ALTERNATORS WITH NOISE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Alternators have recently been introduced as a framework for modeling time-dependent data. They often outperform other popular frameworks, such as state-space models and diffusion models, on challenging time-series tasks. This paper introduces a new Alternator model, called **Alternator++**, which enhances the flexibility of traditional Alternators by explicitly modeling the noise terms used to sample the latent and observed trajectories, drawing on the idea of noise models from the diffusion modeling literature. Alternator++ optimizes the sum of the Alternator loss and a noise-matching loss. The latter forces the noise trajectories generated by the two noise models to approximate the noise trajectories that produce the observed and latent trajectories. We demonstrate the effectiveness of Alternator++ in tasks such as density estimation, time series imputation, and forecasting, showing that it outperforms several strong baselines, including Mambas, ScoreGrad, and Dyffusion.

1 INTRODUCTION

Modeling complex time-dependent data is a central challenge in science and engineering. Recent advancements in sequence modeling are based on two popular frameworks: structured state-space models (SSMs) such as Mamba (Gu & Dao, 2023) and diffusion models (Ho et al., 2020; Rasul et al., 2021). These approaches have been successfully applied across various domains, including natural language processing (Gu & Dao, 2023), computer vision (Zhu et al., 2024; Rombach et al., 2022), and computational biology (Xu et al., 2024). They provide powerful tools for sequence modeling by capturing complex dependencies and offering strong generative capabilities.

Despite these successes, SSMs and diffusion models face significant challenges. They employ hidden representations that have the same dimensionality as the data, which leads to large models with high computational training costs. Furthermore, Mamba struggles with capturing long-range dependencies in noisy signals due to its reliance on structured state transitions in its network architecture (Wang et al., 2025). These state transitions can be affected by noise that propagates through time, which can be limiting when processing highly noisy time-series (Wang et al., 2025; Rezaei & Dieng, 2025). Diffusion models, on the other hand, are notably slow to generate new data from, with significant research dedicated to accelerating their sampling process (Song et al., 2020; Vahdat et al., 2021; Salimans & Ho, 2022; Lu et al., 2022; Karras et al., 2022).

Alternators have been recently introduced as an alternative framework for sequence modeling (Rezaei & Dieng, 2024). They offer a more efficient latent representation by maintaining a low-dimensional state space, reducing computational complexity while preserving expressivity. However, Alternators assume a fixed noise distribution when sampling observation and latent trajectories, which may be limiting.

In this paper, we introduce Alternator++, a new member of the Alternator class of models that uses trainable noise models instead of fixed probability distributions to define the noise terms used to generate observation and latent trajectories. Noise models have proven to be very beneficial for diffusion models (Dhariwal & Nichol, 2021; Ho et al., 2022; Nichol & Dhariwal, 2021); they improve the quality of the generated outputs (Ho et al., 2020; Song et al., 2020), enable stable training (Lin et al., 2024; Chen, 2023), and enhance model robustness (Lee et al., 2024). Alternator++ inherits these advantages while efficiently generating observation and latent trajectories following the Alternator framework. More specifically, while the noise terms in the original Alternator had zero means, leveraging noise models lifts that restriction and allows us to learn the mean of the noise

variables instead. These means are modeled using two neural networks, which are trained by adding a noise-matching objective in the Alternator loss.

Through comprehensive experiments across multiple datasets and domains, we demonstrate that Alternator++ consistently outperforms Mamba, diffusion models, and the original Alternator on density estimation, time-series imputation, and forecasting.

2 BACKGROUND

Here we provide background on the two foundations of Alternator++: Alternators and Diffusion models.

2.1 DIFFUSION MODELS

Diffusion models are a powerful approach to generative modeling. The framework consists of two processes: the forward (diffusion) process progressively adds noise to the observations, while the reverse (denoising) process removes the noise from the observations.

Forward diffusion and reverse denoising processes. Let $\boldsymbol{x}_0 \in \mathbb{R}^{D_x}$ be a data point. The forward diffusion process of a diffusion model is a Markov chain which iteratively adds Gaussian noise to $\boldsymbol{x}_0$ until, after T iteration steps, the observation at that time step, denoted by $\boldsymbol{x}_T$, is nearly a sample from a standard Gaussian. Concretely, for a fixed schedule $\{\beta_t \in (0,1)\}_{t=1}^T$, the transition from one step to the next is characterized by the conditional distribution

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}\big(\boldsymbol{x}_t; \sqrt{1-\beta_t}\,\boldsymbol{x}_{t-1},\, \beta_t \mathbf{I}\big), \tag{1}$$

such that $\boldsymbol{x}_t$ is a noised version of $\boldsymbol{x}_{t-1}$. By defining $\alpha_t = \prod_{s=1}^t (1-\beta_s)$, one can directly relate $\boldsymbol{x}_t$ to $\boldsymbol{x}_0$ through the conditional distribution

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}\big(\boldsymbol{x}_t; \sqrt{\alpha_t}\,\boldsymbol{x}_0, (1-\alpha_t)\mathbf{I}\big). \tag{2}$$

The reverse denoising process is characterized by the conditional distribution

$$p_\theta(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \frac{1}{\sqrt{1-\beta_t}}\left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\,\epsilon_\theta(\boldsymbol{x}_t, t)\right), \beta_t \mathbf{I}\right) \tag{3}$$

where ϵ_θ is a neural network, called a *noise model*, that takes $\boldsymbol{x}_t$ and t as input.

Learning. The parameters θ described above are learned via denoising score matching. Specifically, one trains the neural network ϵ_θ to predict the noise ϵ that was added at step t by minimizing

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t, \boldsymbol{x}_0, \epsilon}\big[\|\epsilon - \epsilon_\theta(\boldsymbol{x}_t, t)\|_2^2\big] \tag{4}$$

where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\boldsymbol{x}_t = \sqrt{\alpha_t}\,\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\,\epsilon$. Minimizing this objective makes ϵ_θ an effective denoiser. Equivalently, ϵ_θ approximates the score function $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$ (up to a scaling factor) at each time t.

Sampling. Once trained, the reverse denoising process can be approximated by a discretized Langevin dynamics update:

$$\boldsymbol{x}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\Big(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1-\alpha_t}}\,\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)\Big) + \sqrt{\beta_t}\,\boldsymbol{\epsilon}, \tag{5}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t)$ is replaced by the neural network's score estimate. Each step removes a small amount of noise and adds a controlled Gaussian perturbation, ultimately yielding a fully denoised generated sample $\boldsymbol{x}_0$.

