adapt its training masks to the evaluation task: random masks are used for random missingness, channel-wise contiguous blocks for block missingness and forecasting, and a shared contiguous block across channels for blackout missingness, with approximately $50\%$ of each sequence masked during training.

**Data Guidance.** Data Guidance uses the same unconditional data-space diffusion backbone and training configuration as TSDiff, but replaces its native observation guidance with our iterative refinement procedure applied directly in data space. At each reverse step, the clean estimate is updated using a normalised gradient of the reconstruction error on observed entries. Following limited preliminary hyperparameter tuning, we use a base guidance scale of $1$ and $10$ refinement iterations. Because the constraint is enforced directly on the generated values, observation consistency is reached more quickly than in latent-space guidance, making additional iterations unnecessarily strong and costly. This is particularly important because data-space sampling is substantially more expensive overall, as shown in Appendix H.4.

**Spinning Decoder.** Spinning Decoder uses the same autoencoder and unconditional latent diffusion model as our method, but applies observation guidance through the decoder rather than directly in latent space. At each reverse diffusion step, the current clean-latent estimate is decoded and its reconstruction error over the observed data-space entries is backpropagated through the frozen decoder to update the latent estimate. Following limited preliminary hyperparameter tuning and the same logic applied to the data guidance hyperparemeter choosing, we use a base guidance scale of $1$ and a base number of refinement iterations of $10$. Importantly, decoder forward and backward passes are repeatedly required throughout reverse diffusion, making this baseline incur substantial inference overhead, which limits its practicality for efficient imputation.

Together, Data Guidance and Spinning Decoder provide controlled comparisons for disentangling the contributions of latent-space diffusion and latent-space guidance. Comparing Data Guidance with Spinning Decoder highlights the effect of training the diffusion process in the learned latent space, while comparing Spinning Decoder with our method shows the trade-off between efficiency and performance of guiding itself directly in latent space.

# E Theoretical Motivation for Variance-Based Latent Confidence

We provide a formal interpretation of the variance-based confidence estimator introduced in Sec. 3.2. Consider a fixed partially observed sequence $\mathbf{x}_{\text{obs}}$ with observation mask $\mathbf{m}$. Let $\boldsymbol{\xi} \sim Q$ denote a random perturbation used to fill the missing entries, where $Q$ may be an arbitrary joint distribution. The corresponding completed sequence is

$$\mathbf{X} = \mathbf{x}_{\text{obs}} + (1 - \mathbf{m}) \odot \boldsymbol{\xi}.$$

Define the random latent representation

$$\mathbf{Z} = E(\mathbf{X}) = E(\mathbf{x}_{\text{obs}} + (1 - \mathbf{m}) \odot \boldsymbol{\xi}) .$$

For latent component $(\ell, d)$, let $\mathbf{Z}_{\ell,d}$ denote the corresponding scalar random variable, i.e., the $(\ell, d)$ entry of $\mathbf{Z}$. We assume $\mathbb{E}[\mathbf{Z}_{\ell,d}^2] < \infty$. Since $\mathbf{x}_{\text{obs}}$ and $\mathbf{m}$ remain fixed, all variability in $\mathbf{Z}_{\ell,d}$ is induced exclusively by changes to the unspecified entries.

Let $\boldsymbol{\xi}^{(1)}, \ldots, \boldsymbol{\xi}^{(R)} \overset{\text{i.i.d.}}{\sim} Q$ denote $R$ independent perturbation draws, and define

$$\mathbf{z}_0^{(r)} = E\Big(\mathbf{x}_{\text{obs}} + (1 - \mathbf{m}) \odot \boldsymbol{\xi}^{(r)}\Big) , \qquad \bar{\boldsymbol{\mu}}_{\ell,d} = \frac{1}{R} \sum_{r=1}^{R} \mathbf{z}_{0,\ell,d}^{(r)}.$$

The empirical variance used by our method is

$$\boldsymbol{\sigma}_{\ell,d} = \frac{1}{R-1} \sum_{r=1}^{R} \Big(\mathbf{z}_{0,\ell,d}^{(r)} - \bar{\boldsymbol{\mu}}_{\ell,d}\Big)^2 .$$

Using the standard pairwise representation of the sample variance,

$$\boldsymbol{\sigma}_{\ell,d} = \frac{1}{R(R-1)} \sum_{r<s} \Big(\mathbf{z}_{0,\ell,d}^{(r)} - \mathbf{z}_{0,\ell,d}^{(s)}\Big)^2 . \tag{11}$$

Thus, the empirical variance aggregates the pairwise disagreement of each latent component across all $\binom{R}{2}$ pairs of stochastic completions.

To show that this estimator is unbiased, let $\boldsymbol{\xi}, \boldsymbol{\xi}' \overset{\text{i.i.d.}}{\sim} Q$ denote two independent perturbation draws and let $\mathbf{Z}_{\ell,d}$ and $\mathbf{Z}'_{\ell,d}$ denote the corresponding latent values. Since they are identically distributed and independent,

$$\begin{aligned}
\mathbb{E}\big[(\mathbf{Z}_{\ell,d} - \mathbf{Z}'_{\ell,d})^2\big] &= \mathbb{E}[\mathbf{Z}_{\ell,d}^2] + \mathbb{E}[{\mathbf{Z}'_{\ell,d}}^2] - 2\mathbb{E}[\mathbf{Z}_{\ell,d}\mathbf{Z}'_{\ell,d}] \\
&= 2\mathbb{E}[\mathbf{Z}_{\ell,d}^2] - 2\mathbb{E}[\mathbf{Z}_{\ell,d}]^2 \\
&= 2\,\mathrm{Var}_Q(\mathbf{Z}_{\ell,d}).
\end{aligned}$$

Taking the expectation of Eq. 11 therefore gives

$$\begin{aligned}
\mathbb{E}[\boldsymbol{\sigma}_{\ell,d}] &= \frac{1}{R(R-1)} \binom{R}{2} \mathbb{E}\Big[\big(\mathbf{Z}_{\ell,d} - \mathbf{Z}'_{\ell,d}\big)^2\Big] \\
&= \mathrm{Var}_Q(\mathbf{Z}_{\ell,d}).
\end{aligned}$$

Hence, for independent Monte Carlo draws from $Q$, $\boldsymbol{\sigma}_{\ell,d}$ is an unbiased estimator of $\mathrm{Var}_Q(\mathbf{Z}_{\ell,d})$.

