Universal Time Series Generation with Neural Controlled Differential Equations

Torben Berndt[1,*] **Elyes Farjallah**[1,2,*] **Leif Seute**[1,3] **Raeid Saqur**[4,5]

Benjamin Walker[4,6,†] **Jan Stühmer**[1,2,†]

[1]Heidelberg Institute for Theoretical Studies, Heidelberg, Germany
[2]IAR, Karlsruhe Institute of Technology, Karlsruhe, Germany
[3]Max Planck Institute for Polymer Research, Mainz, Germany
[4]Mathematical Institute, University of Oxford, Oxford, UK
[5]Vector Institute, Toronto, Canada
[6]Continuous Intelligence Ltd, London, UK

[*]Equal first authorship. [†]Equal senior authorship.

Abstract

Recent work on the sequence universality of State Space Models (SSMs) has introduced efficient, maximally expressive continuous-time approaches for time-series modelling. While these works focus on discriminative settings, we extend this perspective to generative time-series modelling by proving that maximally expressive Structured Linear Controlled Differential Equations (SLiCEs) are universal time-series generators, in the sense that they can approximate the induced path laws of continuous causal pushforwards on compact latent sets in W_∞. Building on these theoretical results, we propose *Generative SLiCEs (G-SLiCEs)*, a maximally expressive continuous-time model for flow matching on path-space. Empirically, we show that expressivity improves performance in probabilistic forecasting and downstream tasks, while retaining the advantages of continuous-time models such as generalising to arbitrary observation grids. This is particularly beneficial for irregular grids, where fixed-grid models often struggle. [1]

1 Introduction

In probabilistic forecasting, capturing the full conditional distribution over future events is crucial, for example, when predicting extreme weather phenomena [51], electricity demand [61], or traffic [52]. Probabilistic forecasting is therefore inherently generative, and recent work has increasingly applied generative machine learning methods, such as diffusion models and flow matching [59, 42, 63], to this task [56, 62, 3, 11, 37, 38]. However, these models typically rely on sequence-model backbones that are not maximally expressive, which can limit their performance.

A parallel line of work develops expressive, parallel-in-time state space models (SSMs) by imposing structures on transition matrices, including block diagonal [22] and diagonal-plus-low-rank constructions [69, 67, 58]. Recent work unifies these approaches in a continuous-time framework building on Linear Controlled Differential Equations and characterise their universality [15, 65].

We extend this theory to generative settings, where distributions on path space are modelled as pushforwards of latent path laws. Building on this perspective, we generalise current flow-based approaches for probabilistic forecasting from sequence-to-sequence mappings to path-to-path mappings.

[1]The source code is available at: `https://github.com/hits-mli/gslices`

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To address both the expressivity gap and the discrete-time formulation of existing state-of-the-art models, we propose Generative Structured Linear Controlled Differential Equations (G-SLiCEs): a maximally expressive continuous-time model for generative time-series modelling.

Main contributions We introduce a new generative approach for time-series data – Generative Structured Linear Controlled Differential Equations (G-SLiCEs) – that achieves state-of-the-art performance and is robust under grid shifts. This relies on two key insights: **(1)** we formulate flow matching on path space using maximally expressive Structured Linear CDE backbones, and **(2)** we prove that pathwise expressivity implies universality for continuous causal pushforwards of compactly supported latent path laws, linking discriminative expressivity of causal path-to-path models to generative capabilities (Theorem 2). We demonstrate that G-SLiCEs outperform current state-of-the-art models on a comprehensive probabilistic forecasting benchmark, which we attribute to their universality, and show that the continuous-time formulation improves robustness to sampling-grid shifts Further, we provide a concrete example showing that transition structure has provable expressivity consequences in the generative setting.

1.1 Related work

Flow matching Flow matching [42, 1] has been proposed as powerful generative modelling technique on continuous, finite-dimensional Euclidean spaces and has since been generalised to Riemannian manifolds [13], discrete data [28] and functionals [36]. These approaches do not, to our knowledge, yield distributional universality results on path space.

Generative models for probabilistic time-series forecasting. Classical neural baselines either parameterise the predictive distribution explicitly (DeepAR [57], TFT [40], autoregressive flows over RNN backbones [56]) or are deterministic and combined with quantile heads (WaveNet [46], PatchTST [45], DLinear [70]). Generative approaches replace this distributional head with a learned density model. Variational auto-encoders (VAEs) such as TimeVAE [18] were explored first, followed by diffusion models, including CSDI [62], SSSD [2], TSDiff [37], and the stochastic-process formulation of Biloš et al. [11]. Most recently, TSFlow [38] advances this line of work by employing flow matching.

Expressivity of state space models. Recent work aims to make parallel-in-time sequence models more expressive while preserving efficiency. One direction parallelises non-linear RNNs by casting recurrence as a fixed-point problem solved with Newton-type methods [41, 32]. Another develops input-dependent linear RNNs and state-space models, including input-dependent block-diagonal LRNNs [22], DeltaNet variants [69, 67, 58], Mamba [33], Mamba-2 [17], RWKV-7 [49], HGRN-2 [54], mLSTM [7], Gated Linear Attention [68], Gated Random Feature Attention [50], Gated Slot Attention [71], TTT-Linear [60], and Titans [8]. These models usually use diagonal or diagonal-plus-low-rank transition matrices.

2 Universality in the context of time series generation

We begin with the path-to-path model classes used in this paper. We then define the distributional notion of expressivity used for time series generation and show how it applies to G-SLiCEs.

2.1 Linear NCDEs and SLiCEs

Fix a time interval $[t_0, t_n]$. For $d \in \mathbb{N}$, let $\mathcal{X}(d) = C^{1,0}([t_0, t_n], \mathbb{R}^d)$ denote the space of absolutely continuous, time-augmented input paths used by the model, all starting from the same point, equipped with the 1-variation topology. Let $\mathcal{Y}(d) = C([t_0, t_n], \mathbb{R}^d)$ denote the output path space, equipped with the supremum metric $\rho_\infty(Y, \widetilde{Y}) = \sup_{t\in[t_0,t_n]} \|Y_t - \widetilde{Y}_t\|_2$. A map $T : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$ is causal if $X|_{[t_0,t]} = \widetilde{X}|_{[t_0,t]}$ implies $T(X)_t = T(\widetilde{X})_t$ for every $t \in [t_0, t_n]$.

Let $X \in \mathcal{X}(d_X)$ denote an input path and let $\omega^X \in C^{1,0}([t_0, t_n], \mathbb{R}^{d_\omega})$ denote a deterministic causal augmentation of X. In applications, ω^X may include time, observed values, masks, lags, and other deterministic features. Throughout this subsection, integrals are understood in the Riemann–Stieltjes sense, which is well defined for the absolutely continuous controls used by the model.

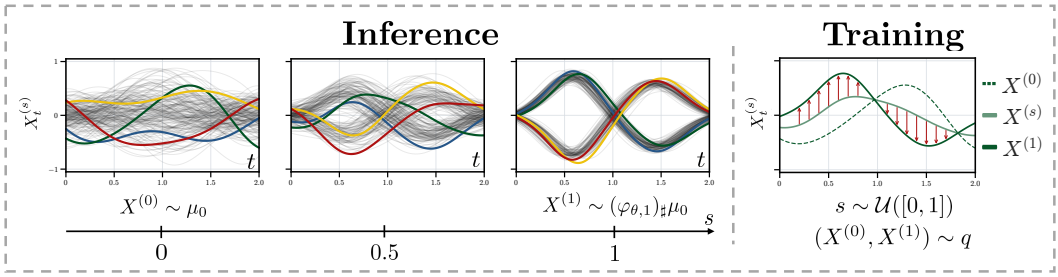


Figure 1: **Path-space flow matching with G-SLiCE.** G-SLiCE models probabilistic time-series generation as a continuous flow on path space. At inference time, an initial path $X^{(0)} \sim \mu_0$ is transported by the learned flow $\varphi_{\theta,s}$, producing a terminal sample $X^{(1)} \sim (\varphi_{\theta,1})_{\#}\mu_0$. Each inference panel shows samples from the evolving distribution on path space. During training, the SLiCE vector field is trained to match the path-space displacement at the interpolated path $X^{(s)}$, which is constructed from prior and data paths sampled from the joint distribution q.

Definition 2.1 (NCDE)**.** An NCDE is a causal path-to-path map $X \mapsto z^\theta(X)$ defined by

$$h_{t_0} = \xi_\varphi(X_{t_0}), \qquad h_t = h_{t_0} + \int_{t_0}^{t} g_\theta(h_s)\, d\omega_s^X, \qquad z_t = r_\psi(h_t), \tag{1}$$

where $h_t \in \mathbb{R}^{d_h}$, $z_t \in \mathbb{R}^{d_z}$, ξ_φ is a learnable initialisation map, $g_\theta(h) \in \mathbb{R}^{d_h \times d_\omega}$ is a learnable vector field, and r_ψ is a learnable readout.

Definition 2.2 (Linear NCDE)**.** A Linear NCDE is an NCDE whose vector field is linear in the hidden state. Equivalently, there are matrices $A_\theta^1, \dots, A_\theta^{d_\omega} \in \mathbb{R}^{d_h \times d_h}$ such that

$$h_{t_0} = \xi_\varphi(X_{t_0}), \qquad h_t = h_{t_0} + \int_{t_0}^{t} \sum_{i=1}^{d_\omega} A_\theta^i h_s\, d\omega_s^{X,i}, \qquad z_t = r_\psi(h_t). \tag{2}$$

The dynamics are linear in h, but the induced map $X \mapsto z^\theta(X)$ can be highly non-linear, as the hidden state is multiplied by increments of the driving path. Linear NCDEs also benefit from an efficient parallel-in-time computation. On piecewise linear controls,

$$h_{t_{j+1}} = \Phi_j^\theta(X) h_{t_j}, \qquad \Phi_j^\theta(X) = \exp\left(\sum_{i=1}^{d_\omega} A_\theta^i (\omega_{t_{j+1}}^{X,i} - \omega_{t_j}^{X,i})\right). \tag{3}$$

Therefore $h_{t_k} = \Phi_{k-1}^\theta(X) \cdots \Phi_0^\theta(X) h_{t_0}$, and after computing the exact interval transition operators $\Phi_j^\theta(X)$ in parallel, evaluating the trajectory reduces to composing matrices. Since matrix multiplication is associative, the sequence of prefix products can be evaluated by a parallel scan.