2.2 ALTERNATORS

Consider a sequence $\boldsymbol{x}_{1:T}$. An Alternator models this sequence by pairing it with latent variables $\boldsymbol{z}_{0:T}$ in a joint distribution Rezaei & Dieng (2024):

$$p_{\theta,\phi}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{0:T}) = p(\boldsymbol{z}_0) \prod_{t=1}^{T} p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_{t-1})\, p_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x}_t). \tag{6}$$

Here $p(\boldsymbol{z}_0) = \mathcal{N}(0, \mathbf{I})$ is a prior over the initial latent variable $\boldsymbol{z}_0$ and $p_\phi(\boldsymbol{z}_t \mid \boldsymbol{z}_{t-1}, \boldsymbol{x}_t)$ models how the other latent variables are generated over time. The observations that it conditions on are modeled through $p_\theta(\boldsymbol{x}_t \mid \boldsymbol{z}_{t-1})$. Both conditional distributions are Gaussians parameterized by neural networks with parameters ϕ and θ,

$$p_\theta(\boldsymbol{x}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}\left(\boldsymbol{\mu}_{x_t}, \sigma_x^2 \mathbf{I}\right) \text{ where } \boldsymbol{\mu}_{x_t} = \sqrt{(1-\sigma_x^2)} \cdot f_\theta(\boldsymbol{z}_{t-1})$$

$$p_\phi(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}, \boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{z_t}, \sigma_z^2 \mathbf{I}\right), \text{ where } \boldsymbol{\mu}_{z_t} = \sqrt{\alpha_t} \cdot g_\phi(\boldsymbol{x}_t) + \sqrt{(1-\alpha_t-\sigma_z^2)} \cdot \boldsymbol{z}_{t-1}$$

The parameters θ and ϕ are learned by minimizing the Alternator loss

$$\mathcal{L}_{\text{Alternator}}(\theta, \phi) = \mathbb{E}_{p(\boldsymbol{x}_{1:T}) p_{\theta,\phi}(\boldsymbol{z}_{0:T})} \left[\sum_{t=1}^{T} \|\boldsymbol{z}_t - \boldsymbol{\mu}_{z_t}\|_2^2 + \frac{D_z \sigma_z^2}{D_x \sigma_x^2} \|\boldsymbol{x}_t - \boldsymbol{\mu}_{x_t}\|_2^2 \right], \tag{7}$$

where $p(\boldsymbol{x}_{1:T})$ is the data distribution and $p_{\theta,\phi}(\boldsymbol{z}_{0:T})$ is the marginal distribution of the latent variables induced by the joint distribution in Eq. 6. Alternators model sequences over time by coupling observations with latent variables, whereas diffusion models rely on iterative denoising. The Alternator++ model, introduced in the next section, extends the Alternator framework by incorporating a diffusion-based refinement step within the latent evolution process.

Alternators offer several key advantages over existing sequence modeling approaches. Unlike state-space models such as Mamba that maintain hidden representations with the same dimensionality as the data ($\mathbf{h}_t \in \mathbb{R}^{D_x}$), Alternators operate with low-dimensional latent variables ($\boldsymbol{z}_t \in \mathbb{R}^{D_z}$ where $D_z \ll D_x$), significantly reducing memory requirements and computational complexity. This efficiency gain is particularly pronounced for high-dimensional sequence data such as images or multivariate time series. Additionally, Alternators enable direct sequence generation through their alternating observation-latent dynamics, avoiding the computationally expensive iterative sampling procedures required by diffusion models that typically need hundreds of denoising steps. The coupled generative process also provides a natural mechanism for encoding new sequences into low-dimensional representations, making Alternators well-suited for both generative modeling and representation learning tasks. Furthermore, the alternating structure allows the model to maintain temporal coherence while adapting to local variations in the data, providing a balance between global consistency and local flexibility that is often challenging to achieve in purely autoregressive or fully parallel approaches.

2.3 MOTIVATION FOR ALTERNATOR++

While Alternators offer computational advantages through low-dimensional latent representations, they assume fixed zero-mean Gaussian noise with constant variance parameters σ_x^2 and σ_z^2. This imposes three critical limitations: noise is independent of system state, stochastic characteristics remain temporally constant, and systematic biases cannot be captured due to the zero-mean constraint.

These assumptions contradict real-world observations where financial markets exhibit volatility clustering (Bollerslev, 1986), biological signals show amplitude-dependent noise (Bar-Gad et al., 2002), and climate data displays seasonal uncertainty variations (Trenberth et al., 2007). Such limitations become pronounced when modeling heteroscedastic or non-stationary time series. Alternator++ addresses these constraints by replacing fixed noise with trainable models $\boldsymbol{\epsilon}_{\psi}^{\boldsymbol{t}}$ and $\boldsymbol{\epsilon}_{\nu}^{\boldsymbol{t}}$ that learn state-dependent, time-varying, and non-zero-mean stochastic patterns. This significantly expands the model's capacity to capture realistic stochastic dynamics in complex temporal data.

3 ALTERNATOR++

We now describe the generative process of Alternator++ and the objective function used to train its parameters.

3.1 GENERATIVE PROCESS

While standard Alternators model a time-indexed sequence $\boldsymbol{x}_{1:T}$ paired with latent variables $\boldsymbol{z}_{0:T}$ using fixed noise distributions as shown in equation 6, Alternator++ uses trainable noise prediction networks $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ that flexibly model stochasticity at each time step t.

The generative process begins by sampling an initial latent variable $\boldsymbol{z}_0 \sim \mathcal{N}(0, I_{D_z})$ from a standard Gaussian. Then we generate a sequence by alternating between generating observation $\boldsymbol{x}_t$ conditioned on the previous latent state $\boldsymbol{z}_{t-1}$ and updating the latent representation $\boldsymbol{z}_t$ using the previous latent state $\boldsymbol{z}_{t-1}$ and the current observation $\boldsymbol{x}_t$. The key innovation in Alternator++ lies in its explicit noise modeling through the specialized networks $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ that dynamically adjust stochasticity levels when sampling the observed and latent trajectories. Indeed, for any t, we sample $\boldsymbol{x}_t$ and $\boldsymbol{z}_t$ as

$$\boldsymbol{x}_t = \sqrt{\beta_t} \cdot f_\theta(\boldsymbol{z}_{t-1}) + \sqrt{1 - \beta_t - \sigma_x^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}) + \sigma_x \boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{x}_t}} \tag{8}$$

$$\boldsymbol{z}_t = \sqrt{\alpha_t} \cdot g_\phi(\boldsymbol{x}_t) + \sqrt{1 - \alpha_t - \sigma_z^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}, \boldsymbol{x}_t) + \sigma_z \boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{z}_t}} \tag{9}$$

Here, $\boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{x}_t}} \sim \mathcal{N}(0, I_{D_x})$ and $\boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{z}_t}} \sim \mathcal{N}(0, I_{D_z})$ are standard Gaussian noise variables. The functions f_θ and g_ϕ map latent variables and observations, respectively, as in the original Alternator framework. They are both neural networks with parameters θ and ϕ, respectively. The noise models $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ are modulated by time-dependent noise schedules $\beta_{1:T}$ and $\alpha_{1:T}$, with base variance parameters σ_x^2 and σ_z^2, respectively.

The noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ takes both $\boldsymbol{z}_{t-1}$ and $\boldsymbol{x}_t$ as inputs to drive the dynamics of $\boldsymbol{z}_t$, whereas the original Alternator used a simple interpolation of $\boldsymbol{z}_{t-1}$ to update the latent $\boldsymbol{z}_t$. Taking $\boldsymbol{x}_t$ as an additional input adds more expressivity and makes the latent variables more context-aware. Another departure from the original Alternator is the network $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$, which enhances the model's ability to capture complex and time-varying noise patterns in the observation space.