The pairwise representation also provides the desired sensitivity interpretation. Since $\mathbf{x}_{\text{obs}}$ and $\mathbf{m}$ remain fixed across completions, disagreement between $\mathbf{z}_{0,\ell,d}^{(r)}$ and $\mathbf{z}_{0,\ell,d}^{(s)}$ can only arise from changes to the unspecified entries. A low variance therefore indicates that latent component $(\ell, d)$ is insensitive to those entries under $Q$, whereas a high variance indicates that its representation depends strongly on them. This motivates assigning larger confidence to latent components with lower variance.

Importantly, the interpretation in Eq. 11 does not require the coordinates of $\boldsymbol{\xi}$ to be independent. The reference distribution $Q$ may be an arbitrary joint distribution and may therefore contain temporal and cross-variable dependencies. Independence is required only between separate Monte Carlo draws $\boldsymbol{\xi}$ and $\boldsymbol{\xi}'$ from $Q$, not between the entries within an individual perturbation. This interpretation is conceptually related to variance-based global sensitivity analysis, where variation in model outputs induced by varying input variables is used to quantify their influence (Sobol', 2001). Classical Sobol analysis assumes independent inputs, while variance-based sensitivity analysis has also been generalised to dependent input distributions (Kucherenko et al., 2012). Our estimator is not a Sobol-style global sensitivity index or variance decomposition; instead, it is an instance-conditional sensitivity proxy that adopts the same underlying rationale that output variation induced by perturbing inputs reflects sensitivity to those inputs. In our setting, $\mathbf{x}_{\text{obs}}$ and $\mathbf{m}$ are fixed, and the

unspecified entries are jointly perturbed under $Q$, with the resulting sensitivity captured directly by the pairwise variance representation above. The corresponding population interpretation holds for any joint perturbation distribution $Q$, provided $\mathbf{Z}_{\ell,d}$ has finite second moment under $Q$.

In our implementation, we instantiate the reference distribution as $Q = \mathcal{N}(\mathbf{0}, \mathbf{I})$, which is equivalent to independently sampling each perturbation entry from a standard Gaussian, as described in Sec. 3.2. This choice does not assume that the true missing variables are independent or Gaussian. Instead, it defines a standardised reference distribution for probing the sensitivity of the encoder to the unspecified input coordinates without imposing an assumed temporal or cross-variable dependency structure.

The Gaussian reference distribution induces full support over the missing-value space. Under this Gaussian instantiation, since the encoder $E(\cdot)$ is continuous, if

$$\mathrm{Var}_{\boldsymbol{\xi}\sim\mathcal{N}(\mathbf{0},\mathbf{I})}\left[\mathbf{Z}_{\ell,d}\right] = 0,$$

then latent component $(\ell, d)$ is invariant to all possible values assigned to the missing entries. To see this, variance zero implies that the component is constant almost surely under the Gaussian reference distribution. If there existed a completion for which its value differed, continuity of the encoder would imply a neighbourhood of differing values; because the Gaussian distribution assigns positive probability to every non-empty open neighbourhood, this would contradict zero variance.

For finite nonzero variance, we make the weaker claim that $\boldsymbol{\sigma}_{\ell,d}$ provides a relative measure of sensitivity under the chosen reference distribution. It should therefore be interpreted as a confidence proxy for determining which latent components are least affected by the unspecified entries, rather than as calibrated uncertainty under the unknown missing-data distribution.

## F Feature Dimensionality and Latent Compression

Among the evaluated datasets, TEP presents the most challenging setting for VIGIL. While TEP uses a shorter temporal window than several other datasets, it contains substantially more variables per timestep ($C = 52$). In our autoencoder, the feature vector at each timestep is first projected into a shared $D_z = 128$ representation, after which the temporal sequence is compressed into $L_z = 16$ latent tokens. Hence, the difficulty of TEP cannot be characterised solely by the total number of scalar observations in each input window. Rather, its 52 strongly coupled process variables must be represented jointly within the same per-timestep embedding before temporal compression.

This distinction may help explain why TEP is more difficult for our fixed-capacity latent representation. Consecutive observations in a time series are often temporally redundant and can therefore be compressed relatively efficiently, whereas a large number of interacting process variables may contain more heterogeneous information at each timestep. As a result, latent components may depend on a broader combination of variables, making the representation more entangled and potentially making anchor optimisation from partial observations more difficult.

We note that the current experiments do not isolate feature dimensionality or variable coupling from other dataset-specific properties, so this should be interpreted as a plausible explanation rather than a demonstrated cause of the weaker TEP results. Our experiments intentionally use the same autoencoder architecture and latent dimensionality across all datasets to avoid dataset-specific tuning. Adapting the latent capacity or representation architecture to datasets with many strongly coupled variables, and studying how feature dimensionality and dependency structure affect latent compression, is therefore a natural direction for future work.

## G Intuition on Interpolation Weighting Strategy

The interpolation-based weighting strategy performs particularly well on the Appliances dataset in the guidance-mask ablation. In this section, we first describe how these weights are constructed and then illustrate why they can provide a useful approximation of latent reliability when the encoder preserves local temporal structure. We finally discuss how this interpretation relates to both variance-based and full guidance in the ablation results.

### G.1 Interpolation Weighting

Let $\mathbf{m} \in \{0,1\}^{T\times C}$ denote the data-space observation mask for one sample, where $\mathbf{m}_{s,c} = 1$ indicates that channel $c$ is observed at timestep $s$, and $\mathbf{m}_{s,c} = 0$ indicates that it is missing. Because different channels may have different missing regions, we first define the observation density at timestep $s$ as $\mathbf{u}_s = C^{-1}\sum_{c=1}^{C}\mathbf{m}_{s,c}$. The resulting signal $\mathbf{u} \in [0,1]^T$ represents the fraction of observed channels at each timestep: $\mathbf{u}_s = 1$ indicates that all channels are observed, $\mathbf{u}_s = 0$ that all are missing, and intermediate values that only a subset is observed.