Structured Linear CDEs, or SLiCEs, balance expressivity and efficiency by restricting each A_θ^i to a prescribed structured matrix family. Dense Linear NCDEs sit at the expressive but computationally expensive end of this spectrum, while diagonal transition matrices give efficient models that are not maximally expressive. Other structures, such as block-diagonal, diagonal-plus-low-rank, sparse, and Walsh–Hadamard families, retain maximal expressivity while reducing recurrent cost and parameter count. If the structure is also closed under matrix multiplication, as in fixed block-diagonal matrices, then the transition products remain structured, improving the efficiency of parallel-in-time evaluation.

Theorem 1 (Path-to-path universality of maximally expressive SLiCEs)**.** *Let $\mathcal{K} \subset \mathcal{X}(d_X)$ be compact, $T : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$ be continuous and causal, and $\omega_s^X = X_s$. Any SLiCE class that is path-to-point maximally expressive in the sense of Walker et al. [65] with a linear readout satisfies the following property. For every $\varepsilon > 0$, there exists a model in the class, with hidden dimension $d_h \in \mathbb{N}$ and feed-forward neural network readout r_ψ, such that*

$$\sup_{X \in \mathcal{K}} \rho_\infty\big(z^\theta(X), T(X)\big) \le \varepsilon. \tag{4}$$

The proof is given in Appendix B. Section 2.2 lifts this deterministic approximation result to a distributional statement for time series generation.

2.2 Universal time series generation

We now define the distributional notion of expressivity used in this paper. If $X \sim \mu$ and $F_\theta : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$ is Borel measurable, then $(F_\theta)_\#\mu$ denotes the law of $F_\theta(X)$. For probability measures $\nu, \widetilde{\nu}$ on $\mathcal{Y}(d_y)$, the ∞-Wasserstein distance with respect to ρ_∞ is

$$W_\infty(\nu, \widetilde{\nu}) = \inf_{\pi \in \Pi(\nu, \widetilde{\nu})} \operatorname*{ess\,sup}_{(Y, \widetilde{Y}) \sim \pi} \rho_\infty(Y, \widetilde{Y}), \tag{5}$$

where $\Pi(\nu, \widetilde{\nu})$ is the set of couplings of ν and $\widetilde{\nu}$.

Definition 2.3 (Universal causal time series generator)**.** Let

$$\mathcal{F} = \{F_\theta : \mathcal{X}(d_X) \to \mathcal{Y}(d_y) \mid \theta \in \Theta\} \tag{6}$$

be a class of Borel measurable causal maps. We say that $\mathcal{F}$ is a universal causal time series generator if, for every compact set $\mathcal{K} \subset \mathcal{X}(d_X)$, every Borel probability measure μ satisfying $\mu(\mathcal{K}) = 1$, every continuous causal map $T : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$, and every $\varepsilon > 0$, there exists $\theta \in \Theta$ such that

$$W_\infty\big((F_\theta)_\#\mu, T_\#\mu\big) \leq \varepsilon, \tag{7}$$

where W_∞ is computed on $\mathcal{Y}(d_y)$ with respect to ρ_∞.

Theorem 2 (Pathwise approximation implies generative approximation)**.** *Let*

$$\mathcal{F} = \{F_\theta : \mathcal{X}(d_X) \to \mathcal{Y}(d_y) \mid \theta \in \Theta\} \tag{8}$$

be a class of Borel measurable causal maps. Suppose that, for every compact set $\mathcal{K} \subset \mathcal{X}(d_X)$, every continuous causal map $T : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$, and every $\varepsilon > 0$, there exists $\theta \in \Theta$ such that

$$\sup_{X \in \mathcal{K}} \rho_\infty\big(F_\theta(X), T(X)\big) \leq \varepsilon. \tag{9}$$

Then $\mathcal{F}$ is a universal causal time series generator.

The proof is given in Appendix A. It uses the coupling obtained by evaluating F_θ and T on the same input path $X \sim \mu$.

Corollary 2.1 (Maximally expressive SLiCEs are universal causal time series generators)**.** *Any SLiCE class satisfying Theorem 1 is a universal causal time series generator.*

Corollary 2.1 applies to the block-diagonal, diagonal-plus-low-rank, sparse, and Walsh–Hadamard SLiCEs [65, 44]. However, not every efficient state-space structure has this expressivity. As shown by Cirone et al. [15], S4 corresponds to a non-selective Linear NCDE whose transition is driven only by the time channel $\omega_t = t$, while Mamba corresponds to a selective SLiCE whose driving path depends on X but whose transition matrices are diagonal. These restrictions enable more efficient parallel-in-time evaluation, but they also limit expressivity. Indeed, single-layer diagonal SLiCEs, S4, and Mamba with linear readouts are not universal even for terminal-time path-to-point functions [15]. Universality results are available for S4D-style recurrences with nonlinear projections and for stacked SSMs with layer-wise nonlinearities [47, 66], but these are discrete sequence-to-sequence approximation theorems rather than the continuous causal path-to-path universality considered here. The next section gives a simple state-tracking example that makes the distinction between these transition structures explicit.

2.3 Example: expressivity gap

Theorem 2 establishes that maximal expressivity in the path-to-path sense implies universality for modelling push-forward distributions, and Corollary 2.1 instantiates this for SLiCEs. This does not mean non-universal models, such as Mamba or S4(D), are also not universal on push-forward distributions. We therefore give a minimal example that isolates the importance of two architectural properties which differentiate SLiCE from these models: selectivity, meaning that the transition can depend on the current input, and state mixing, meaning that different hidden coordinates can interact.

Let $\mathcal{H}_n$ denote the set of binary sequences of length n with no consecutive ones. Given $Z_1, \dots, Z_n \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p)$, define the causal target sequence

$$C_1 = Z_1, \qquad C_k = Z_k(1 - C_{k-1}), \qquad 2 \leq k \leq n. \tag{10}$$

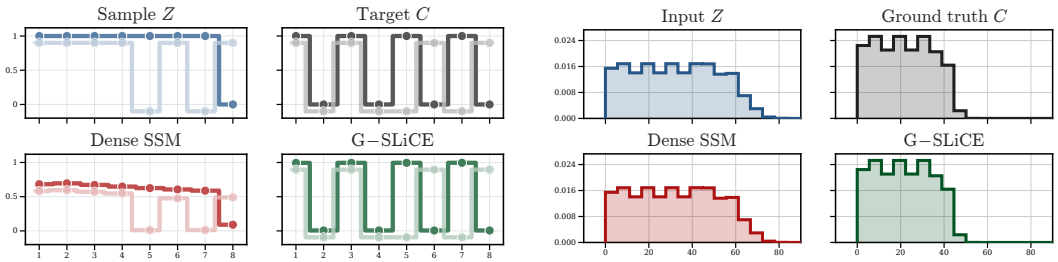


Figure 2: Expressivity gap on the sequence task (Section 2.3). Left: for representative Bernoulli input sequences Z, the target map C removes consecutive ones by the recursion in Equation (10). The dense non-selective SSM fails to learn this state-dependent rule, while G-SLiCE reproduces the target sequence. Right: empirical distributions of cumulative sums over generated sequences. G-SLiCE closely matches the ground-truth pushforward law $\mu_{n,p}$, whereas the dense non-selective SSM produces a visibly distorted distribution.

Thus, a one in the input is retained only when the preceding output was zero, enforcing the constraint $C \in \mathcal{H}_n$ using one step of memory. We write $\mu_{n,p}$ for the law of the resulting output sequences.

We compare zero-order-hold exact-flow discretisations of continuous-time linear state-space models,

$$h_k = \exp(A(Z_k))h_{k-1} + \beta(Z_k), \qquad \widehat{C}_k = w^\top h_k + b, \tag{11}$$

where A and β are affine maps and h_0 is fixed independently of the input sequence. The non-selective case, where A is constant, abstracts S4-style transitions. The diagonal selective case, where $A(Z_k)$ is a function of Z_k but diagonal, abstracts Mamba-style selective diagonal transitions. Dense selective transitions correspond to Linear NCDEs. The following proposition shows that these structural differences result in a strict separation between the induced output laws.

Proposition 3. *Fix $n \geq 1$. For every $p \in [0,1]$ and every $\varepsilon > 0$, there exists a width-2 dense selective exact-flow SSM whose induced output law $\widehat{\mu}_{n,p}$ satisfies*

$$W_\infty\big(\widehat{\mu}_{n,p}, \mu_{n,p}\big) < \varepsilon$$

On the other hand, if $n \geq 2$, no dense non-selective SSM can approximate $\mu_{n,1/2}$ with error strictly less than $1/4$. If $n \geq d+2$ and $p \in \big(1 - \frac{1}{2n}, 1\big]$, no width-$d$ diagonal selective SSM can approximate $\mu_{n,p}$ with error strictly less than $1/2$.

Figure 2 illustrates this expressivity gap empirically. On the left, for two representative input sequences Z (top left), the selective G-SLiCE (bottom right) accurately reproduces the target sequences C (top right), whereas the dense non-selective SSM (bottom left) fails to capture the underlying state-dependent rule. On the right, G-SLiCE (bottom right) closely matches the ground-truth pushforward law $\mu_{n,p}$ (top right) of the input distribution (top left), while the dense non-selective SSM (bottom left) produces a visibly distorted distribution. The proof of Proposition 2.3 is given in Appendix C.