The noise schedules $\beta_{1:T}$ and $\alpha_{1:T}$ modulate the influence of the learned noise models. When $\beta_t \to 1 - \sigma_x^2$ and $\alpha_t \to 1 - \sigma_z^2$, the contributions of the noise prediction networks $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ diminish, and the generative dynamics revert to those of the original Alternator model. In contrast, as $\beta_t \to 0$, the generation of $\boldsymbol{x}_t$ becomes increasingly influenced by the learned noise model $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$. This allows the model to capture complex and time-varying noise patterns that are dependent on the latent state $\boldsymbol{z}_{t-1}$, thus enabling a richer and more flexible description of stochasticity in the observation domain. Similarly, as $\alpha_t \to 0$, the noise model $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ jointly driven by current observation $\boldsymbol{x}_t$ and previous latent variable $\boldsymbol{z}_{t-1}$ has a greater influence on the prediction of the latent variable $\boldsymbol{z}_t$.

3.2 TRAINING OBJECTIVE

The Alternator++ training objective adds a noise-matching objective to the original Alternator loss,

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{alternator}}(\theta, \phi, \psi, \nu) + \lambda \cdot \mathcal{L}_\epsilon(\theta, \phi, \psi, \nu) \tag{10}$$

$$\mathcal{L}_{\text{alternator}}(\theta, \phi, \psi, \nu) = \frac{1}{B} \sum_{b=1}^{B} \sum_{t=1}^{T} \left[\left\| \boldsymbol{z}_t^{(b)} - \boldsymbol{\mu}_{z_t^{(b)}} \right\|_2^2 + \frac{D_z \sigma_z^2}{D_x \sigma_x^2} \cdot \left\| \boldsymbol{x}_t^{(b)} - \boldsymbol{\mu}_{x_t^{(b)}} \right\|_2^2 \right]$$

$$\mathcal{L}_\epsilon(\theta, \phi, \psi, \nu) = \frac{1}{B} \sum_{b=1}^{B} \sum_{t=1}^{T} \left\| \boldsymbol{\epsilon}_z^{(b)} - \boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}^{(b)}, \boldsymbol{x}_t^{(b)}) \right\|_2^2 + \gamma_t \cdot \left\| \boldsymbol{\epsilon}_x^{(b)} - \boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}^{(b)}) \right\|_2^2$$

where $\boldsymbol{x}_t^{(b)}$ is the t^{th} observation of the b^{th} sequence in the batch, it is drawn from the training data. On the other hand $\boldsymbol{z}_t^{(b)}$ is the latent variable at time t for the b^{th} sequence in the batch, it is sampled using the generative process (8, 9) with $\boldsymbol{z}_0^{(b)} \sim \mathcal{N}(0, I_{D_z})$. The means $\boldsymbol{\mu}_{x_t}^{(b)}$ and $\boldsymbol{\mu}_{z_t}^{(b)}$ are defined as

$$\boldsymbol{\mu}_{x_t}^{(b)} = \sqrt{\beta_t} f_\theta^z(\boldsymbol{z}_{t-1}^{(b)}) + \sqrt{1 - \beta_t - \sigma_x^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}^{(b)}) \tag{11}$$

$$\boldsymbol{\mu}_{z_t}^{(b)} = \sqrt{\alpha_t} \cdot g_\phi^x(\boldsymbol{x}_t^{(b)}) + \sqrt{1 - \alpha_t - \sigma_z^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}^{(b)}, \boldsymbol{x}_t^{(b)}). \tag{12}$$

The terms $\boldsymbol{\epsilon}_x^{(b)} \sim \mathcal{N}(0, ID_x)$ and $\boldsymbol{\epsilon}_z^{(b)} \sim \mathcal{N}(0, ID_z)$ are standard Gaussian noise variables sampled for each time step and batch element. Here $\gamma_t = \frac{D_z \sigma_z^2 \alpha_t}{D_x \sigma_x^2 \beta_t}$ balances the two noise-matching

Algorithm 1: Sequence modeling with Alternator++

Inputs: Data $\boldsymbol{x}_{1:T}^{(1:n)}$, batch size B, variances σ_x^2, σ_z^2, noise schedules $\beta_{1:T}$, $\alpha_{1:T}$
Initialize model parameters θ, ϕ, ψ, ν
while *not converged* **do**
  Sample a batch of sequences $\{\boldsymbol{x}_{1:T}^{(b)}\}_{b=1}^{B}$ from the dataset
  for $b = 1, \ldots, B$ **do**
    Draw initial latent $\boldsymbol{z}_0^{(b)} \sim \mathcal{N}(0, I_{D_z})$
    for $t = 1, \ldots, T$ **do**
      Draw noise samples $\boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{z}_t}} \sim \mathcal{N}(0, I_{D_z})$ and $\boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{x}_t}} \sim \mathcal{N}(0, I_{D_x})$
      Compute $\boldsymbol{\mu}_{x_t}^{(b)} = \sqrt{\beta_t} \cdot f_\theta(\boldsymbol{z}_{t-1}^{(b)}) + \sqrt{1 - \beta_t - \sigma_x^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}^{(b)})$
      Sample observation $\boldsymbol{x}_t^{(b)} = \boldsymbol{\mu}_{x_t}^{(b)} + \sigma_x \boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{x}_t}}$
      Compute $\boldsymbol{\mu}_{z_t}^{(b)} = \sqrt{\alpha_t} \cdot g_\phi(\boldsymbol{x}_t^{(b)}) + \sqrt{1 - \alpha_t - \sigma_z^2} \cdot \boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}(\boldsymbol{z}_{t-1}^{(b)}, \boldsymbol{x}_t^{(b)})$
      Sample latent $\boldsymbol{z}_t^{(b)} = \boldsymbol{\mu}_{z_t}^{(b)} + \sigma_z \boldsymbol{\epsilon}_{\boldsymbol{\mu}_{\boldsymbol{z}_t}}$
    end
  end
  Compute loss $\mathcal{L}(\theta, \phi, \psi, \nu)$ using $(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{0:T}, \boldsymbol{\mu}_{z_{0:T}}, \boldsymbol{\mu}_{x_{1:T}})$
  Backpropagate and update parameters θ, ϕ, ψ, ν using the Adam optimizer
end

loss terms. This balancing prevents the model from prioritizing one space over the other simply due to differences in dimensionality or noise magnitude, ensuring consistent learning across both the observation and latent space noise models. Finally, λ is a hyperparameter controlling the relative importance of noise prediction. When λ is small, the model behaves more like the original Alternator, focusing on reconstruction accuracy. As λ increases, the model places greater emphasis on learning accurate noise distributions, which improves its ability to model complex stochastic patterns. Algorithm 1 summarizes the training procedure for Alternator++.

The regression term $\left\| \boldsymbol{z}_t^{(b)} - \boldsymbol{\mu}_{z_t^{(b)}} \right\|_2^2$ in $\mathcal{L}_{\text{alternator}}$ serves a critical role in ensuring that the learned latent dynamics accurately capture the intended stochastic evolution of the latent space. Without this term, the latent trajectory could drift arbitrarily from the target distribution, leading to poor generation quality and unstable training. This regression provides essential training signal for both the latent update network g_ϕ and the noise prediction network $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$, ensuring they learn complementary representations where g_ϕ captures deterministic dynamics and $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ captures state-dependent stochasticity. The term acts as a regularizer that constrains the latent variables to follow the prescribed generative process while allowing the noise models to learn complex, time-varying patterns.