We then linearly resample this observation-density signal from the input sequence length $T$ to the latent sequence length $L_z$. Let $\mathbf{H} \in \mathbb{R}^{L_z\times T}$ denote the corresponding fixed linear-interpolation operator, giving $\widetilde{\mathbf{u}} = \mathbf{H}\mathbf{u}$. Equivalently, if latent position $\ell$ maps between neighbouring input timesteps $i_\ell$ and $i_\ell+1$, then $\widetilde{\mathbf{u}}_\ell = (1-\alpha_\ell)\mathbf{u}_{i_\ell} + \alpha_\ell\mathbf{u}_{i_\ell+1}$, where $\alpha_\ell \in [0,1]$ is fixed by $T$, $L_z$, and $\ell$. The same temporal confidence is assigned to every latent embedding dimension, giving $\mathbf{w}^{\text{interp}}_{\ell,d} = \delta + (1-\delta)\widetilde{\mathbf{u}}_\ell$ for $d \in \{1,\dots,D_z\}$. We then apply the same top-$p$ selection and attenuation by $\gamma$ used for variance-based confidence in Sec. 3.2. Interpolation therefore directly projects the temporal geometry of the data-space observation mask onto the latent-token axis, implicitly assuming that latent-token position approximately preserves input temporal position.

### G.2 Local-Encoder Intuition

To illustrate when interpolation can approximate latent reliability, Fig. 4 considers an idealised encoder applied to $\mathbf{x}_0 \in \mathbb{R}^{4\times 4}$. For simplicity, the encoder produces $L_z = 2$ latent positions with $D_z = 1$. It applies the local averaging kernel

$$\mathbf{A} = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{bmatrix}$$

with temporal stride two. The first application aggregates all channels over the first two timesteps, while the second aggregates the final two timesteps. For the values shown in Fig. 4, this gives

$$\mathbf{z}_0 = \begin{bmatrix} \frac{1}{8}\sum_{s=1}^{2}\sum_{c=1}^{4}\mathbf{x}_{0,s,c} \\ \frac{1}{8}\sum_{s=3}^{4}\sum_{c=1}^{4}\mathbf{x}_{0,s,c} \end{bmatrix} = \begin{bmatrix} 1.0 \\ 1.8 \end{bmatrix}.$$

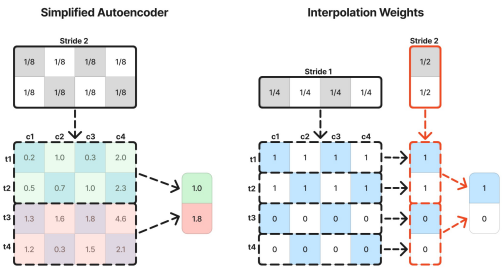


Figure 4: Interpolation weighting for an idealised local encoder. The encoder averages local data-space regions, while interpolation projects the observation density onto the latent-token axis. In this example, both operations induce the same local weighting pattern.

Interpolation processes the corresponding observation mask $\mathbf{m}$ in two analogous stages. First, averaging across channels with $\mathbf{q} = \frac{1}{4}[1,1,1,1]^\top$ gives $\mathbf{u} = \mathbf{m}\mathbf{q} = [1,1,0,0]^\top$ for the mask shown in the figure. Resampling from $T = 4$ to $L_z = 2$ with linear interpolation and `align_corners=False` corresponds in this example to

$$\mathbf{H} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

Hence $\widetilde{\mathbf{u}} = \mathbf{H}\mathbf{u} = [1,0]^\top$ before applying the confidence floor. With $\delta = 0.005$, these weights become $[1, 0.005]^\top$ before top-$p$ selection and attenuation.

The connection with the simplified encoder becomes explicit by writing the temporal averaging kernel as $\mathbf{h} = [\frac{1}{2},\frac{1}{2}]^\top$. Its composition with the channel average is

$$\mathbf{h}\mathbf{q}^\top = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}\begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix} = \begin{bmatrix} \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \\ \frac{1}{8} & \frac{1}{8} & \frac{1}{8} & \frac{1}{8} \end{bmatrix} = \mathbf{A}.$$

Thus, in this idealised case, the operation applied to the observation mask has exactly the same local support as the encoder applies to the data. Consequently, $\widetilde{\mathbf{u}}_1 = 1$ because every input contributing to

the first latent position is observed, whereas $\widetilde{\mathbf{u}}_2 = 0$ because every input contributing to the second position is missing. Therefore, even though our actual encoder is substantially more expressive, interpolation can remain a useful approximation on datasets for which learned latent queries learn approximately temporally ordered attention patterns.

### G.3 Limitations and Comparison with Variance Weighting

The example above explains why interpolation can provide a meaningful latent mask, but it does not imply that interpolation should outperform variance weighting. In fact, both strategies identify the two latent positions correctly in this idealised case. If the missing values vary across stochastic completions, then $\mathrm{Var}(\mathbf{z}_{0,1}) = 0$ because the first latent position depends only on observed inputs, whereas $\mathrm{Var}(\mathbf{z}_{0,2}) > 0$ because the second depends on missing inputs.

The two strategies differ when the encoder mixes observed and missing inputs more intricately. Interpolation measures a structural quantity: the amount of observed data associated with a latent position, whereas variance weighting measures how strongly each latent component changes when unspecified entries are varied. These quantities need not induce the same ranking. Moreover, when the compression ratio $T/L_z$ is large, linear interpolation relies only on nearby observation-density values and may not reflect all observations within the temporal region represented by a latent token. A component can therefore contain useful observed information while remaining sensitive to missing inputs, or have little direct observed support while remaining relatively insensitive. When the encoder preserves strong temporal locality, interpolation can still provide a better selection of latent positions, which may explain its stronger performance on Appliances.