3 Generative Structured Linear CDEs

We now instantiate the expressivity result of Corollary 2.1 in a concrete generative model class, which we call *Generative Structured Linear CDEs (G-SLiCEs)*. A G-SLiCE consists of a causal SLiCE map $G_\theta : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$ and a prior distribution μ. Sampling from the generative model means drawing $X \sim \mu$ and returning $G_\theta(X)$, so the generated law is $\nu_\theta = (G_\theta)_\# \mu$. This is exactly the pushforward setting of Definition 2.3, and hence G-SLiCEs are universal causal time series generators. Thus, G-SLiCEs inherit continuous path-space distributional universality while retaining the parallel-in-time and continuous-time structure of SLiCEs.

Our practical implementation of G-SLiCEs uses the flow matching construction of Kollovieh et al. [38], but generalised from grids to paths. The implementation defines three components, a conditional prior law on path space, a deterministic flow on path space, and a training mechanism. To avoid confusion with physical (path-parametrising) time t, we denote the flow-matching-time s by bracketed

superscripts. Given an initial path $X^{(0)} \sim \mu$, the model defines a path-valued flow

$$\frac{d}{ds}X^{(s)} = F_\theta\Big(s, X^{(s)}\Big), \qquad X^{(0)} \sim \mu, \qquad s \in [0,1], \tag{12}$$

where $F_\theta : [0,1] \times \mathcal{X}(d_X) \to \mathcal{X}(d_X)$ is a causal SLiCE vector-field network. The terminal path $X^{(1)}$ induces the generated law

$$\nu_\theta = (\varphi_{\theta,1})_\# \mu, \tag{13}$$

where $\varphi_{\theta,1}$ denotes the flow map from $s = 0$ to $s = 1$. When the target output space differs from the model input path space, the terminal path is followed by the corresponding readout or projection. Figure 1 visualises this process in its right panel.

The flow time s is injected by concatenating it as an additional constant channel to the current input path. The vector-field network F_θ is parameterised by a SLiCE backbone, so the learned velocity field preserves causality along the physical time axis and uses the same exact-flow and parallel-in-time structure described above.

Prior. Following Kollovieh et al. [38], we use Gaussian processes (GPs) [55] as the noise distribution μ, since they provide a canonical non-parametric model distribution on path space. The construction of the GP depends on the use case. In the unconditional setting, we choose an unfitted Gaussian-process prior $\mu = \mathcal{GP}(m, k)$ on path space. In the conditional setting, each target trajectory $X^{(1)}$ determines conditioning information C, such as a partially observed prefix $C = X^{(1)}\big|_{[t_0,t_c]}$ or a finite set of observations $C = \{(t_i, X^{(1)}_{t_i})\}_{i=0}^m$. Denoting the posterior means and kernels of a GP fitted on C by m_{post} and k_{post}, we use the conditional prior law $\mu(\cdot \mid C) = \mathcal{GP}\big(m_{\text{post}}(\cdot \mid C), k_{\text{post}}(\cdot,\cdot \mid C)\big)$. Conditional generation is performed by sampling $X^{(0)} \sim \mu(\cdot \mid C)$ and returning the terminal path $X^{(1)}$. The resulting conditional model law is $\nu_\theta(\cdot \mid C) = (\varphi^C_{\theta,1})_\# \mu(\cdot \mid C)$, where $\varphi^C_{\theta,1}$ denotes the flow map with context C.

Training. We train the vector-field network using conditional flow matching [42]. In the unconditional case, we independently sample data paths $X^{(1)}$ and noise paths $X^{(0)}$ from the corresponding prior and match them using the mini-batched optimal transport coupling $q(X^{(0)}, X^{(1)})$ [63]. In the conditional case, each data path $X^{(1)} \sim \nu$ determines a context $C = \Gamma(X^{(1)})$, and we sample $X^{(0)} \sim \mu(\cdot \mid C)$. Equivalently,

$$q(X^{(0)}, X^{(1)}) = \nu(X^{(1)})\mu\big(X^{(0)} \mid \Gamma(X^{(1)})\big).$$

For the straight-line interpolant $X^{(s)} = (1-s)X^{(0)} + sX^{(1)}$, the target velocity is $u_s(X^{(0)}, X^{(1)}) = X^{(1)} - X^{(0)}$. The training objective is

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{s\sim\mathcal{U}[0,1] \\ (X^{(0)},X^{(1)})\sim q}} \left[\rho_2\left(F_\theta\big(s, X^{(s)}\big), X^{(1)} - X^{(0)}\right)^2\right].$$

In implementation, the metric is evaluated on the discretised physical-time grid. We visualise the training process in the right panel of Figure 1.

Practical considerations. There are two technical obstacles to applying the universality statement in Section 2.2 to the flow-matching implementation used in our experiments. First, Corollary 2.1 concerns a G-SLiCE generator acting directly from the source to the target distribution. Such a generator can also be realised as an augmented path-space flow by setting

$$\frac{dU^{(s)}}{ds} = 0, \qquad \frac{dY^{(s)}}{ds} = G_\theta(U^{(s)}), \qquad (U^{(0)}, Y^{(0)}) = (X, 0).$$

Then $U^{(s)} = X$ and $Y^{(1)} = G_\theta(X)$, so a linear terminal readout recovers the direct generator. Under the closure assumptions detailed in Appendix D, the augmented flow hypothesis class therefore inherits the direct-map expressivity guarantee.

Second, the Gaussian process input laws used in practice are not compactly supported. However, after discretisation, interpolation, and augmentation, they assign arbitrarily high probability to compact subsets of $\mathcal{X}(d_X)$. For any $\delta > 0$, we may therefore restrict the uniform approximation argument to a compact set carrying probability at least $1 - \delta$. The same coupling then yields an approximation guarantee holding with this probability, rather than a global W_∞ guarantee over the Gaussian tails. Full details are given in Appendix A.

Table 1: Comparison of statistical baselines, neural forecasting models, diffusion-based generative models, and flow-based generative models against G-SLiCE on conditional probabilistic forecasting. We report mean and standard deviation of CRPS over five random seeds on eight GluonTS datasets [5, 6]. Best scores are in **bold**; second-best scores are underlined. The last row indicates whether the selected G-SLiCE configuration uses a dense state-dependent transition block.

Method	Electr.	Exch.	KDD	M4-H	Solar	Traffic	Uber	Wiki
SN	$0.069_{\pm 0.000}$	$0.013_{\pm 0.000}$	$0.561_{\pm 0.000}$	$0.048_{\pm 0.000}$	$0.512_{\pm 0.000}$	$0.221_{\pm 0.000}$	$0.299_{\pm 0.000}$	$0.423_{\pm 0.000}$
ARIMA	$0.344_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.514_{\pm 0.000}$	$0.031_{\pm 0.000}$	$0.558_{\pm 0.003}$	$0.486_{\pm 0.000}$	$0.478_{\pm 0.000}$	$0.654_{\pm 0.000}$
ETS	$0.055_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.584_{\pm 0.000}$	$0.070_{\pm 0.000}$	$0.550_{\pm 0.000}$	$0.492_{\pm 0.000}$	$0.520_{\pm 0.000}$	$0.651_{\pm 0.000}$
DLinear	$0.058_{\pm 0.001}$	$0.015_{\pm 0.004}$	$0.318_{\pm 0.015}$	$0.055_{\pm 0.007}$	$0.794_{\pm 0.027}$	$0.131_{\pm 0.000}$	$0.250_{\pm 0.006}$	$0.259_{\pm 0.002}$
DeepAR	$0.051_{\pm 0.000}$	$0.013_{\pm 0.004}$	$0.362_{\pm 0.017}$	$0.045_{\pm 0.013}$	$0.429_{\pm 0.055}$	$0.103_{\pm 0.002}$	$0.168_{\pm 0.002}$	$0.215_{\pm 0.003}$
TFT	$0.060_{\pm 0.001}$	$\mathbf{0.007}_{\pm \mathbf{0.000}}$	$0.543_{\pm 0.048}$	$0.038_{\pm 0.002}$	$0.371_{\pm 0.006}$	$0.128_{\pm 0.005}$	$0.202_{\pm 0.009}$	$0.219_{\pm 0.004}$
WaveNet	$0.058_{\pm 0.008}$	$0.012_{\pm 0.001}$	$0.305_{\pm 0.018}$	$0.055_{\pm 0.014}$	$0.360_{\pm 0.009}$	$0.099_{\pm 0.002}$	$0.180_{\pm 0.013}$	$\mathbf{0.207}_{\pm \mathbf{0.003}}$
PatchTST	$0.055_{\pm 0.001}$	$0.010_{\pm 0.001}$	$0.420_{\pm 0.011}$	$0.034_{\pm 0.004}$	$0.728_{\pm 0.015}$	$0.151_{\pm 0.007}$	$0.219_{\pm 0.004}$	$0.209_{\pm 0.001}$
CSDI	$0.051_{\pm 0.000}$	$0.013_{\pm 0.001}$	$0.309_{\pm 0.006}$	$0.043_{\pm 0.004}$	$0.360_{\pm 0.006}$	$0.152_{\pm 0.001}$	$0.213_{\pm 0.007}$	$0.318_{\pm 0.012}$
SSSD	$0.048_{\pm 0.001}$	$0.010_{\pm 0.001}$	$\mathbf{0.274}_{\pm \mathbf{0.009}}$	$0.050_{\pm 0.007}$	$0.384_{\pm 0.023}$	$0.097_{\pm 0.002}$	$0.156_{\pm 0.007}$	$0.209_{\pm 0.004}$
Biloš et al.	$0.067_{\pm 0.002}$	$0.012_{\pm 0.004}$	$1.147_{\pm 0.300}$	–	$0.379_{\pm 0.009}$	$0.317_{\pm 0.053}$	$0.450_{\pm 0.086}$	$0.318_{\pm 0.022}$
TSDiff	$0.049_{\pm 0.000}$	$0.011_{\pm 0.001}$	$0.311_{\pm 0.026}$	$0.036_{\pm 0.001}$	$0.358_{\pm 0.020}$	$0.098_{\pm 0.002}$	$0.172_{\pm 0.005}$	$0.221_{\pm 0.001}$
TSFlow	$0.045_{\pm 0.000}$	$0.008_{\pm 0.000}$	$0.288_{\pm 0.004}$	$0.028_{\pm 0.008}$	$0.344_{\pm 0.006}$	$0.082_{\pm 0.000}$	$0.154_{\pm 0.002}$	$\mathbf{0.207}_{\pm \mathbf{0.001}}$ [†]
G-SLiCEs	$\mathbf{0.044}_{\pm \mathbf{0.000}}$	$\mathbf{0.007}_{\pm \mathbf{0.000}}$	$0.275_{\pm 0.009}$	$\mathbf{0.023}_{\pm \mathbf{0.001}}$	$\mathbf{0.342}_{\pm \mathbf{0.012}}$	$\mathbf{0.080}_{\pm \mathbf{0.000}}$	$\mathbf{0.150}_{\pm \mathbf{0.002}}$	$0.218_{\pm 0.002}$
Dense block	✗	✓	✓	✓	✗	✗	✓	✓

[†] We were not able to reproduce the reported TSFlow result on Wiki2000; the best value we obtained was a CRPS of $0.218_{\pm 0.001}$.