3.3 Sampling and Encoding New Sequences

After training, one can sample from Alternator++ to generate new sequences by simply using the generative process described in Section 3.1. That same generative process also indicates how to encode, i.e., get the low-dimensional representation, of a new sequence $\boldsymbol{x}_{1:T}^*$: simply replace the sampled $\boldsymbol{x}_t$ with the given $\boldsymbol{x}_t^*$ and return $\boldsymbol{\mu}_{z_t}$ for $t \in \{1, \ldots, T\}$.

4 Experiments

In this section, we present a comprehensive evaluation of Alternator++ across multiple time-series datasets and tasks. Our experiments aim to answer the following questions:

- How well does Alternator++ capture complex temporal dependencies and multimodal densities in time-series data compared to existing dynamical generative models?
- Can Alternator++ effectively handle missing values and outperform state-of-the-art methods in time-series imputation?

Table 1: Alternator++ tends to outperform several strong baselines, and by a wide margin. Here, performance is measured in terms of the MMD between the distribution learned by each model and the ground truth distribution, using generated samples from the models and the data.

Method	Solar	Covid	Fred	NN5
Alternator++	**0.051±0.004**	0.043±0.031	**0.039±0.005**	**0.088±0.008**
Alternator	0.123±0.002	0.592±0.063	0.281±0.002	0.310±0.002
Alpha-Alternator	0.080±0.001	0.060±0.015	0.070±0.001	0.120±0.015
Mamba	0.131±0.001	**0.025±0.052**	0.185±0.003	0.253±0.021
DiffWave	0.097±0.007	0.050±0.012	0.102±0.008	0.150±0.010
ScoreGrad	0.115±0.003	0.573±0.012	0.142±0.020	0.155±0.009
VRNN	0.848±0.005	1.106±0.002	1.328±0.005	2.109±0.001
SRNN	1.013±0.030	1.240±0.001	1.367±0.003	2.480±0.002
NODE-RNN	0.132±0.013	0.621±0.081	0.479±0.127	0.427±0.103
TimeGrad	0.140±0.010	0.630±0.030	0.170±0.010	0.200±0.020

- Does Alternator++ demonstrate superior forecasting accuracy, particularly in challenging real-world applications such as sea surface temperature prediction?

To systematically address these questions, we compare Alternator++ against widely recognized baselines, including VRNN, SRNN, NODE-RNN, ScoreGrad, TimeGrad, DiffWave, Mamba, and Dyffusion. Our results demonstrate that Alternator++ tends to outperform these baselines across multiple datasets. Notably, it captures time-series distributions better as evidenced by its lower maximum mean discrepancy (MMD) scores. Furthermore, Alternator++ can perform well at imputation even when the missing rate is very high. Finally, it also performs well at forecasting, while being significantly more computationally efficient. The following sections provide a detailed breakdown of these findings. For comprehensive details regarding implementation specifics and hyperparameter configurations across each experiment, we refer the reader to the Appendix C.

4.1 Density Estimation

To evaluate how well Alternator++ captures the underlying probability distribution of time-series data, we perform density estimation by generating samples from the trained model and comparing them to the ground truth distribution. Specifically, after training Alternator++ on a dataset, we generate synthetic sequences by sampling from the learned generative process described in Section 3.1. We then compute the Maximum Mean Discrepancy (MMD) between the generated samples and the true data distribution using a Gaussian RBF kernel. MMD provides a non-parametric measure of distributional distance that is particularly well-suited for high-dimensional time-series data (Gretton et al., 2012). Lower MMD values indicate that the generated samples more closely match the true data distribution, demonstrating better density modeling capability.

We benchmark Alternator++ against Alternators (Rezaei & Dieng, 2024), ScoreGrad (Yan et al., 2021), Mamba (Gu & Dao, 2023), VAE-based models (VRNN Chung et al. (2015), SRNN Fraccaro et al. (2016)), and Neural ODE-based models (NODE-RNN Chen et al. (2019)) in modeling the underlying probability distribution of time-series datasets. Table 1 summarizes the results of this experiment.

Alternator++ achieves the lowest MMD scores on three of the four datasets, outperforming the previous best method, ScoreGrad, by 66% on Solar, 72% on Fred, and 47% on NN5. On the Covid dataset, Mamba exceeds Alternator++ by 42%, albeit with greater variability. Among the baselines, ScoreGrad consistently beats Mamba—particularly on NN5 and Fred, where it reduces MMD by 40% and 24%, respectively—demonstrating its superior generalization across diverse time-series distributions.

These quantitative findings are corroborated in Figure 1. Alternator++'s samples align more closely with the target distribution on three out of four datasets. In highly skewed cases like Solar and NN5, it captures the distribution mode more accurately than any competitor. On the Covid and Fred datasets, Alternator++ correctly identifies both modes and assigns probability mass appropriately. The sole exception is the Covid dataset, where Mamba achieves better alignment.

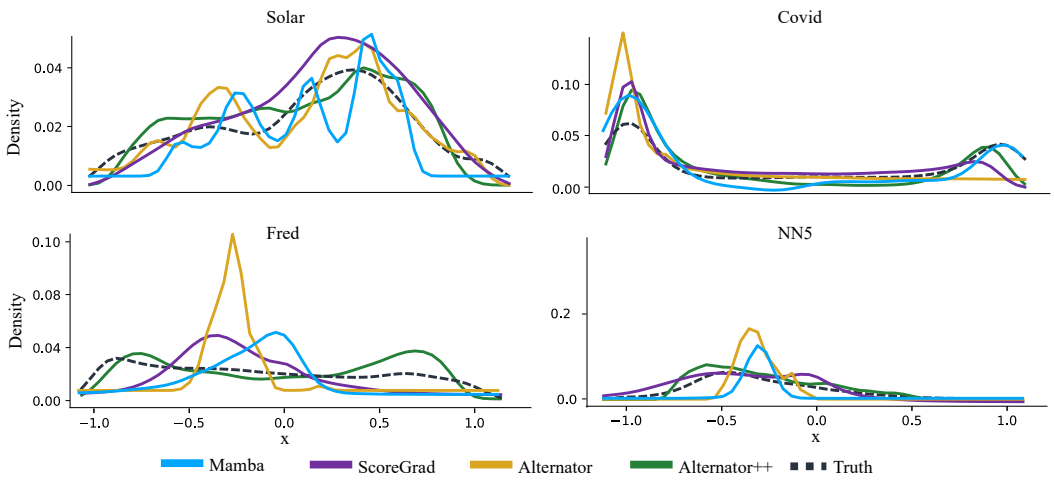

Figure 1: Comparing the distributions learned by various models against the ground truth distribution. Alternator++ captures multimodal distributions better than Alternator, Mamba, and ScoreGrad.