The Appliances results also suggest a complementary explanation. Full guidance performs particularly well on this dataset under random missingness and, more generally, achieves the lowest CRPS on four of the five datasets in this scenario. When the optimised anchor is informative across most of the latent representation, restricting guidance to the top-$p$ components can unnecessarily attenuate useful anchor information. In such settings, lower performance from variance-based selection need not indicate an inaccurate sensitivity estimate; selective guidance itself may simply be unnecessary. On the other hand, forecasting illustrates the opposite case. Here the missing region lies beyond the observed prefix, so constructing the corresponding part of the anchor requires extrapolation. Full guidance deteriorates substantially, consistent with heterogeneous anchor reliability and selective guidance becoming more valuable.

The Appliances behaviour may therefore arise from two complementary effects: the projected observation mask may correspond unusually well to the encoder's latent structure, and the optimised anchor may already be reliable across a large fraction of the latent representation. Outside such cases, interpolation relies on a fixed correspondence between input and latent positions that need not hold for a nonlinear encoder with global attention. Variance weighting instead operates after the encoder and measures sensitivity separately for every latent component, allowing it to account for learned temporal mixing, channel-specific effects, nonlinearities, and differences across embedding dimensions. This is consistent with the overall ablation, where variance-based selection achieves the strongest performance across four of the five datasets.

## H Additional Results

### H.1 MSE Results

While CRPS is our primary metric because it evaluates the full predictive distribution by jointly accounting for accuracy and uncertainty, we additionally report mean squared error (MSE) to assess pointwise reconstruction accuracy on the missing entries. For each missing entry, we first take the median across the stochastic imputations and then average the squared error over all missing entries, using the median as a representative point estimate that is less sensitive to extreme samples. MSE therefore provides a complementary view by measuring how closely the imputed values match the ground truth irrespective of the spread or calibration of the predictive distribution, allowing us to distinguish accurate reconstruction from improvements that arise primarily through better uncertainty estimation.

Table 4: Mean MSE over five inference seeds under $50\%$ block, forecasting, random, and blackout missingness; standard deviations across seeds are reported in parentheses in units of $10^{-3}$. Bold and underlined values denote the best and second-best mean results, respectively, among the conditional inference methods; rankings are based on the unrounded means.

| | | Conditional Inference | | | | Conditional Training | |
|---|---|---|---|---|---|---|---|
| | | Latent-Space Diffusion | Data-Space Diffusion | | | Data-Space Diffusion | |
| Dataset | Scenario | VIGIL (Ours) | Data Guidance | TSDiff | Diff.-TS | CSDI | SSSD-S4 |
| | | Mean MSE (SD $\times 10^{-3}$) | | | | | |
| Appliances | Block | 0.47 (1.35) | **0.35 (0.75)** | 0.74 (1.38) | 0.76 (0.06) | 0.22 (0.65) | 0.44 (0.50) |
| | Forecast | **0.61 (2.17)** | 0.79 (4.98) | 0.89 (0.91) | 0.95 (0.19) | 0.41 (1.53) | 1.02 (1.38) |
| | Random | 0.17 (0.64) | **0.11 (0.42)** | 0.61 (0.35) | 0.56 (0.05) | 0.02 (0.12) | 0.11 (0.10) |
| | Blackout | **0.57 (2.82)** | 0.65 (6.58) | 0.85 (3.47) | 0.86 (0.16) | 0.30 (1.19) | 0.87 (1.04) |
| HAR | Block | 0.76 (1.01) | 0.55 (3.54) | 0.65 (3.76) | **0.53 (2.45)** | 0.49 (1.57) | 0.31 (1.98) |
| | Forecast | 0.85 (3.20) | 0.88 (3.55) | 0.97 (6.41) | **0.75 (2.11)** | 0.69 (2.35) | 0.59 (2.38) |
| | Random | 0.19 (0.69) | **0.16 (1.17)** | 0.42 (0.97) | 0.30 (0.31) | 0.12 (0.62) | 0.15 (0.56) |
| | Blackout | 0.82 (2.67) | 0.78 (4.77) | 0.80 (5.72) | **0.72 (2.17)** | 0.65 (2.66) | 0.54 (1.53) |
| Metro | Block | 0.53 (1.44) | 0.79 (4.21) | 0.83 (2.58) | **0.46 (1.27)** | 0.40 (1.85) | 0.44 (3.29) |
| | Forecast | 0.60 (1.05) | 0.92 (3.87) | 0.89 (11.59) | **0.59 (2.01)** | 0.56 (5.45) | 0.58 (2.27) |
| | Random | 0.19 (1.37) | 0.27 (0.88) | 0.59 (6.00) | **0.14 (0.28)** | 0.09 (0.42) | 0.14 (0.59) |
| | Blackout | 0.52 (4.48) | 0.82 (4.60) | 0.84 (2.75) | **0.51 (2.19)** | 0.47 (4.14) | 0.52 (3.82) |
| TEP | Block | 0.98 (0.65) | **0.68 (1.88)** | 0.84 (1.97) | 0.73 (0.37) | 0.60 (0.69) | 0.94 (1.49) |
| | Forecast | 1.11 (2.84) | 1.11 (4.07) | 1.10 (2.79) | **0.96 (0.60)** | 0.92 (1.15) | 1.56 (2.91) |
| | Random | 0.95 (2.54) | **0.50 (0.70)** | 0.71 (1.38) | 0.60 (0.44) | 0.44 (0.29) | 1.22 (1.45) |
| | Blackout | 1.05 (2.14) | 0.96 (1.78) | 1.01 (3.24) | **0.89 (0.66)** | 0.82 (1.38) | 1.53 (3.15) |
| Air Quality | Block | 0.57 (1.31) | 0.57 (2.84) | 0.79 (2.60) | **0.53 (0.54)** | 0.38 (0.91) | 0.47 (1.45) |
| | Forecast | 0.69 (3.59) | 0.87 (7.86) | 1.00 (3.82) | **0.68 (1.54)** | 0.68 (1.71) | 0.70 (1.28) |
| | Random | **0.28 (1.14)** | 0.29 (0.77) | 0.59 (2.25) | 0.39 (0.10) | 0.17 (1.26) | 0.28 (0.56) |
| | Blackout | 0.69 (3.17) | 0.84 (6.38) | 0.99 (4.45) | **0.66 (1.46)** | 0.65 (5.14) | 0.70 (3.30) |

The MSE results in Table 4 further show that VIGIL remains competitive with the centralised inference-time methods, following a similar overall trend as for CRPS. Compared with Data Guidance and Diffusion-TS, VIGIL performs only $4.0\%$ and $1.1\%$ worse on average, respectively, while outperforming TSDiff by $24.1\%$. The conditionally trained baselines are generally stronger, although the gap differs substantially between them: VIGIL performs $83.8\%$ worse than CSDI but only $12.7\%$ worse than SSSD-S4. CSDI achieves lower MSE in all 20 settings, whereas VIGIL matches or outperforms SSSD-S4 in 9 of the 20 settings.