4 Experiments

We evaluate G-SLiCE as a generative time-series model in three regimes: First, we study conditional probabilistic forecasting and unconditional generation, in Sections 4.1 and 4.2. Second, we test whether the continuous-time construction improves robustness under changes in the sampling grid in Section 4.3. We consider two such distribution shifts: out-of-distribution sampling frequencies and irregularly observed grids. Experimental details, hyperparameter ranges, and compute resources are given in Appendix G.

Datasets. We use univariate datasets from the GluonTS benchmark suite [5, 6]: Electricity [21], Exchange [39], KDDCup [31], M4-Hourly [43], Solar [39], Traffic [21], UberTLC-Hourly [23], Wikipedia [27], and ETTSmall [72]. The high-frequency ETTSmall dataset is used for the grid-shift experiments; the other datasets are used for probabilistic forecasting and unconditional generation.

Baselines. We compare G-SLiCE against four groups of baselines. The first group contains classical statistical forecasting methods: Seasonal Naive (SN), AutoARIMA, and AutoETS [35]. The second group contains neural forecasting models: DLinear [70], DeepAR [57], Temporal Fusion Transformer (TFT) [40], WaveNet [46], and PatchTST [45]. The third group contains diffusion-based generative models: CSDI [62], SSSD [2], the model of Biloš et al. [11], and TSDiff [37]. The fourth group contains the flow-based model TSFlow [38]. For unconditional generation, we focus on TSFlow as the strongest and conceptually closest flow-based baseline. For the grid-generalisation experiments in Section 4.3, we additionally include the DSPD-OU model of Biloš et al. [11].

4.1 Probabilistic forecasting

Table 1 reports conditional probabilistic forecasting results in terms of *continuous ranked probability score* (CRPS) [29]. G-SLiCE is competitive with the full set of statistical, neural, diffusion-based, and flow-based baselines. It obtains the best score on $6/8$ datasets and outperforms TSFlow on $7/8$ datasets. The last row indicates whether the best G-SLiCE configuration uses a dense state-dependent transition block. The selected structure varies by dataset, suggesting that dense transitions are useful only when the additional expressivity is needed. On the Wiki2000 dataset, we were unable to reproduce the TSFlow value reported by Kollovieh et al. [38] using their specified hyperparameters. We therefore report both the published TSFlow value and the best value obtained in our reproduction.

We confirm statistically significant rank differences with a global Friedman test. A paired Wilcoxon signed-rank test between G-SLiCE and TSFlow gives a p-value of 0.055 including Wiki2000 and $p = 0.008$ excluding it. Details are in Appendix G.7. Subsequent experiments use TSFlow as the

principal flow-based baseline, with DSPD-OU included as an additional continuous-time baseline for the grid-generalisation experiments in Section 4.3.

4.2 Unconditional generation

To evaluate the unconditional generation capabilities of G-SLiCE, we sample input paths from an isotropic Gaussian process prior and train the model to generate sequences from the target data distributions. We assess the quality of the generated sequences by comparing the 2-Wasserstein distance between 10,000 generated samples and 10,000 real samples for both G-SLiCE and TSFlow. The results are reported in Table 2. Second, we evaluate the downstream predictive utility of the generated samples using the Linear Predictive Score (LPS). For this metric, we split 10,000 generated sequences into an observed context and a held-out future window, train a linear ridge regression model to predict the future from the context, and evaluate the resulting predictor on real test sequences. The LPS is defined as the test CRPS of this predictor. Results are shown in Table 3.

Table 2: Unconditional generation quality measured by 2-Wasserstein distance on eight GluonTS datasets. For each model, we generate 10,000 synthetic sequences and compare them against 10,000 real sequences from the corresponding dataset. Results are reported as mean ± standard deviation over five random seeds. Best scores are in **bold**.

Method	Electr.	Exchange	KDDCup	M4 (H)	Solar	Traffic	UberTLC	Wiki2000
TSFlow	$2.050_{\pm 0.308}$	$1.060_{\pm 0.366}$	$\mathbf{7.192_{\pm 0.481}}$	$3.687_{\pm 0.194}$	$5.758_{\pm 0.125}$	$3.971_{\pm 0.135}$	$13.521_{\pm 2.936}$	$\mathbf{13.685_{\pm 6.216}}$
G-SLiCE	$\mathbf{1.958_{\pm 0.264}}$	$\mathbf{1.059_{\pm 0.384}}$	$\mathbf{7.192_{\pm 0.481}}$	$\mathbf{3.441_{\pm 0.213}}$	$\mathbf{4.086_{\pm 0.280}}$	$\mathbf{3.175_{\pm 0.163}}$	$\mathbf{13.431_{\pm 3.052}}$	$13.874_{\pm 6.394}$

Table 3: Unconditional generation quality measured by Linear Predictive Score (LPS) on eight GluonTS datasets. For each model, we generate 10,000 synthetic sequences, split them into context and prediction windows, train a linear ridge predictor on the generated samples, and evaluate its test CRPS on real held-out sequences. Results are reported as mean ± standard deviation over five random seeds. Best scores are in **bold**.

Method	Electr.	Exchange	KDDCup	M4 (H)	Solar	Traffic	UberTLC	Wiki2000
TSFlow	$0.131_{\pm 0.035}$	$\mathbf{0.010_{\pm 0.000}}$	$\mathbf{0.466_{\pm 0.015}}$	$0.087_{\pm 0.007}$	$0.744_{\pm 0.007}$	$0.326_{\pm 0.002}$	$0.473_{\pm 0.003}$	$\mathbf{0.369_{\pm 0.006}}$
G-SLICE	$\mathbf{0.109_{\pm 0.008}}$	$\mathbf{0.010_{\pm 0.000}}$	$\mathbf{0.466_{\pm 0.015}}$	$\mathbf{0.064_{\pm 0.009}}$	$\mathbf{0.699_{\pm 0.010}}$	$\mathbf{0.246_{\pm 0.001}}$	$\mathbf{0.397_{\pm 0.004}}$	$0.371_{\pm 0.009}$

4.3 Generalisation across sampling grids

We test whether the continuous-time construction of G-SLiCE improves robustness to changes in the observation grid. We consider two grid shifts on ETTSmall15min: changes in sampling frequency and irregular subsampling. In all cases, context and prediction windows are fixed to 24 hours.

Cross-frequency generalisation. We train models on one of four uniform grids: 15-minute, hourly, 6-hourly, or 12-hourly, and evaluate each trained model on all four grids. G-SLiCE is evaluated directly on the requested time grid. For TSFlow, we report direct evaluation and a grid-mismatch adaptation that holds the latest value until a new observation is available (zero-order hold).

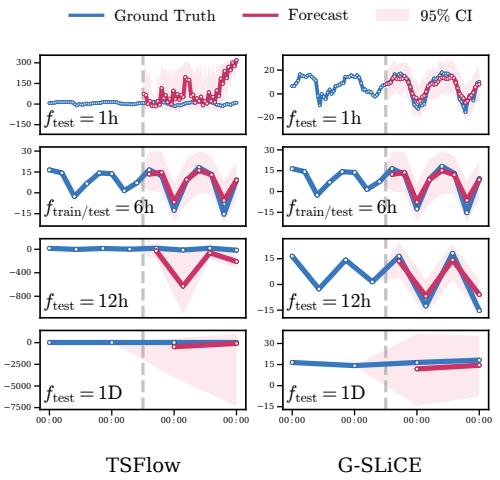


Figure 3: Representative cross-frequency forecasts on ETTSmall1h for models trained at 6-hour resolution and evaluated on different test frequencies. Curves show ground truth and predictive means; shaded regions indicate 95% confidence intervals. G-SLiCE remains stable across changes in the observation grid, while TSFlow is more sensitive to frequency shifts.

Table 4 reports CRPS. G-SLiCE remains stable across most frequency shifts, whereas direct TSFlow evaluation can fail severely when the grids differ. The strongest example is the 15-minute-train/12-hour-test setting: G-SLiCE obtains CRPS 0.189, while direct TSFlow deteriorates to 906.205. The repeat adaptation improves

TSFlow in some cases, but its cost scales with the higher resolution of the training grid rather than with the requested evaluation grid. Furthermore, this route is only available when testing on grids coarser than those trained on. Figure 3 shows representative forecasts for models trained at 6-hour resolution. Compared with DSPD-OU, G-SLiCE achieves lower CRPS in 10 of the 12 cross-frequency settings, with one tie, while DSPD-OU performs better on three of the four matched-frequency settings.

Table 4: Cross-frequency generalisation on ETTSmall15min. Each model is trained at the frequency indicated by the column and evaluated at the frequency indicated by the row. Entries report CRPS. G-SLiCE is evaluated directly on each physical time grid. For TSFlow, we report direct evaluation and adaptation by repeating the latest update. We additionally include DSPD-OU as a continuous-time baseline.