4.2 Time Series Imputation

Time series imputation addresses scenarios where temporal observations contain missing values due to sensor malfunctions, transmission failures, or non-uniform sampling. We evaluate model robustness by varying the Missing At Random (MAR) rates from 10% to 90%. The results are summarized in Figure 2. Note, we did not use ScoreGrad for imputation because it uses diffusion-like processes optimized for unconditionally generating new samples from learned distributions. However, adapting this framework for imputation would require significant architectural modifications to the model investigated in the previous section to handle conditioning on partial observations. Additionally, imputation with ScoreGrad requires computationally expensive iterative sampling procedures per time step, which would be prohibitive for systematic evaluation across multiple missing rates from 10% to 90%. For these reasons, we excluded ScoreGrad from the imputation experiment.

On the Solar dataset, Alternator++ outperforms both Mamba and the original Alternator in mean absolute error (MAE), improving by over 20% and 10%, respectively. In mean squared error (MSE), Alternator++ reduces error by roughly 50%, and its correlation coefficient is about 10% higher. On the FRED dataset, Alternator++ again outperforms the baselines, achieving the lowest MAE, a substantially reduced MSE, and a correlation coefficient that exceeds competing methods by approximately 10%. For NN5, Alternator++ maintains the best MAE, albeit with smaller margins, and consistently superior MSE and correlation. Finally, on the Covid dataset, Alternator++ outperforms the original Alternator in both MSE and correlation, though Mamba performs better on this dataset.

In summary, across Solar, FRED, and NN5, Alternator++ consistently achieves the lowest errors and highest correlations, demonstrating robust performance under varying patterns of missing data. Compared to the original Alternator, these results reflect clear gains in both accuracy and alignment with the true time series.

4.3 Sea surface temperature forecasting

In climate science, sea surface temperature (SST) prediction is crucial for weather forecasting and climate modeling (Haghbin et al., 2021). We apply Alternator++ to forecast SST using a daily dataset from 1982-2021, with data split into training (1982-2019, 15,048 samples), validation (2020, 396 samples), and testing (2021, 396 samples). Following (Cachay et al., 2023), we transform the global data into 60×60 (latitude$\times$longitude) tiles, selecting 11 patches in the eastern tropical Pacific Ocean for forecasting horizons of 1-7 days.

We compare against Alternators (Rezaei & Dieng, 2024), DDPM (Ho et al., 2020), MCVD (Voleti et al., 2022), DDPM variants (DDPM-D (Gal & Ghahramani, 2016) and DDPM-P (Pathak et al.,

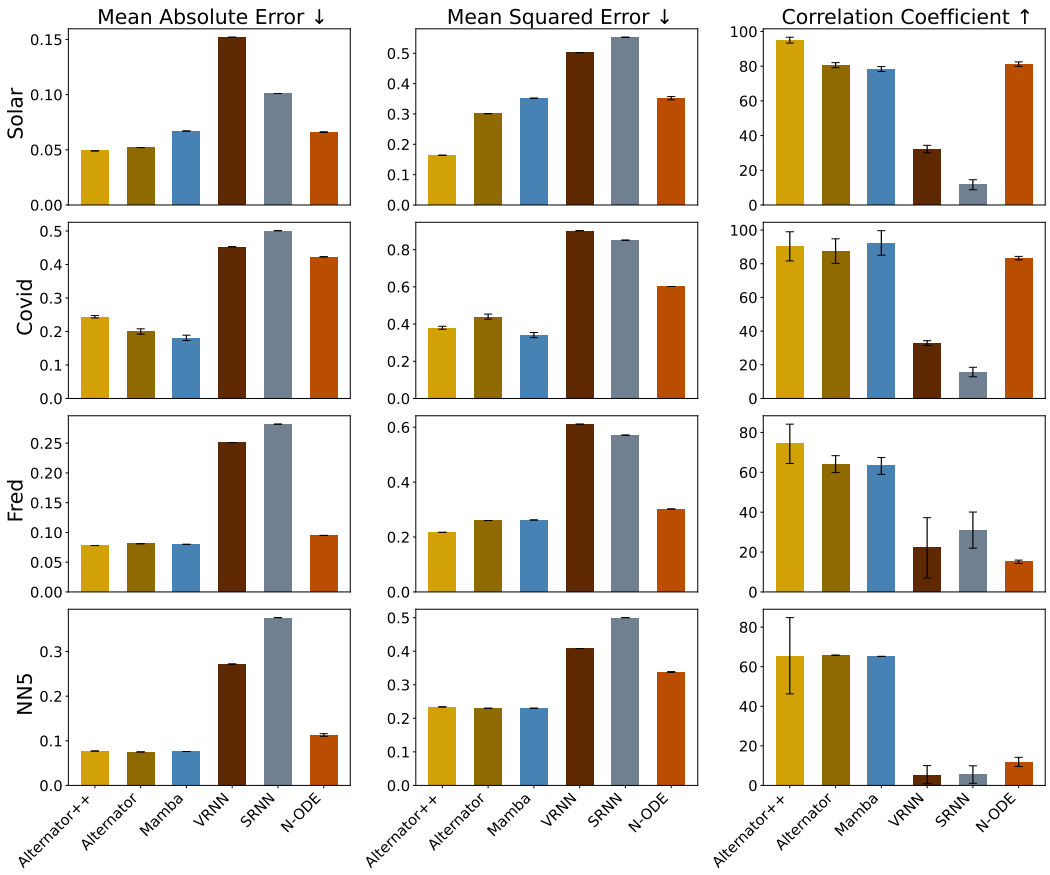

Figure 2: Performance on missing data imputation across several datasets, evaluated in terms of MAE, MSE, and CC. Results are averaged over missing rates ranging from 10% to 90%. Alternator++ generally outperforms the baselines in terms of MSE and CC. However, for MAE, it faces challenges on the Covid dataset, where Alternator and Mamba perform better.

2022)), and Dyffusion (Cachay et al., 2023), evaluating with CRPS (Matheson & Winkler, 1976) and MSE, where CRPS is a proper scoring rule for probabilistic forecasting (Gneiting & Katzfuss, 2014; de Bézenac et al., 2020), and MSE is measured on the mean prediction from a 50-member ensemble.

Table 2 shows that Alternator++ achieves good performance on both metrics, improving CRPS by approximately 20% over MCVD and reducing MSE by a similar margin compared to Dyffusion. While Alternator++ has slightly longer inference time than some baselines, it remains more than $50\times$ faster than MCVD and more than $3\times$ faster than Dyffusion for multi-lead time forecasts.

5 Related Work

The landscape of generative time-series modeling is rich with sophisticated approaches addressing the complex challenges posed by time-dependent data. Our work, Alternator++, builds upon and meaningfully distinguishes itself from several key paradigms that we discuss next.

Neural ordinary differential equations (N-ODEs), as explored by (Chen et al., 2018; Rubanova et al., 2019), provide differential equation solvers based on neural networks. However, their fundamentally deterministic nature is at odds with the stochasticity characterizing real-world time series data. Neural stochastic differential equations (Neural SDEs), introduced by Liu et al. (2019), address this by incorporating stochastic terms to model randomness. However, Neural SDEs typically

Table 2: Performance on sea surface temperature forecasting with forecasting horizons ranging from 1 to 7 days ahead. Metrics are averaged over the entire evaluation horizon, with standard errors reported. For both CRPS and MSE, lower values indicate better performance. The time column indicates the total duration required to forecast all 7 future timesteps for a single batch. Alternator++ outperforms the baselines in terms of MSE and is relatively fast compared to MSDV and Dyffusion. However, it exhibits high standard errors and may underperform Mamba and Dyffusion in terms of CRPS.