### H.2 SPINNING DECODER

Spinning Decoder serves as a reference for the quality achievable when the observation constraint is evaluated directly in the data space throughout reverse diffusion. At each refinement, the current latent estimate is sent to the client, decoded, compared against the observed values, and the resulting correction is returned to the server. This provides a strong guidance signal, but requires repeated client–server interaction during sampling.

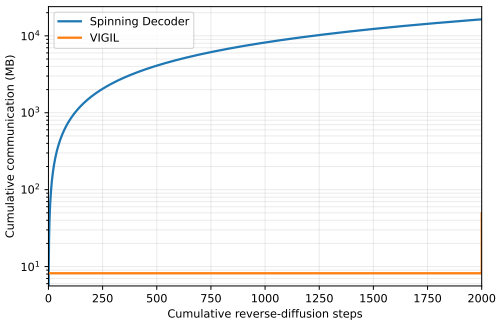


Figure 5: Cumulative communication over 10 imputations during reverse diffusion. Spinning Decoder repeatedly exchanges information with the client throughout sampling, whereas VIGIL incurs an essentially fixed communication cost.

Fig. 5 highlights the communication advantage of our approach. The reported communication volume is aggregated over the full evaluation set of 500 sequences, each with 10 imputations, corresponding to 5,000 generated sequences in total. For the 10 imputations reported here, Spinning Decoder requires client participation throughout all 2000 reverse-diffusion steps, whereas VIGIL only communicates once at the beginning of the request and once at the end to return the final imputations. As a result, Spinning Decoder incurs

Table 5: Comparison between our latent-anchor guidance and Spinning Decoder. Mean CRPS is reported over five inference seeds, with standard deviations in parentheses in units of $10^{-3}$ and $\Delta = \mathrm{CRPS}_{\mathrm{VIGIL}} - \mathrm{CRPS}_{\mathrm{Spinning}}$. The best (unrounded) result in each setting is shown in bold.

| Dataset | Scenario | VIGIL (Ours) | Spinning Decoder | Δ CRPS |
|---|---|---|---|---|
| Appliances | Block | 0.43 (0.64) | **0.32 (0.27)** | +0.11 |
| | Forecast | 0.51 (0.78) | **0.47 (0.74)** | +0.04 |
| | Random | 0.25 (0.48) | **0.09 (0.08)** | +0.16 |
| | Blackout | 0.48 (0.80) | **0.41 (0.40)** | +0.07 |
| HAR | Block | 0.68 (0.39) | **0.45 (0.93)** | +0.23 |
| | Forecast | 0.69 (1.61) | **0.57 (1.62)** | +0.12 |
| | Random | 0.32 (0.47) | **0.26 (0.29)** | +0.06 |
| | Blackout | 0.69 (1.37) | **0.52 (1.41)** | +0.17 |
| Metro | Block | 0.56 (1.15) | **0.49 (1.00)** | +0.07 |
| | Forecast | 0.58 (0.79) | **0.57 (1.11)** | +0.01 |
| | Random | 0.30 (0.98) | **0.20 (0.19)** | +0.10 |
| | Blackout | 0.53 (1.71) | **0.51 (1.56)** | +0.02 |
| TEP | Block | **0.76 (0.35)** | 0.79 (0.29) | -0.03 |
| | Forecast | **0.82 (1.08)** | 0.84 (0.67) | -0.02 |
| | Random | 0.74 (1.30) | **0.71 (0.17)** | +0.03 |
| | Blackout | **0.79 (0.88)** | 0.84 (0.38) | -0.05 |
| Air Quality | Block | 0.45 (0.67) | **0.41 (0.42)** | +0.04 |
| | Forecast | **0.52 (2.69)** | 0.52 (0.88) | 0.00 |
| | Random | 0.28 (0.51) | **0.21 (0.11)** | +0.07 |
| | Blackout | **0.50 (1.21)** | 0.50 (0.70) | 0.00 |

communication that grows throughout sampling, while the communication cost of VIGIL remains essentially constant.

Table 5 compares this interactive procedure with our latent-anchor guidance. Despite avoiding repeated decoding and communication, our method remains close to Spinning Decoder in many settings. The gap is particularly small on Metro forecasting and blackout, as well as on Air Quality forecasting and blackout, and our method even outperforms Spinning Decoder on three of the four TEP scenarios. Larger gaps appear on some settings, most notably Appliances random and HAR block, showing the cost of replacing repeated exact data-space corrections with a fixed latent-space approximation. Overall, the comparison shows that the proposed latent guidance retains much of the imputation performance of the considerably more expensive interactive alternative while removing the need for client involvement during reverse diffusion.

### H.3 ANCHOR CONSTRUCTION

We compare our optimisation-based anchor against two alternative anchor constructions. The *interpolation* anchor is obtained by first filling the missing values through linear interpolation and directly encoding the resulting complete sequence, without subsequent latent optimisation. We additionally consider an *oracle* anchor obtained by encoding the complete ground-truth sequence, including the values hidden by the evaluation mask. The oracle is unavailable in practice, but provides a privileged reference whose latent representation is informed by the true values in the missing regions. Importantly, it is not necessarily optimal for reconstructing the observed entries or for downstream guided sampling, since it remains subject to autoencoder reconstruction error and is not explicitly optimised to satisfy the observed constraints.