	G-SLiCE				TSFlow				TSFlow (repeat)				DSPD-OU			
test \ train	15min	1h	6h	12h	15min	1h	6h	12h	15min	1h	6h	12h	15min	1h	6h	12h
15min	0.204	0.215	0.675	0.459	0.205	0.768	24.622	7.319	0.205				0.226	0.639	3.748	5.617
1h	0.204	0.210	1.460	0.230	0.809	0.203	0.209	0.208	0.206	0.203			1.440	0.178	0.524	1.210
6h	0.205	0.208	0.216	0.220	328.794	0.645	0.206	0.206	0.210	0.206	0.206		0.926	0.290	0.129	0.220
12h	0.189	0.197	0.207	0.206	906.205	40.452	4.025	0.186	0.195	0.671	0.198	0.186	0.586	0.459	0.371	0.129

Irregular-grid generalisation. We next subsample each 24-hour window to 12 irregular observation times drawn from a Gamma renewal process. The shape parameter k controls regularity: $k = 1$ produces highly irregular grids, while larger values approach uniform spacing. Details of the construction are given in Appendix G.5. We train with $k_{\text{train}} \in \{1, 10, 100\}$ and evaluate either on matching irregular grids, $k_{\text{test}} = k_{\text{train}}$, or on the regular grid, denoted $k_{\text{test}} \to \infty$.

Table 5 reports CRPS and NRMSE over five random seeds. G-SLiCE is stable across both irregular and regular evaluation grids: its CRPS remains between 0.13 and 0.15, and its NRMSE between 0.27 and 0.33. TSFlow obtains comparable CRPS in some cases, but its NRMSE is consistently much larger, indicating unstable predictive means or large outliers. For example, at $k_{\text{train}} = 100$ on matching irregular grids, TSFlow has CRPS 0.21 and NRMSE 1.72, while G-SLiCE has CRPS 0.13 and NRMSE 0.27. G-SLiCE and DSPD-OU generally show comparable performance: G-SLiCE performs better with respect to CRPS, while DSPD-OU generally achieves lower NRMSE. Figure 4 shows representative forecasts for $k_{\text{train}} = 1$. Additional values of k are reported in Appendix E.2.

4.4 Computational cost

We additionally quantify the computation cost of both TSFlow and G-SLiCE for the experiments in Section 4.1. Across the eight evaluated dataset-specific configurations, G-SLiCE has lower rollout latency on six out of eight datasets. With 32 Euler steps on one H100 GPU, its latency ranges from 49.17 to 174.06 ms, compared with 171.15 ms for TSFlow. Training exhibits a stronger dependence on transition structure: the diagonal configurations used for Solar and Exchange train faster than TSFlow, whereas more expressive dense-block configurations incur additional training time and memory costs. The measured peak training-memory ratio ranges from $0.95\times$ to $16.83\times$ relative to TSFlow. Thus, G-SLiCE preserves parallel-in-time backbone evaluation and often reduces rollout latency, while greater transition expressivity increases training cost in the evaluated configurations. Appendix F provides the benchmark setup, complete measurements, and a discussion of these trade-offs.

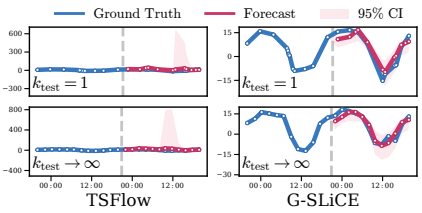


Figure 4: Example forecasts for TSFlow (left) and G-SLiCEs (right) trained with $k_{\text{train}} = 1$ and evaluated for $k_{\text{test}} = 1$ (top) or $k_{\text{test}} \to \infty$ (bottom). Curves show ground truth and predictive means; shaded regions indicate 95% confidence intervals.

5 Conclusion

This paper introduced G-SLiCE, a path-space flow-matching model with maximally expressive Structured Linear CDE backbone. Theoretically, we showed that pathwise universality implies universality of the induced pushforward laws, connecting deterministic expressivity to distributional time-series generation. On a concrete hard-core sequence example, we showed that the transition

Table 5: Comparison of G-SLiCE, TSFlow, and the continuous-time baseline DSPD-OU on the ETT 15-minute dataset from GluonTS with context and prediction lengths of 24 hours. Each sequence is subsampled to 12 random time points drawn from a Gamma distribution with shape parameter k; larger k corresponds to a more regular grid. Columns indicate the training irregularity k_{train}. For each k_{train}, we report CRPS and NRMSE separately as mean $\pm$ standard deviation. The first row per model shows i.i.d. irregular evaluation $k_{\text{test}} = k_{\text{train}}$, and the second row shows evaluation on the regular grid, $k_{\text{test}} \to \infty$.

		$k_{\text{train}} = 1$		$k_{\text{train}} = 10$		$k_{\text{train}} = 100$	
		CRPS	NRMSE	CRPS	NRMSE	CRPS	NRMSE
G-SLiCE	$k_{\text{test}} = k_{\text{train}}$	0.13 ± 0.01	0.28 ± 0.05	0.13 ± 0.01	0.32 ± 0.08	0.13 ± 0.02	0.27 ± 0.05
	$k_{\text{test}} \to \infty$	0.15 ± 0.03	0.33 ± 0.08	0.14 ± 0.01	0.27 ± 0.08	0.13 ± 0.02	0.27 ± 0.04
TSFlow	$k_{\text{test}} = k_{\text{train}}$	0.20 ± 0.07	1.01 ± 0.37	0.13 ± 0.03	0.88 ± 0.65	0.21 ± 0.06	1.72 ± 0.85
	$k_{\text{test}} \to \infty$	0.20 ± 0.07	1.24 ± 0.58	0.17 ± 0.06	1.31 ± 0.71	0.22 ± 0.06	1.75 ± 0.87
DSPD-OU	$k_{\text{test}} = k_{\text{train}}$	0.18 ± 0.00	0.33 ± 0.00	0.15 ± 0.01	0.30 ± 0.04	0.13 ± 0.00	0.25 ± 0.03
	$k_{\text{test}} \to \infty$	0.17 ± 0.00	0.32 ± 0.00	0.15 ± 0.01	0.26 ± 0.03	0.13 ± 0.00	0.24 ± 0.02

Table 6: Computational cost of TSFlow and G-SLiCE. Rollout timings use 32 Euler steps, batch size 64, and 24-step context and prediction windows on an H100 GPU. Peak memory is measured during training.

	Rollout time (ms)		Training time (s)		Peak memory (MiB)	
Dataset	TSFlow	G-SLiCE	TSFlow	G-SLiCE	TSFlow	G-SLiCE
Electricity	171.15	117.97	2.22	4.98	596.9	6073.6
Exchange	171.15	67.72	2.37	1.63	730.6	1045.9
KDD	171.15	115.98	2.55	8.00	1123.4	12314.9
M4-Hourly	171.15	136.77	2.08	4.18	1112.4	8048.2
Solar	171.15	49.17	2.25	1.35	596.9	567.9
Traffic	171.15	83.88	2.21	2.46	596.9	3081.1
Uber	171.15	173.55	2.18	6.55	596.9	9863.0
Wiki2000	171.15	174.06	2.14	7.78	730.0	12287.8

structure matters: G-SLiCE can uniformly approximate the induced distribution, whereas dense non-selective and diagonal selective exact-flow SSMs, which abstract S4D- and Mamba-style transitions, provably cannot.

Empirically, G-SLiCE is competitive with strong statistical, neural, diffusion-based, and flow-based baselines on probabilistic forecasting and unconditional time-series generation. Its continuous-time formulation also improves robustness under changes in sampling frequency and irregular observation grids: G-SLiCE can be evaluated directly on the requested physical-time grid, while fixed-grid SSM backbones require auxiliary resampling rules and often become unstable.

5.1 Limitations and future work

One avenue for future work is improving the runtime of the SLiCE backbone, which would directly speed up our method. Our current implementation uses a first-order approximation of the exponential in Equation 3. Although this supports parallel-in-time computation and parallel scan evaluation, efficient GPU kernels for structured matrix exponentials remain an important direction. Log-ODE-style approximations are another possibility, but are not currently compatible with our stacked architecture. Second, our theory does not fully characterise the expressivity of the flow-matching dynamics used in practice. ODE-based flows impose structural constraints such as invertibility, which some distributional maps lack. Although we show that G-SLiCEs can be implemented as augmented flows, a sharper theory of which path-space distributional maps are reachable by the practical training is desirable. Finally, Neural CDEs have been extended beyond Euclidean domains, including to graph-valued paths [53, 9]. Extending G-SLiCEs to structured domains is a natural next step.

Acknowledgements

We thank Marcel Kollovieh for engaging and insightful discussions and assistance with the TSFlow codebase.

This study received funding from the Klaus Tschira Stiftung gGmbH (HITS and HITS Lab). Benjamin Walker and Raeid Saqur are supported by the UK Research and Innovation (UKRI) through the Engineering and Physical Sciences Research Council (EPSRC) via Programme Grant [Grant No. UKRI1010: High order mathematical and computational infrastructure for streamed data that enhance contemporary generative and large language models]. Benjamin Walker is supported by CIMDA@Oxford, part of the AIR@InnoHK initiative funded by the Innovation and Technology Commission, HKSAR Government. Raeid Saqur acknowledges support from Canada's Vector Institute through a Vector Research Grant and the Fields Institute through an Elliott–Yui Distinguished Visitor (EYDV) appointment. Continuous Intelligence Limited is supported by SPRIND through their Next Frontier AI Challenge.

References

[1] Michael S. Albergo, Mark Goldstein, Nicholas M. Boffi, Rajesh Ranganath, and Eric Vanden-Eijnden. Stochastic interpolants with data-dependent couplings. *arXiv preprint arXiv:2310.03725*, 2023.

[2] Juan Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=hHiIbk7ApW.

[3] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *arXiv preprint arXiv:2208.09399*, 2022.

[4] J. M. Aldaz, O. Kounchev, and H. Render. Bernstein operators for exponential polynomials. *Constructive Approximation*, 29:345–367, 2009.