Method	CRPS	MSE	Time [s]
Perturbation	0.281 ± 0.004	0.180 ± 0.011	0.4241
Dropout	0.267 ± 0.003	0.164 ± 0.004	0.4241
DDPM	0.246 ± 0.005	0.177 ± 0.005	0.3054
MCVD	0.216	0.161	79.167
Dyffusion	0.224 ± 0.001	0.173 ± 0.001	4.6722
Mamba	0.219 ± 0.002	0.134 ± 0.003	0.6452
Alternator	0.221 ± 0.031	0.144 ± 0.045	0.7524
Alternator++	$\mathbf{0.212 \pm 0.040}$	$\mathbf{0.116 \pm 0.035}$	1.4277

rely on computationally expensive numerical solvers and maintain high-dimensional state representations. In contrast, Alternator++ maintains stochasticity through noise models for both latent and observation spaces that lean on more computationally efficient methods (score matching) compared to ODE/SDE solvers, while preserving expressiveness for complex temporal patterns.

Variational recurrent neural networks (VRNNs) marry RNNs with latent variables for sequential data modeling (Fabius & Van Amersfoort, 2014; Fortunato et al., 2017; Krishnan et al., 2015). Fitting these models is often done using variational inference (VI). Different works have explored different ways of representing the variational distribution of the latent variables, with the richer variational distributions leveraging both past and future sequence elements for a given time step using bidirectional RNNs (Bayer & Osendorfer, 2014; Fraccaro et al., 2016; Martinez et al., 2017; Doerr et al., 2018; Karl et al., 2016; Castrejon et al., 2019). However, they all face the challenge that at test time, the future isn't available, and sampling highly plausible sequences becomes difficult because of this. Alternator++ also relies on low-dimensional latent variables. However, instead of using VI for training, it uses the Alternator loss, which is a cross-entropy objective function on the observed and latent trajectories (Rezaei & Dieng, 2024).

State-space models have proven effective across domains (Gu & Dao, 2023; Rezaei et al., 2022; 2021; Rangapuram et al., 2018). Mamba (Gu & Dao, 2023) introduced selective SSMs, with subsequent domain-specific adaptations including Vision Mamba (Zhu et al., 2024), MambaStock (Shi, 2024), and protein models (Xu et al., 2024). Despite its versatility, Mamba uses a high-dimensional hidden state space ($\mathbf{h}_t \in \mathbb{R}^d$), making it computationally expensive, especially for long-horizon modeling. In addition to Mamba, other recent advances have significantly pushed the boundaries of state-space modeling by introducing long convolutions as an alternative to recurrence (Smith et al., 2023), enabling subquadratic context length processing while maintaining competitive performance with transformers (Gu et al., 2022). Alternator++ differs from these approaches by maintaining a low-dimensional latent state $\mathbf{z}_t \in \mathbb{R}^{d_z}$ where $d_z \ll d$ and employing trainable noise models, thus reducing complexity and improving generalization.

Diffusion models. Alternator++ shares conceptual similarities with diffusion-based models like TimeGrad (Rasul et al., 2021), Dyffusion (Cachay et al., 2023), and others (Karras et al., 2022; Dhariwal & Nichol, 2021; Voleti et al., 2022; Pathak et al., 2022; Li et al., 2024). TimeGrad introduced diffusion for probabilistic forecasting, requiring hundreds of iterations to reconstruct signals. ScoreGrad (Yan et al., 2021) advanced this with continuous-time score-based frameworks, while Dyffusion (Cachay et al., 2023) incorporated physics-informed priors. Recent advances include CSDI (Tashiro et al., 2021), DiffWave (Kong et al., 2020), TimeDiff (Shen & Kwok, 2023), DiffuSeq (Gong et al., 2022), TDPM (Ye et al., 2024), and ANT (Lee et al., 2024). Alternator++ differs in two main ways: (1) it uses noise-conditioned transitions, enabling direct next-step esti-

mation without iterative perturbations, and (2) it models state transitions in a non-Markovian way, incorporating richer temporal relationships via learned noise components.

A key distinction emerges in how noise is treated across these paradigms. While diffusion models start from fixed noise distributions, they operate through iterative denoising over hundreds of steps, allowing them to progressively learn complex noise structures through this multi-step process. Alternators, by contrast, generate sequences in a single forward pass, making it crucial to capture appropriate noise characteristics at each step directly. The learned noise terms $\boldsymbol{\epsilon}_{\boldsymbol{\psi}}^{\boldsymbol{t}}$ and $\boldsymbol{\epsilon}_{\boldsymbol{\nu}}^{\boldsymbol{t}}$ are not simply adding extra noise for regularization, but rather learning the appropriate mean of the noise distribution that should be applied at each time step given the current state. Empirically, if these terms were merely acting as normalization, we would expect minimal performance differences from the original Alternator. However, our results show substantial improvements compared to ScoreGrad, which is a diffusion-based model, indicating that the learned noise models capture meaningful signal-dependent stochastic patterns that fixed noise cannot represent.

Alternators. The original Alternator (Rezaei & Dieng, 2024) employed a two-network architecture alternating between observation processing and latent state evolution. The α-Alternator (Rezaei & Dieng, 2025) dynamically adjusts the dependence on observations and latents when predicting an element of the sequence by using the Vendi Score (Friedman & Dieng, 2023), which makes it robust to varying noise levels in sequence data. Alternator++ is yet another extension of standard Alternators. It shifts from implicit to explicit noise modeling, using neural networks to model the means of the noise terms for both the observation and latent trajectories. Prior work has explored learned noise in various contexts. Score matching (Song & Ermon, 2019) learns the score function $\nabla_x \log p(x)$, which relates to noise prediction through Tweedie's formula. Variational diffusion models (Kingma et al., 2021) learn the noise schedule jointly with the model. Our approach differs by learning conditional noise predictors that adapt based on the latent state, providing a middle ground between fixed noise (Alternator) and fully iterative noise modeling (diffusion models).

6 Conclusion

We developed *Alternator++*, a new Alternator model that uses noise models from the diffusion modeling literature for improved performance. The noise models are neural networks whose parameters are learned by adding a noise-matching objective to the Alternator loss. By modeling the noise terms in both the latent and observed trajectories, Alternator++ captures complex temporal dynamics more accurately. We demonstrate this in experiments on density estimation, imputation, and forecasting tasks, where we found Alternator++ outperforms strong baselines such as Mamba, ScoreGrad, and Dyffusion. In addition to its generalization capabilities, Alternator++ offers fast sampling and low-dimensional latent variables, two features that diffusion models and state-space models lack. In combining low-dimensional latent representations with trainable noise models, Alternator++ enables both accurate modeling and computational efficiency.

Limitations Despite the promising results shown in this paper, Alternator++ can be extended to enhance performance even further. Indeed, the schedule parameters β_t and α_t of Alternator++ need to be tuned for each application and each dataset, making them domain-specific. This can be time-consuming and may limit the application of Alternator++ to a narrower set of domains. Future work can consider adaptive noise scheduling techniques that dynamically adjust to varying noise levels within sequences, potentially improving performance on temporally heterogeneous data.