Table 6a evaluates how closely the decoded anchors reproduce the observations that are actually available to the client. The optimised anchor achieves the lowest observed-entry MSE in every dataset and missingness scenario, with substantial reductions relative to both direct interpolation and the oracle encoding. This is consistent with the purpose of the optimisation procedure: the latent is explicitly optimised through the frozen decoder to reproduce the known observations, whereas the interpolation and oracle anchors are obtained by a single encoding pass and therefore inherit the reconstruction error of the autoencoder. Accurate reconstruction is particularly important for the

Table 6: Comparison of three latent-anchor constructions over 500 evaluation sequences under four $50\%$ missingness scenarios. (a) Data-space reconstruction MSE on the observed entries. (b) Imputation CRPS using the same variance-based latent guidance for all variants, reported as means over five inference seeds, with standard deviations in parentheses in units of $10^{-3}$. Bold values denote the lowest mean error; underlined values denote the second-lowest CRPS, with rankings based on unrounded means.

| Dataset | Scenario | Optimised (Ours) | Interpolation | Oracle |
|---|---|---|---|---|
| Appliances | Block | **0.0113** | 0.0872 | 0.0821 |
| | Forecast | **0.0153** | 0.0803 | 0.0802 |
| | Random | **0.0172** | 0.0799 | 0.0815 |
| | Blackout | **0.0155** | 0.0813 | 0.0823 |
| HAR | Block | **0.0330** | 0.2908 | 0.1981 |
| | Forecast | **0.0328** | 0.2390 | 0.1879 |
| | Random | **0.0291** | 0.1800 | 0.1905 |
| | Blackout | **0.0310** | 0.2486 | 0.1923 |
| Metro | Block | **0.0023** | 0.0275 | 0.0272 |
| | Forecast | **0.0014** | 0.0224 | 0.0220 |
| | Random | **0.0004** | 0.0145 | 0.0240 |
| | Blackout | **0.0024** | 0.0271 | 0.0270 |
| TEP | Block | **0.0481** | 0.2322 | 0.1300 |
| | Forecast | **0.0479** | 0.1099 | 0.1057 |
| | Random | **0.0287** | 0.1571 | 0.1234 |
| | Blackout | **0.0540** | 0.1360 | 0.1293 |
| Air Quality | Block | **0.0684** | 0.1410 | 0.1323 |
| | Forecast | **0.0818** | 0.1451 | 0.1437 |
| | Random | **0.0701** | 0.1207 | 0.1399 |
| | Blackout | **0.0775** | 0.1390 | 0.1382 |

(a) Observed-entry reconstruction MSE.

| Dataset | Scenario | Optimised (Ours) | Interpolation | Oracle |
|---|---|---|---|---|
| Appliances | Block | 0.43 (0.64) | 0.42 (0.59) | **0.39 (0.62)** |
| | Forecast | 0.51 (0.78) | 0.50 (0.71) | **0.49 (0.77)** |
| | Random | **0.25 (0.48)** | 0.27 (0.31) | 0.27 (0.35) |
| | Blackout | 0.48 (0.80) | 0.48 (0.61) | **0.46 (0.53)** |
| HAR | Block | 0.68 (0.39) | 0.76 (0.21) | **0.33 (0.15)** |
| | Forecast | 0.69 (1.61) | 0.72 (0.99) | **0.38 (1.31)** |
| | Random | **0.32 (0.47)** | 0.37 (0.30) | 0.34 (0.31) |
| | Blackout | 0.69 (1.37) | 0.72 (1.02) | **0.35 (0.62)** |
| Metro | Block | 0.56 (1.15) | 0.57 (1.37) | **0.33 (1.61)** |
| | Forecast | 0.58 (0.79) | 0.58 (0.72) | **0.45 (0.65)** |
| | Random | 0.30 (0.98) | 0.30 (0.83) | **0.26 (0.82)** |
| | Blackout | 0.53 (1.71) | 0.53 (1.93) | **0.43 (1.98)** |
| TEP | Block | 0.76 (0.35) | 0.75 (0.47) | **0.66 (0.19)** |
| | Forecast | 0.82 (1.08) | 0.81 (0.89) | **0.78 (0.87)** |
| | Random | 0.74 (1.30) | 0.70 (0.85) | **0.65 (1.05)** |
| | Blackout | 0.79 (0.88) | 0.78 (0.60) | **0.76 (0.57)** |
| Air Quality | Block | 0.45 (0.67) | 0.45 (0.30) | **0.27 (1.09)** |
| | Forecast | 0.52 (2.69) | 0.51 (2.00) | **0.46 (1.61)** |
| | Random | 0.28 (0.51) | 0.24 (0.54) | **0.22 (0.42)** |
| | Blackout | 0.50 (1.21) | 0.48 (0.88) | **0.44 (0.99)** |

(b) Imputation CRPS.

observed regions, since these are the constraints that latent guidance should preserve. In contrast, the unobserved regions should not be forced to reproduce the arbitrary completion used to construct the anchor; the confidence map instead attenuates guidance where the anchor is uncertain, leaving the diffusion prior free to generate plausible missing values.

However, Table 6b shows that improved anchor optimisation does not always translate into a large difference in final imputation quality. Simple interpolation remains extremely close to the optimised anchor in many settings, and where interpolation performs better, the improvement is generally marginal, indicating that the proposed optimisation procedure does not meaningfully degrade performance when the simpler anchor is already sufficient. The oracle results provide further insight into when anchor optimisation is useful: HAR exhibits the largest gap between interpolation and the oracle anchor, suggesting that interpolation provides a comparatively poor approximation of the desired latent target on this dataset. In precisely these settings, the optimised anchor consistently improves over interpolation across all four missingness scenarios. This suggests that explicit anchor optimisation becomes more valuable when a simple interpolation-based initialisation is insufficient, while introducing little risk of meaningful degradation when interpolation already provides a strong latent target. Since anchor construction accounts for only a small fraction of the overall inference time in our implementation (Appendix H.4), this additional optimisation provides a low-cost opportunity to obtain meaningful gains in settings where interpolation is inadequate.

Notably, on Appliances and HAR under random missingness, the optimised anchor slightly outperforms the oracle anchor in final CRPS. This is not contradictory to the oracle having access to the complete sequence: the oracle is obtained by directly encoding the ground truth and is therefore still affected by autoencoder reconstruction error, whereas the optimised anchor explicitly optimises the decoded representation to satisfy the observed values. When observations are densely distributed throughout the sequence, this stronger agreement with the available constraints can provide a more effective guidance target despite the oracle's privileged access to the hidden values.