[5] A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic Time Series Modeling in Python. *arXiv preprint arXiv:1906.05264*, 2019.

[6] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Türkmen, and Yuyang Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116): 1–6, 2020. URL http://jmlr.org/papers/v21/19-820.html.

[7] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. In *Proceedings of the 38th Conference on Neural Information Processing Systems*, 2024.

[8] Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

[9] Torben Berndt, Benjamin Walker, Tiexin Qin, Jan Stühmer, and Andrey Kormilitzin. Permutation equivariant neural controlled differential equations for dynamic graph representation learning. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[10] Patrick Billingsley. *Convergence of Probability Measures*. Wiley, 2 edition, 1999.

[11] Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling temporal data as continuous functions with stochastic process diffusion. In *International Conference on Machine Learning*, pages 2452–2470. PMLR, 2023.

[12] Vladimir I. Bogachev. *Measure Theory*. Springer, Berlin, 2007. doi: 10.1007/978-3-540-34514-5.

[13] Ricky T. Q. Chen and Yaron Lipman. Flow matching on general geometries. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Patrick Cheridito, Hiroshi Kawaguchi, and Makoto Maejima. Fractional ornstein–uhlenbeck processes. *Electronic Journal of Probability*, 8, 2003.

[15] Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons. Theoretical foundations of deep selective state-space models. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*, 2024.

[16] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[17] Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*. JMLR.org, 2024.

[18] Abhyuday Desai, Cynthia Freeman, Zuhui Wang, and Ian Beaver. Timevae: A variational auto-encoder for multivariate time series generation. *arXiv preprint arXiv:2111.08095*, 2021.

[19] Dua Dheeru and E. Karra Taniskidou. Uci machine learning repository, 2017.

[20] Joseph L. Doob. The brownian movement and stochastic equations. *Annals of Mathematics*, 43 (2), 1942.

[21] Dheeru Dua, Casey Graff, et al. Uci machine learning repository. Beijing, 2017.

[22] Ting-Han Fan, Ta-Chung Chi, and Alexander Rudnicky. Advancing regular language reasoning in linear recurrent neural networks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2: Short Papers*, pages 45–53, Mexico City, Mexico, 2024. Association for Computational Linguistics.

[23] FiveThirtyEight. Uber tlc foil response. https://github.com/fivethirtyeight/uber-tlc-foil-response, 2016.

[24] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL http://jmlr.org/papers/v22/20-451.html.

[25] Rémi Flamary, Cédric Vincent-Cuaz, Nicolas Courty, Alexandre Gramfort, Oleksii Kachaiev, Huy Quang Tran, Laurène David, Clément Bonet, Nathan Cassereau, Théo Gnassounou, Eloi Tanguy, Julie Delon, Antoine Collas, Sonia Mazelet, Laetitia Chapel, Tanguy Kerdoncuff, Xizheng Yu, Matthew Feickert, Paul Krzakala, Tianlin Liu, and Eduardo Fernandes Montesuma. Pot python optimal transport (version 0.9.5), 2024. URL https://github.com/PythonOT/POT.

[26] Gerald B. Folland. *Real Analysis: Modern Techniques and Their Applications*. Wiley, 2 edition, 1999.

[27] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function rnns. In *The 22nd international conference on artificial intelligence and statistics*, pages 1901–1910. PMLR, 2019.

[28] Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[29] Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[30] Rakshitha Godahewa, Christoph Bergmeir, Geoffrey I. Webb, Rob J. Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

[31] Rakshitha Wathsadini Godahewa, Christoph Bergmeir, Geoffrey I Webb, Rob Hyndman, and Pablo Montero-Manso. Monash time series forecasting archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[32] Xavier Gonzalez, Andrew Warrington, Jimmy T. H. Smith, and Scott W. Linderman. Towards scalable and stable parallelization of nonlinear rnns. In *Advances in Neural Information Processing Systems*, 2024.

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

[34] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[35] Rob Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

[36] Gavin Kerrigan, Giosue Migliorini, and Padhraic Smyth. Functional flow matching. *arXiv preprint arXiv:2305.17209*, 2023.

[37] 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, 2023.

[38] Marcel Kollovieh, Marten Lienen, David Lüdke, Leo Schwinn, and Stephan Günnemann. Flow matching with gaussian process priors for probabilistic time series forecasting. *The Thirteenth International Conference on Learning Representations*, 2025.

[39] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[40] Bryan Lim, Sercan Ö. Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4): 1748–1764, 2021.

[41] Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing nonlinear sequential models over the sequence length. *International Conference on Learning Representations*, 2024.

[42] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[43] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020.

[44] Sajad Movahedi, Felix Sarnthein, Nicola Muca Cirone, and Antonio Orvieto. Fixed-point rnns: From diagonal to dense in a few iterations. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[45] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[46] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[47] Antonio Orvieto, Soham De, Caglar Gulcehre, Razvan Pascanu, and Samuel L. Smith. Universality of linear recurrences followed by non-linear projections: Finite-width guarantees and benefits of complex eigenvalues. In *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[49] Bo Peng, Ruichong Zhang, Daniel Goldstein, Eric Alcaide, Xingjian Du, Haowen Hou, Jiaju Lin, Jiaxing Liu, Janna Lu, William Merrill, Guangyu Song, Kaifeng Tan, Saiteja Utpala, Nathan Wilce, Johan S. Wind, Tianyi Wu, Daniel Wuttke, and Christian Zhou-Zheng. Rwkv-7 "goose" with expressive dynamic state evolution. *arXiv preprint arXiv:2503.14456*, 2025.

[50] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. Random feature attention. In *International Conference on Learning Representations*, 2021.

[51] Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter Battaglia, et al. Probabilistic weather forecasting with machine learning. *Nature*, 637(8044):84–90, 2025.

[52] Weizhu Qian, Dalin Zhang, Yan Zhao, Kai Zheng, and James JQ Yu. Uncertainty quantification for traffic forecasting: A unified approach. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 992–1004. IEEE, 2023.

[53] Tiexin Qin, Benjamin Walker, Terry Lyons, Hong Yan, and Haoliang Li. Learning dynamic graph embeddings with neural controlled differential equations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[54] Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

[55] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[56] 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*, pages 8857–8868. PMLR, 2021.

[57] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

[58] Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Imprpoving state-tracking in linear rnns via householder products. *arXiv preprint arXiv:2502.10297*, 2025.

[59] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[60] Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2025.

[61] Souhaib Ben Taieb, Raphael Huser, Rob J Hyndman, Marc G Genton, et al. Probabilistic time series forecasting with boosted additive models: an application to smart meter data. *Department of Economics and business statistics, Monash University*, 2015.

[62] 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.

[63] Alexander Tong, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Kilian FATRAS, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. In *ICML Workshop on New Frontiers in Learning, Control, and Dynamical Systems*, 2023.

[64] Cédric Villani. *Optimal Transport: Old and New*. Springer, Berlin, Heidelberg, 2009. doi: 10.1007/978-3-540-71050-9.

[65] Benjamin Walker, Lingyi Yang, Nicola Muca Cirone, Cristopher Salvi, and Terry Lyons. Structured linear cdes: Maximally expressive and parallel-in-time sequence models. In *Proceedings of the 39th Conference on Neural Information Processing Systems (NeurIPS)*, 2025.

[66] Shida Wang and Beichen Xue. State-space models with layer-wise nonlinearity are universal approximators with exponential decaying memory. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[67] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

[68] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2024.

[69] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[70] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023.

[71] Yu Zhang, Songlin Yang, Ruijie Zhu, Yue Zhang, Leyang Cui, Yiqiao Wang, Bolun Wang, Freda Shi, Bailin Wang, Wei Bi, Peng Zhou, and Guohong Fu. Gated slot attention for efficient linear-time sequence modeling. In *Proceedings of the Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[72] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press, 2021.

Appendices

Appendix Contents

A Universal Time Series Generation **16**

A.1 From Maximal Expressivity to Distributional Approximation 16

A.2 Path-to-Path Models 19

B Proof that SLiCEs are path-to-path universal **21**

C Proof of expressivity gap example **21**

D G-SLiCE as an augmented path-space flow **23**

E Additional results **24**

E.1 Ablation on importance of dense blocks 24

E.2 Additional results for irregular sampling generalisation 24

F Computational cost: runtime and memory **25**

G Experimental details **26**

G.1 Expressivity Gap: Hard-core Example 26

G.2 Time Series Forecasting: Datasets 27

G.3 Training protocol and hyperparameters 28

G.4 Metrics 30

G.5 Construction of irregular sampling grid 31

G.6 Compute resources 31

G.7 Statistical tests 32

A Universal Time Series Generation

This section makes precise the sense in which Linear NCDEs are universal models for time series generation: they can approximate any target path law obtained by applying a continuous causal transformation to a latent path law. Under compact support, this approximation holds with exact W_∞ control. For tight non-compact latent laws, the same argument yields high-probability approximation on compact sets carrying arbitrarily large mass. First, we show that in a general setting, maximal expressivity at the level of deterministic functions implies approximation of the corresponding pushforward distributions in W_∞ whenever the input law is concentrated on a compact set. We then apply this observation to path-to-path models, using recent universality results for Linear NCDEs with non-linear readouts to obtain the corresponding distributional statement on path space. Finally, we contrast this with S4 and Mamba, and discuss how these ideas apply in the conditional Gaussian process setting used in practice, where interpolation and non-compact latent laws introduce additional technical considerations.

A.1 From Maximal Expressivity to Distributional Approximation

This subsection explains how a deterministic approximation statement can be converted into a distributional approximation result. The starting point is maximal expressivity, which is the ability to approximate continuous functions uniformly on compact sets. The conclusion is that, when

the input law is concentrated on a compact set and the target map is continuous on that set, the pushforward distributions induced by the model can approximate the target pushforward distribution in W_∞. The argument is elementary, but it is useful to state it explicitly because it clarifies how approximation properties of functions transfer to approximation properties of the corresponding pushforward measures.