References

Izhar Bar-Gad, Ya'acov Ritov, Eilon Vaadia, and Hagai Bergman. Noise in neurons is message dependent. *Proceedings of the National Academy of Sciences*, 99(18):12186–12191, 2002.

Justin Bayer and Christian Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2014.

Tim Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of econometrics*, 31(3):307–327, 1986.

Salva Rühling Cachay, Bo Zhao, Hailey James, and Rose Yu. Dyffusion: A dynamics-informed diffusion model for spatiotemporal forecasting. *arXiv preprint arXiv:2306.01984*, 2023.

Lluis Castrejon, Nicolas Ballas, and Aaron Courville. Improved conditional vrnns for video prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7608–7617, 2019.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

RT Chen, D Duvenaud, and Y Rubanova. Latent odes for irregularly-sampled time series. *Advances in Neural Information Processing Systems*, 32:3, 2019.

Ting Chen. On the importance of noise scheduling for diffusion models. *arXiv preprint arXiv:2301.10972*, 2023.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. *Advances in neural information processing systems*, 28, 2015.

Emmanuel de Bézenac, Syama Sundar Rangapuram, Konstantinos Benidis, Michael Bohlke-Schneider, Richard Kurle, Lorenzo Stella, Hilaf Hasson, Patrick Gallinari, and Tim Januschowski. Normalizing kalman filters for multivariate time series analysis. *Advances in Neural Information Processing Systems*, 33:2995–3007, 2020.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

Andreas Doerr, Christian Daniel, Martin Schiegg, Nguyen-Tuong Duy, Stefan Schaal, Marc Toussaint, and Trimpe Sebastian. Probabilistic recurrent state-space models. In *International conference on machine learning*, pp. 1280–1289. PMLR, 2018.

Ensheng Dong, Hongru Du, and Lauren Gardner. An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, 20(5):533–534, 2020.

Otto Fabius and Joost R Van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

Marco Fraccaro, Søren Kaae Sønderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. *Advances in neural information processing systems*, 29, 2016.

Dan Friedman and Adji Bousso Dieng. The Vendi Score: A Diversity Evaluation Metric for Machine Learning. *Transactions on Machine Learning Research*, 2023.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1:125–151, 2014.

Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob J Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. *arXiv preprint arXiv:2105.06643*, 2021.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022. URL https://arxiv.org/abs/2111.00396.

Masoud Haghbin, Ahmad Sharafati, Davide Motta, Nadhir Al-Ansari, and Mohamadreza Hosseinian Moghadam Noghani. Applications of soft computing models for predicting sea surface temperature: a comprehensive review and assessment. *Progress in earth and planetary science*, 8:1–19, 2021.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P Kingma, Ben Poole, Mohammad Norouzi, David J Fleet, et al. Imagen video: High definition video generation with diffusion models. *arXiv preprint arXiv:2210.02303*, 2022.

Maximilian Karl, Maximilian Soelch, Justin Bayer, and Patrick Van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. corr abs/1703.07015 (2017). *arXiv preprint arXiv:1703.07015*, 2017.

Seunghan Lee, Kibok Lee, and Taeyoung Park. Ant: Adaptive noise schedule for time series diffusion models. *arXiv preprint arXiv:2410.14488*, 2024.

Anjian Li, Zihan Ding, Adji Bousso Dieng, and Ryne Beeson. Constraint-aware diffusion models for trajectory optimization. *arXiv preprint arXiv:2406.00990*, 2024.

Shanchuan Lin, Bingchen Liu, Jiashi Li, and Xiao Yang. Common diffusion noise schedules and sample steps are flawed. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 5404–5411, 2024.

Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. Neural sde: Stabilizing neural ode networks with stochastic noise, 2019. URL https://arxiv.org/abs/1906.02355.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *arXiv preprint arXiv:2206.00927*, 2022.

Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2891–2900, 2017.

James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 22(10):1087–1096, 1976.

Michael W McCracken and Serena Ng. Fred-md: A monthly database for macroeconomic research. *Journal of Business & Economic Statistics*, 34(4):574–589, 2016.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. *Advances in neural information processing systems*, 31, 2018.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International Conference on Machine Learning*, pp. 8857–8868. PMLR, 2021.

Mohammad R Rezaei, Alex E Hadjinicolaou, Sydney S Cash, Uri T Eden, and Ali Yousefi. Direct discriminative decoder models for analysis of high-dimensional dynamical neural data. *Neural Computation*, 34(5):1100–1135, 2022.

Mohammad Reza Rezaei and Adji Bousso Dieng. Alternators for sequence modeling. *arXiv preprint arXiv:2405.11848*, 2024.

Mohammad Reza Rezaei and Adji Bousso Dieng. The $alpha$-alternator: Dynamic adaptation to varying noise levels in sequences using the vendi score for improved robustness and performance. *arXiv preprint arXiv:2502.04593*, 2025.

Mohammad Reza Rezaei, Kensuke Arai, Loren M Frank, Uri T Eden, and Ali Yousefi. Real-time point process filter for multidimensional decoding problems using mixture models. *Journal of neuroscience methods*, 348:109006, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *International Conference on Machine Learning*, pp. 31016–31029. PMLR, 2023.

Zhuangwei Shi. Mambastock: Selective state space model for stock prediction. *arXiv preprint arXiv:2402.18959*, 2024.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling, 2023. URL https://arxiv.org/abs/2208.04933.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Souhaib Ben Taieb, Gianluca Bontempi, Amir F Atiya, and Antti Sorjamaa. A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. *Expert systems with applications*, 39(8):7067–7083, 2012.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in Neural Information Processing Systems*, 34:24804–24816, 2021.

Kevin E Trenberth, Philip D Jones, Peter Ambenje, Roxana Bojariu, David Easterling, Albert Klein Tank, David Parker, Fatemeh Rahimzadeh, James A Renwick, Matilde Rusticucci, et al. Observations: surface and atmospheric climate change. *Climate change 2007: the physical science basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*, pp. 235–336, 2007.

Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. *Advances in Neural Information Processing Systems*, 34:11287–11302, 2021.

Vikram Voleti, Alexia Jolicoeur-Martineau, and Chris Pal. Mcvd-masked conditional video diffusion for prediction, generation, and interpolation. *Advances in Neural Information Processing Systems*, 35:23371–23385, 2022.

Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is mamba effective for time series forecasting? *Neurocomputing*, 619:129178, 2025.

Bohao Xu, Yingzhou Lu, Yoshitaka Inoue, Namkyeong Lee, Tianfan Fu, and Jintai Chen. Protein-mamba: Biological mamba models for protein function prediction. *arXiv preprint arXiv:2409.14617*, 2024.

Tijin Yan, Hongwei Zhang, Tong Zhou, Yufeng Zhan, and Yuanqing Xia. Scoregrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models. *arXiv preprint arXiv:2106.10121*, 2021.

Zilyu Ye, Zhiyang Chen, Tiancheng Li, Zemin Huang, Weijian Luo, and Guo-Jun Qi. Schedule on the fly: Diffusion time prediction for faster and better image generation. *arXiv preprint arXiv:2412.01243*, 2024.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

A Appendix

B Datasets Details

To test the Alternator++ on real datasets, We use the Monash time series repository Godahewa et al. (2021), which contains a group of 30 diverse real-world time-series datasets. From this repository, we specifically select the Solar Weekly, COVID death, FRED-MD, and NN5 Daily datasets as our focus of analysis and experimentation. These datasets have been chosen due to they reflect a variety of dynamics and challenges that allow us to thoroughly assess the capabilities of our model. We used the same setting for Alternator++ here as we used for the spiral dataset.