### H.4 TIMING

Table 7 shows that VIGIL is substantially faster than the conditional inference diffusion baselines across all 20 dataset–scenario combinations. This is particularly notable because, like these base-

Table 7: End-to-end method runtime over five inference seeds. Values are mean runtime in seconds, with standard deviations across seeds reported in parentheses.

| Dataset | Scenario | VIGIL (Ours) | Spinning Decoder | Data Guidance | TSDiff | Diff.-TS | CSDI | SSSD-S4 |
|---|---|---|---|---|---|---|---|---|
| Appliances | Block | 229.43 (0.57) | 1412.76 (0.78) | 2191.84 (0.62) | 7020.27 (8.62) | 1124.71 (0.71) | 277.12 (0.06) | 633.76 (4.96) |
| | Forecast | 229.16 (0.44) | 1412.21 (0.12) | 2192.36 (0.59) | 7024.61 (4.43) | 1124.80 (0.61) | 277.18 (0.05) | 632.95 (6.31) |
| | Random | 229.54 (0.21) | 1412.42 (0.29) | 2192.97 (0.52) | 7022.10 (5.11) | 1125.19 (0.69) | 277.25 (0.17) | 628.25 (2.68) |
| | Blackout | 229.76 (0.40) | 1412.22 (0.42) | 2191.21 (0.57) | 7020.70 (8.34) | 1124.87 (1.33) | 277.16 (0.02) | 631.38 (3.28) |
| HAR | Block | 223.34 (0.45) | 944.34 (0.63) | 1529.11 (0.55) | 4946.17 (4.61) | 741.96 (0.53) | 82.84 (0.05) | 622.21 (6.02) |
| | Forecast | 223.53 (0.48) | 943.88 (0.50) | 1529.47 (0.58) | 4949.73 (3.77) | 742.15 (1.49) | 82.84 (0.06) | 620.73 (2.57) |
| | Random | 223.36 (0.52) | 943.97 (0.43) | 1529.92 (0.65) | 4948.67 (8.66) | 741.41 (0.70) | 82.86 (0.04) | 622.88 (3.66) |
| | Blackout | 223.27 (0.37) | 943.40 (0.38) | 1528.58 (0.49) | 4945.09 (7.08) | 741.98 (0.58) | 82.84 (0.07) | 624.97 (4.81) |
| Metro | Block | 229.13 (0.48) | 1409.16 (0.49) | 2189.32 (1.19) | 7030.26 (8.41) | 1128.26 (0.47) | 83.24 (0.08) | 618.06 (7.26) |
| | Forecast | 228.58 (0.42) | 1408.87 (0.47) | 2189.61 (1.28) | 7026.12 (7.83) | 1126.97 (1.75) | 83.23 (0.04) | 620.93 (7.40) |
| | Random | 228.95 (1.07) | 1409.07 (0.20) | 2189.96 (1.52) | 7030.34 (4.53) | 1127.12 (1.54) | 83.30 (0.15) | 621.01 (5.38) |
| | Blackout | 228.85 (0.64) | 1409.15 (0.53) | 2188.96 (1.07) | 7027.32 (7.34) | 1127.44 (0.87) | 83.24 (0.07) | 619.77 (3.95) |
| TEP | Block | 219.17 (0.49) | 561.69 (0.37) | 838.13 (0.42) | 2739.84 (6.49) | 684.54 (0.84) | 165.94 (0.04) | 630.64 (6.36) |
| | Forecast | 218.87 (0.44) | 561.35 (0.39) | 838.47 (0.46) | 2741.01 (3.39) | 684.81 (0.56) | 165.92 (0.05) | 630.01 (5.26) |
| | Random | 219.53 (0.42) | 561.55 (0.52) | 838.81 (0.58) | 2740.84 (7.53) | 685.17 (0.69) | 165.99 (0.03) | 628.97 (5.05) |
| | Blackout | 218.97 (0.31) | 561.41 (0.33) | 837.78 (0.33) | 2736.87 (6.85) | 685.31 (0.44) | 165.95 (0.08) | 628.82 (3.23) |
| Air Quality | Block | 229.02 (0.46) | 1411.07 (0.52) | 2192.29 (0.67) | 7024.80 (5.89) | 1127.02 (2.34) | 133.79 (0.10) | 622.64 (2.26) |
| | Forecast | 229.26 (0.11) | 1410.61 (0.43) | 2192.48 (0.71) | 7025.45 (3.65) | 1127.70 (1.18) | 133.78 (0.08) | 626.19 (6.63) |
| | Random | 229.43 (0.28) | 1410.35 (0.64) | 2192.68 (0.90) | 7021.37 (6.22) | 1128.34 (1.29) | 133.74 (0.06) | 625.98 (7.47) |
| | Blackout | 229.30 (0.58) | 1410.69 (0.70) | 2192.11 (0.57) | 7023.05 (6.48) | 1127.88 (0.82) | 133.76 (0.07) | 619.64 (3.71) |

lines, VIGIL performs conditioning entirely at inference time and does not require retraining the diffusion model for the conditional task. Despite this additional inference-time conditioning, VIGIL is also faster than SSSD-S4 across all evaluated settings, even though SSSD-S4 learns its conditioning mechanism during training and would therefore be expected to incur less conditional-processing overhead at inference time.

CSDI provides a different reference point because it is trained explicitly for conditional imputation and can therefore incorporate the conditioning information directly through its trained model at inference time. Its lower runtime in many settings is therefore expected. Nevertheless, VIGIL remains competitive in several cases and is faster than CSDI across all four Appliances scenarios, while retaining the flexibility of an unconditional diffusion backbone whose conditioning constraints are introduced only at inference time.