Definition A.1 (Maximal expressivity [65])**.** Let $(\mathcal{X}, \rho_{\mathcal{X}})$ and $(\mathcal{Y}, \rho_{\mathcal{Y}})$ be metric spaces, and let $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}$ be a class of functions. We say that $\mathcal{F}$ is maximally expressive, or universal, if for every compact set $\mathcal{K} \subset \mathcal{X}$ and every continuous function $f : \mathcal{K} \to \mathcal{Y}$,

$$\forall \varepsilon > 0,\ \exists \theta \in \Theta \quad \text{s.t.} \quad \sup_{x \in \mathcal{K}} \rho_{\mathcal{Y}}\big(f(x), f_\theta(x)\big) \le \varepsilon. \tag{14}$$

When $\mathcal{X} = \mathbb{R}^d$ with $\rho_{\mathcal{X}}(x_1, x_2) = \|x_1 - x_2\|_2$ and $\mathcal{Y} = \mathbb{R}$ with $\rho_{\mathcal{Y}}(y_1, y_2) = |y_1 - y_2|$, Definition A.1 is the standard requirement of uniform approximation on compact subsets of $\mathbb{R}^d$. In this setting, classical universal approximation theorems show that multi-layer perceptrons with suitable activation functions form a maximally expressive function class in the sense of Definition A.1 [16, 34]. Our goal here is to show that this notion of uniform approximation also yields a natural approximation guarantee at the level of probability measures.

To state this precisely, we recall the basic measure-theoretic notions appearing in the argument. These definitions are standard, but we include them for completeness and to keep the appendix self-contained. We begin with the underlying measurable structure.

Definition A.2 (σ-algebra [12])**.** Let $\mathcal{X}$ be a set. A collection Σ of subsets of $\mathcal{X}$ is called a σ-algebra on $\mathcal{X}$ if

1. $\mathcal{X} \in \Sigma$,
2. if $A \in \Sigma$ then $\mathcal{X} \setminus A \in \Sigma$,
3. if $(A_n)_{n \ge 1}$ is a sequence of sets in Σ, then $\bigcup_{n \ge 1} A_n \in \Sigma$.

The pair $(\mathcal{X}, \Sigma)$ is called a measurable space.

A σ-algebra specifies which subsets of $\mathcal{X}$ are measurable, and therefore which events can be assigned probabilities. Since $\mathcal{X}$ is assumed to be a metric space, there is a canonical choice of measurable structure, namely the Borel σ-algebra generated by the open sets.

Definition A.3 (Borel σ-algebra [12])**.** Let $(\mathcal{X}, \rho_{\mathcal{X}})$ be a metric space. The Borel σ-algebra on $\mathcal{X}$, denoted $\mathcal{B}(\mathcal{X})$, is the smallest σ-algebra containing all open subsets of $\mathcal{X}$.

Once the measurable structure has been fixed, we can speak of probability measures on $\mathcal{X}$.

Definition A.4 (Borel probability measure [12])**.** Let $(\mathcal{X}, \rho_{\mathcal{X}})$ be a metric space. A Borel probability measure on $\mathcal{X}$ is a probability measure on the measurable space $(\mathcal{X}, \mathcal{B}(\mathcal{X}))$.

The next notion records where a measure is locally non-trivial.

Definition A.5 (Support [12])**.** Let μ be a Borel probability measure on a metric space $(\mathcal{X}, \rho_{\mathcal{X}})$. The support of μ is

$$\operatorname{supp}(\mu) = \big\{x \in \mathcal{X} :\ \mu\big(B_{\rho_{\mathcal{X}}}(x, r)\big) > 0 \text{ for all } r > 0\big\}. \tag{15}$$

Given a measurable map $T : \mathcal{X} \to \mathcal{Y}$ and an input law μ on $\mathcal{X}$, the natural output law is the distribution obtained by transporting μ through T. This is captured by the pushforward measure.

Definition A.6 (Pushforward measure [12])**.** Let μ be a Borel probability measure on $\mathcal{X}$ and let $T : \mathcal{X} \to \mathcal{Y}$ be Borel measurable. The pushforward of μ by T is the Borel probability measure $T_{\#}\mu$ on $\mathcal{Y}$ defined by

$$(T_{\#}\mu)(A) = \mu\big(T^{-1}(A)\big), \qquad A \in \mathcal{B}(\mathcal{Y}). \tag{16}$$

Thus, if $X \sim \mu$, then $T_{\#}\mu$ is simply the law of the random variable $T(X)$. In our setting, T will denote the target transformation and f_θ a model approximation to T. The question is whether closeness of f_θ to T at the level of points implies closeness of $(f_\theta)_{\#}\mu$ to $T_{\#}\mu$ at the level of probability measures.

To measure this closeness between probability measures, we use the ∞-Wasserstein distance on the output space $\mathcal{Y}$. This distance records the smallest possible essential worst-case transport cost over all couplings of the two measures. For the W_∞ argument below, we additionally assume that the output space $(\mathcal{Y}, \rho_\mathcal{Y})$ is separable, meaning that it contains a countable dense subset.

Definition A.7 (W_∞ [64])**.** Let $(\mathcal{Y}, \rho_\mathcal{Y})$ be a separable metric space, and let μ, ν be Borel probability measures on $\mathcal{Y}$. A coupling of μ and ν is a Borel probability measure π on $\mathcal{Y} \times \mathcal{Y}$ whose first marginal is μ and whose second marginal is ν. Write $\Pi(\mu, \nu)$ for the set of all such couplings. The ∞-Wasserstein distance, possibly taking the value $+\infty$, is

$$W_\infty(\mu, \nu) = \inf_{\pi \in \Pi(\mu,\nu)} \operatorname*{ess\,sup}_{(y,z)\sim\pi} \rho_\mathcal{Y}(y, z). \tag{17}$$

The relevance of W_∞ here is that it interacts very naturally with uniform approximation. If two functions are uniformly close on a set of full μ-measure, then applying them to the same input sample immediately produces a coupling whose transport cost is uniformly controlled. The next lemma formalises this simple observation.

Lemma 4 (W_∞ control via a pointwise transport bound)**.** *Let $(\mathcal{X}, \rho_\mathcal{X})$ be a metric space, let $(\mathcal{Y}, \rho_\mathcal{Y})$ be a separable metric space, and let μ be a Borel probability measure on $\mathcal{X}$. Let $T : \mathcal{X} \to \mathcal{Y}$ and $f : \mathcal{X} \to \mathcal{Y}$ be Borel measurable. Then*

$$W_\infty(f_\#\mu, T_\#\mu) \le \operatorname*{ess\,sup}_{x\sim\mu} \rho_\mathcal{Y}\big(f(x), T(x)\big). \tag{18}$$

Proof. Since $\mathcal{Y}$ is separable, the map $x \mapsto (f(x), T(x))$ from $\mathcal{X}$ to $\mathcal{Y} \times \mathcal{Y}$ is Borel measurable. Define

$$\pi = (f, T)_\#\mu. \tag{19}$$

Then $\pi \in \Pi(f_\#\mu, T_\#\mu)$, and

$$\operatorname*{ess\,sup}_{(y,z)\sim\pi} \rho_\mathcal{Y}(y, z) = \operatorname*{ess\,sup}_{x\sim\mu} \rho_\mathcal{Y}\big(f(x), T(x)\big). \tag{20}$$

Taking the infimum over all couplings in (17) yields (18). □

The lemma shows that distributional approximation in W_∞ follows immediately from an almost-sure pointwise bound. We now combine this observation with maximal expressivity. The key approximation-theoretic assumptions in the corollary are that the input law is concentrated on a compact set and that the target map is continuous on that set.

Corollary 4.1 (Distributional approximation from maximal expressivity)**.** *Let $(\mathcal{X}, \rho_\mathcal{X})$ be a metric space, let $(\mathcal{Y}, \rho_\mathcal{Y})$ be a separable metric space, and let $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}$, where each f_θ is Borel measurable, be maximally expressive in the sense of Definition A.1. Let μ be a Borel probability measure on $\mathcal{X}$ such that $\mu(\mathcal{K}) = 1$ for some compact set $\mathcal{K} \subset \mathcal{X}$. Let $T : \mathcal{X} \to \mathcal{Y}$ be Borel measurable and assume that $T|_\mathcal{K} : \mathcal{K} \to \mathcal{Y}$ is continuous. Then for every $\varepsilon > 0$ there exists $\theta \in \Theta$ such that*

$$W_\infty\big((f_\theta)_\#\mu, T_\#\mu\big) \le \varepsilon. \tag{21}$$

Proof. Fix $\varepsilon > 0$. By maximal expressivity applied to the compact set $\mathcal{K}$ and the continuous map $T|_\mathcal{K} : \mathcal{K} \to \mathcal{Y}$, choose $\theta \in \Theta$ such that

$$\sup_{x\in\mathcal{K}} \rho_\mathcal{Y}\big(f_\theta(x), T(x)\big) \le \varepsilon. \tag{22}$$

Since $\mu(\mathcal{K}) = 1$, it follows that

$$\operatorname*{ess\,sup}_{x\sim\mu} \rho_\mathcal{Y}\big(f_\theta(x), T(x)\big) \le \varepsilon. \tag{23}$$

Applying Lemma 4 with $f = f_\theta$ yields (21). □

Corollary 4.1 shows that maximal expressivity at the level of deterministic functions automatically yields a corresponding universality statement for pushforward measures generated from input laws that are concentrated on a compact set, provided the target map is Borel measurable on $\mathcal{X}$ and continuous on a compact full-measure set. In other words, once a model class can approximate continuous

functions uniformly on compact sets, it can also approximate the distributional transformations induced by such functions on any input law μ satisfying $\mu(\mathcal{K}) = 1$ for some compact $\mathcal{K} \subset \mathcal{X}$.

This observation is useful in machine learning settings where one is interested not only in approximating a target map pointwise, but also in reproducing the distribution of outputs generated by that map over a population of inputs. The corollary shows that, under concentration on a compact full-measure set and continuity on that set, no separate distributional approximation theorem is needed: it follows directly from the standard uniform approximation property together with the elementary coupling argument of Lemma 4.