Solar. The Solar dataset represents the temporal aspects of solar power production within the United States during the year 2006. This specific sub-collection is dedicated to the state of Alabama and encompasses 137 individual time series, each delineating the weekly solar power production for a discrete region within the state during the aforementioned yearLai et al. (2017). The temporal sequences encapsulated within this dataset effectively capture nuanced patterns, reflecting both seasonal fluctuations and geographical disparities in solar power generation across the United States. Therefore, the Solar dataset serves as a valuable resource for the evaluation and validation of generative models within the domain of time series analysis, with a specific emphasis on seasonal data dynamics.

Covid. The Covid dataset time series represents the fatalities for various countries and regions worldwide and was sourced from the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE). This dataset encompasses 266 daily time series, delineating the trajectory of COVID-19 fatalities across 43 distinct regions, comprising both states and countries. These temporal sequences within the dataset show trends and patterns in fatality rates across diverse regions. Consequently, the COVID-19 dataset assumes a pivotal role as a valuable resource for the examination and validation of generative models, particularly within the realm of time series analysis, with a specific emphasis on dynamic trends Dong et al. (2020).

Fred. The Federal Reserve Economic Data (FRED) dataset represents an extensive and dynamic repository encompassing a diverse range of macroeconomic indicators, meticulously curated from the FRED-MD databaseMcCracken & Ng (2016). This dataset is intentionally structured to facilitate recurring monthly updates, encapsulating 107 distinct time series spanning a duration of roughly 12 years (equivalent to 146 months). These time series eloquently depict various macroeconomic metrics, primarily procured from the Federal Reserve Bank.

NN5. The NN5 dataset shows daily ATM cash withdrawals in cities across the United KingdomTaieb et al. (2012). This dataset became well-known as a central part of the NN5 International Forecasting Competition, providing a deep look into the complex dynamics of ATM cash transactions in the banking field. There are 111 individual time series in this dataset, each covering around two years of daily cash withdrawal data, totaling 735 data points for each series. Its complexity comes from various time patterns, such as different seasonal cycles, local trends, and significant changes in how people withdraw cash over time. These features make it a valuable resource for testing and assessing generative models in the area of time series analysis.

C Implementation and Hyperparameter Search

We conducted comprehensive hyperparameter optimization and implementation refinements for Alternator++, carefully customized for each experimental task—density estimation, imputation, and forecasting—to maximize performance across all datasets. This section provides detailed insights into our implementation strategies, hyperparameter optimization approaches, and presents the results in the accompanying tables.

Density Estimation. For density estimation tasks, we configured Alternator++ with a latent dimensionality (D_z) of 32, identified through exhaustive grid search across values of 16, 32, and 64. The architecture incorporates four specialized networks—$f_\theta^x(.)$, $f_\phi^z(.)$, $\boldsymbol{\epsilon}_z^{(b)}$, and $\boldsymbol{\epsilon}_x^{(b)}$—each constructed with two layers of self-attention mechanisms. Training utilized the Adam optimizer beginning with a learning rate of 1×10^{-3}, which gradually decreased to 1×10^{-5} over 1000 epochs following a cosine annealing schedule. We processed data in batches of 100 samples. The noise variance parameters were determined through rigorous hyperparameter exploration, evaluating σ_x and σ_z across a range of values including 0.05, 0.1, 0.15, 0.2, and 0.3. Our experiments revealed optimal performance with $\sigma_x = 0.3$ and $\sigma_z = 0.15$ consistently across datasets. We employed a fixed, linearly spaced noise schedule throughout all density estimation experiments to ensure methodological consistency.

Table 3: Hyperparameters for Density Estimation Experiments

Hyperparameter	**Value**
Latent Dimension (D_z)	32
Learning Rate	1×10^{-3} to 1×10^{-5} (cosine annealing)
Batch Size	100
Noise Variances (σ_x, σ_z)	0.3, 0.15
Epochs	1000
Noise Schedule	Fixed (linearly spaced)

Time-Series Imputation. The imputation experiments leveraged the architectural foundation established in our density estimation setup, with modifications tailored to the unique challenges of handling missing data. Our hyperparameter optimization strategy prioritized developing robust performance across varying levels of data missingness, with particular emphasis on scenarios with high proportions of missing values. We trained the model using the Adam optimizer with an initial learning rate of 5×10^{-4}, gradually reducing to 5×10^{-6} over 800 epochs through cosine annealing. To accommodate the increased variability inherent in imputation tasks, we employed a reduced batch size of 32 samples. Through systematic experimentation, we determined that noise variances of $\sigma_x = 0.15$ and $\sigma_z = 0.15$ provided optimal performance. Missing values were systematically introduced using a Missing At Random (MAR) protocol to simulate realistic data scenarios. Across all

imputation tasks, we maintained a consistent approach with a fixed, linearly spaced noise schedule to ensure experimental rigor and comparability.

Table 4: Hyperparameters for Time-Series Imputation Experiments

Hyperparameter	**Value**
Latent Dimension (D_z)	64
Learning Rate	5×10^{-4} to 5×10^{-6} (cosine annealing)
Batch Size	32
Noise Variances (σ_x, σ_z)	0.15, 0.15
Epochs	800
Missing Data Rate	10% to 90% (MAR)
Noise Schedule	Fixed (linearly spaced)

Sea Surface Temperature Forecasting. For our Sea Surface Temperature (SST) forecasting task, we train two Adversarial Diffusion Models (ADM) (Dhariwal & Nichol, 2021): one for the OTN and FTN components, and another for the scoring functions. Each model employs a U-Net backbone with attention modules placed after each CNN block. The backbone is configured with 128 base channels, 2 ResNet blocks per resolution, and a hierarchical channel multiplier structure of $\{1, 2, 4\}$ to capture complex spatial-temporal dynamics across multiple resolutions.

The models are trained for 700K iterations using a batch size of 8. We use the AdamW optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and an initial learning rate of 1×10^{-4}. For the diffusion process, noise schedules are calibrated with $\sigma_z = 0.1$, $\sigma_x = 0.2$, and $\alpha_t = 0.5$ for all time steps. These settings balance stochastic exploration with deterministic prediction, allowing the model to capture both short-term patterns and long-range uncertainties inherent in climate dynamics.

Table 5: Model configuration for SST forecasting.

Number of ResNet blocks	2
Base channels	128
Channel multipliers	1, 2, 4
Attention resolutions	16
Label dimensions	10
Parameters (M)	55.39

Table 6: Training hyperparameters for SST forecasting.

Learning rate	1×10^{-4}
AdamW (β_1, β_2)	(0.9, 0.999)
Batch size	8
Number of iterations	700K
GPU	NVIDIA A100

All SST experiments were conducted on NVIDIA A6000 GPUs with 48GB of memory, enabling efficient processing of the high-dimensional spatial-temporal inputs essential for accurate SST forecasting.