Table 8: Runtime decomposition of our method over five inference seeds. Values are mean runtime in seconds. Anchor and latent-mask estimation are performed locally on the client, while guided diffusion is performed on the server.

| Dataset | Scenario | Anchor Estimation | Latent Mask Estimation | Diffusion | Total |
|---|---|---|---|---|---|
| Appliances | Block | 12.63 | 1.55 | 215.24 | 229.43 |
| | Forecast | 12.63 | 1.55 | 214.98 | 229.16 |
| | Random | 12.64 | 1.55 | 215.35 | 229.54 |
| | Blackout | 12.64 | 1.55 | 215.57 | 229.76 |
| HAR | Block | 7.45 | 1.00 | 214.89 | 223.34 |
| | Forecast | 7.45 | 1.00 | 215.08 | 223.53 |
| | Random | 7.45 | 1.00 | 214.91 | 223.36 |
| | Blackout | 7.45 | 1.00 | 214.82 | 223.27 |
| Metro | Block | 12.11 | 1.54 | 215.48 | 229.13 |
| | Forecast | 12.10 | 1.54 | 214.94 | 228.58 |
| | Random | 12.11 | 1.54 | 215.30 | 228.95 |
| | Blackout | 12.11 | 1.54 | 215.20 | 228.85 |
| TEP | Block | 4.20 | 0.46 | 214.51 | 219.17 |
| | Forecast | 4.20 | 0.46 | 214.20 | 218.87 |
| | Random | 4.21 | 0.46 | 214.86 | 219.53 |
| | Blackout | 4.20 | 0.46 | 214.31 | 218.97 |
| Air Quality | Block | 12.25 | 1.54 | 215.22 | 229.02 |
| | Forecast | 12.26 | 1.54 | 215.46 | 229.26 |
| | Random | 12.26 | 1.54 | 215.63 | 229.43 |
| | Blackout | 12.25 | 1.54 | 215.51 | 229.30 |

Table 8 further shows that the computationally expensive component of VIGIL is intentionally placed on the server. Guided latent diffusion remains nearly constant at approximately 214–216 s across

all datasets and missingness scenarios and accounts for the majority of the end-to-end runtime. In contrast, the client-side preparation is comparatively lightweight. Latent-mask estimation requires at most approximately 1.6 s, while anchor optimisation requires only a small fraction of the overall runtime. The optimised latent anchor and confidence map are then communicated to the server, where the computationally intensive generative process is performed.

Overall, the decomposition demonstrates that VIGIL preserves the intended client–server division of computation: only lightweight conditioning information is constructed locally, while the substantially more expensive diffusion process is outsourced to the server.

### H.5 QUALITATIVE RESULTS

Fig. 6 provides qualitative examples of VIGIL under the four evaluated missingness scenarios. The generated trajectories are compared with the ground-truth sequence, complementing the aggregate CRPS and MSE results with a direct view of the reconstructed temporal dynamics.

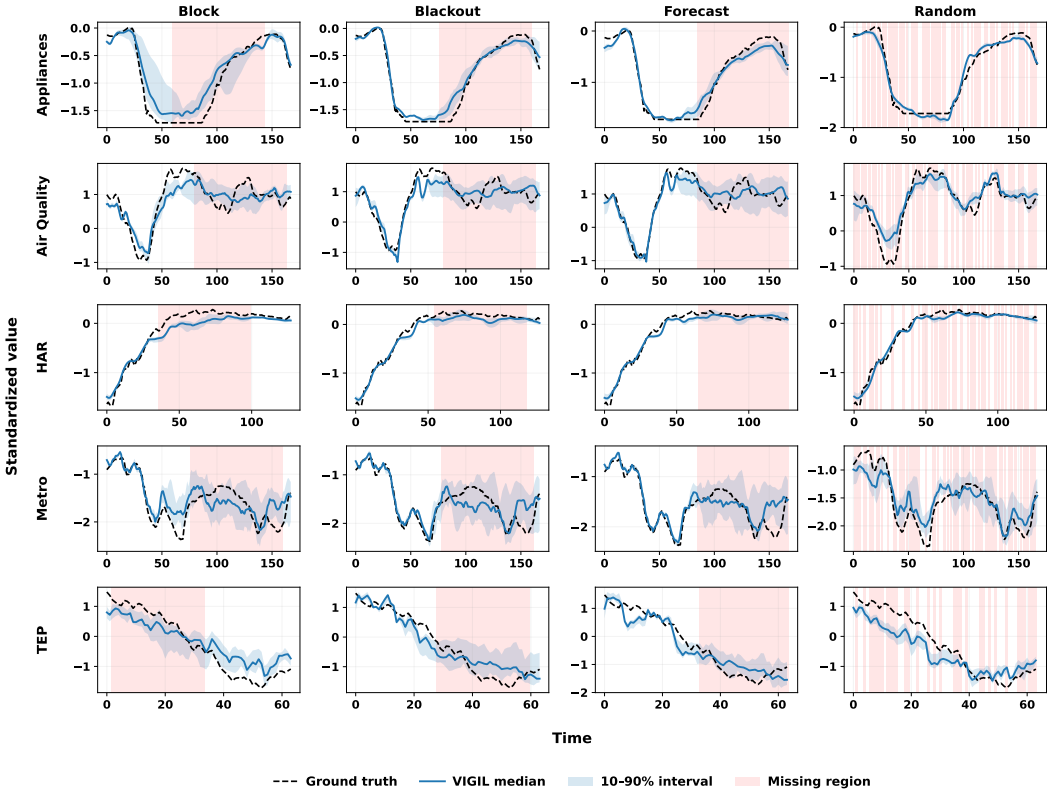


Figure 6: Qualitative examples of VIGIL under the evaluated missingness scenarios. The blue line shows the median prediction, the shaded band the 10–90% interval over 10 imputations, the dashed line the ground truth, and red shading the missing region.

Across the examples, the decoded trajectories generally follow the ground-truth signal in the observed regions. This indicates that the optimised latent anchor and confidence-based guidance are able to preserve most of the information contained in the available observations, despite the diffusion process operating entirely in latent space. At the same time, the generated trajectories also remain close to the ground truth within the hidden regions. This shows that the method does not merely reproduce the observed constraints, but that the latent diffusion model is able to generate coherent values for the unobserved part of the sequence. Together, the agreement in both observed and missing regions illustrates the complementary roles of the latent guidance mechanism and the learned diffusion prior.