A.2 Path-to-Path Models

We now specialise to path space. Throughout this subsection, let $\mathcal{X}(d)$ denote the space

$$\mathcal{X}(d) = C^{1,0}([t_0, t_n], \mathbb{R}^d), \tag{24}$$

consisting of absolutely continuous, time-augmented d-dimensional paths on the interval $[t_0, t_n]$ which all begin at the same point, endowed with the 1-variation topology. For output paths, we use the supremum metric

$$\rho_\infty(Y, \widetilde{Y}) = \sup_{t \in [t_0, t_n]} \|Y_t - \widetilde{Y}_t\|_2, \qquad Y, \widetilde{Y} \in C([t_0, t_n], \mathbb{R}^{d_y}). \tag{25}$$

We next define a Linear NCDE with a generic readout.

Definition A.8 (Linear NCDE [15, 65])**.** Let $L_\theta^1 \in \mathbb{R}^{d_h \times d_X}$, $b_\theta \in \mathbb{R}^{d_h}$, and $A_\theta \in \mathbb{R}^{d_h \times d_X \times d_h}$ be trainable parameters, and let

$$R_\theta : \mathbb{R}^{d_h} \to \mathbb{R}^{d_y} \tag{26}$$

be a readout map. Then the corresponding Linear NCDE is defined by

$$\begin{aligned} h_{t_0} &= L_\theta^1 X_{t_0} + b_\theta, \\ h_t &= h_{t_0} + \int_{t_0}^{t} A_\theta h_s \, \mathrm{d}X_s, \\ Y_t^\theta(X) &= R_\theta(h_t). \end{aligned} \tag{27}$$

When R_θ is linear, Linear NCDEs are universal for terminal-time path-to-point functions [15]. Allowing R_θ to be non-linear extends this to causal path-to-path functions.

Definition A.9 (Causal path-to-path functions)**.** Let

$$T : \mathcal{X}(d_X) \to C([t_0, t_n], \mathbb{R}^{d_y}). \tag{28}$$

We say that T is causal if, for every $t \in [t_0, t_n]$ and every $X, \widetilde{X} \in \mathcal{X}(d_X)$,

$$X|_{[t_0,t]} = \widetilde{X}|_{[t_0,t]} \quad \Longrightarrow \quad T(X)_t = T(\widetilde{X})_t. \tag{29}$$

The following theorem is obtained by combining the scalar-valued path-to-path argument of Cirone et al. [15, Proposition 5.1 and Proposition D.2] with the homogeneous Linear NCDE maximal-expressivity framework of Walker et al. [65], after the obvious affine rescaling of time from $[t_0, t_n]$ to $[0, 1]$, and then applying the resulting scalar statement coordinate-wise, with scalar tolerance $\varepsilon/\sqrt{d_y}$. Although some of the intermediate Linear CDE statements in Cirone et al. [15] allow an additive controlled term, this does not change the present homogeneous formulation, as any affine linear controlled system can be written as a homogeneous linear controlled system after augmenting the hidden state by a coordinate initialised to one and assigning that coordinate zero dynamics. Since paths in $\mathcal{X}(d_X)$ are time-augmented, time is included as a channel of X.

Theorem 5 (Universality for causal path-to-path functions)**.** *Let*

$$T : \mathcal{X}(d_X) \to C([t_0, t_n], \mathbb{R}^{d_y}) \tag{30}$$

be continuous and causal, and let $\mathcal{K} \subset \mathcal{X}(d_X)$ be compact. Consider Linear NCDEs as in Definition A.8, where R_θ is a feed-forward neural network. Then for every $\varepsilon > 0$ there exist a hidden dimension $d_h \in \mathbb{N}$ and model parameters such that

$$\sup_{X \in \mathcal{K}} \rho_\infty\big(Y^\theta(X), T(X)\big) \le \varepsilon. \tag{31}$$

Since $C([t_0, t_n], \mathbb{R}^{d_y})$ endowed with ρ_∞ is separable, Theorem 5 and Lemma 4 imply the corresponding causal distributional statement. Indeed, if μ is a Borel probability measure on $\mathcal{X}(d_X)$ satisfying $\mu(\mathcal{K}) = 1$ for some compact set $\mathcal{K} \subset \mathcal{X}(d_X)$, and if

$$T : \mathcal{X}(d_X) \to C([t_0, t_n], \mathbb{R}^{d_y}) \tag{32}$$

is continuous and causal, then for every $\varepsilon > 0$ there exists a Linear NCDE with a feed-forward readout such that

$$W_\infty\big((Y^\theta)_\# \mu, T_\# \mu\big) \leq \varepsilon, \tag{33}$$

where W_∞ is computed on $C([t_0, t_n], \mathbb{R}^{d_y})$ with respect to the metric ρ_∞.

In contrast, the analogous causal path-to-path universality statement is not currently available for standard SSM architectures such as S4 and Mamba. Existing negative results show that single-layer S4 and Mamba with linear readouts are not universal for terminal-time path-to-point functions, which is one of the key ingredients used to obtain the causal path-to-path universality result for Linear NCDEs [15]. Moreover, while universality results do exist for discrete sequence-to-sequence SSMs with non-linear projections or layer-wise nonlinearities [47, 66], these results do not give a continuous causal path-to-path approximation theorem with respect to the supremum metric. Thus they do not directly yield the W_∞ pushforward statement used above. To the best of our knowledge, no directly analogous path-to-path universality theorem is known for stacked S4 or Mamba-style models in the continuous path-space setting considered here.

In practice, the model used in this work is conditional rather than unconditional. Given an observed input time series, we first form a conditional Gaussian process and sample a latent trajectory from its conditional law. This latent trajectory is then sampled on a finite time grid and converted into a path in $\mathcal{X}(d_X)$ by a deterministic interpolation and augmentation map

$$\mathcal{I} : C([t_0, t_n], \mathbb{R}^{d_z}) \to \mathcal{X}(d_X), \tag{34}$$

where d_z is the latent Gaussian process dimension and d_X includes the deterministic augmentation channels. We assume that $\mathcal{I}$ is continuous from the supremum topology on $C([t_0, t_n], \mathbb{R}^{d_z})$ to the 1-variation topology on $\mathcal{X}(d_X)$, as is the case for fixed-grid piecewise linear interpolation together with fixed deterministic augmentation channels. The map $\mathcal{I}$ includes the time channel and any additional deterministic preprocessing needed to ensure that its image lies in $\mathcal{X}(d_X)$, for example a constant channel when the homogeneous initialisation requires one. In particular, the common-initial-point convention is imposed on the interpolated and augmented paths actually fed to the model, not necessarily on the raw Gaussian process sample paths.

This is important, as sample paths from a latent Gaussian process with an Ornstein–Uhlenbeck kernel are almost surely continuous but not absolutely continuous [20, 14]. Thus the raw latent law is not supported on the absolutely continuous path space processed by the NCDE. However, the interpolated and augmented latent law $\mathcal{I}_\# \mu$ is supported on $\mathcal{X}(d_X)$, and this is the object that is actually processed by the model. The interpolated path is then passed through a stacked Linear NCDE. This stacked architecture should be viewed as the practical analogue of the approximation mechanism described above. The first Linear NCDE block produces a hidden path, while the later blocks provide additional non-linear transformations of this hidden path. When the architecture includes sufficiently expressive pointwise readouts, this falls directly under the preceding approximation argument. Conditional on a fixed observed time series, the model defines a distribution on output paths by pushing the conditional interpolated latent law forward through this deterministic stacked Linear NCDE map. From this perspective, the conditional Gaussian process provides the source of pathwise randomness, while the stacked Linear NCDE provides the expressive mechanism that reshapes this randomness into the target output process.

A second important point to note is that the compact-support assumption in the exact W_∞ statement above does not hold, since $\mathcal{I}_\# \mu$ is typically not compactly supported. However, this is mainly a technical distinction rather than a practical obstruction. The original conditional Gaussian process law is a Borel probability measure on $C([t_0, t_n], \mathbb{R}^{d_z})$. Equipped with the supremum norm, this space is Polish, as it is complete by Folland [26, Proposition 4.13], and separable by the Stone–Weierstrass theorem [26, Theorem 4.45]. Hence the law is tight [10, Theorem 1.3]. Consequently, for every $\delta > 0$ there exists a compact set $\mathcal{K}$ of latent paths with $\mu(\mathcal{K}) \geq 1 - \delta$. Since $\mathcal{I}$ is continuous, $\mathcal{I}(\mathcal{K})$ is compact in $\mathcal{X}(d_X)$ and

$$(\mathcal{I}_\# \mu)\big(\mathcal{I}(\mathcal{K})\big) \geq 1 - \delta. \tag{35}$$

Applying the same approximation argument on $\mathcal{I}(\mathcal{K})$ then shows that, for every $\varepsilon, \delta > 0$, and for every continuous causal target map

$$T : \mathcal{X}(d_X) \to C([t_0, t_n], \mathbb{R}^{d_y}), \tag{36}$$

one can choose parameters θ such that

$$(\mathcal{I}_{\#}\mu)\Big(\big\{Z \in \mathcal{X}(d_X) : \rho_\infty\big(F_\theta(Z), T(Z)\big) > \varepsilon\big\}\Big) \leq \delta, \tag{37}$$

where F_θ denotes the path-to-path map induced by the stacked Linear NCDE. In other words, although one does not obtain a global W_∞ statement for a non-compact latent law, the model can still approximate the target transformation arbitrarily well on an arbitrarily high-probability region of latent space. This is the regime that matters in practice, since training and evaluation only ever involve finitely many sampled and discretised trajectories, while the non-compact Gaussian tails correspond to rare excursions rather than a fundamental modelling obstruction.

B Proof that SLiCEs are path-to-path universal

Proof. Cirone et al. [15, Proposition D.2] prove the corresponding path-to-path result for dense Linear NCDEs by reducing causal path-to-path approximation to path-to-point approximation on stopped prefixes. The proof uses dense Linear NCDEs only through their terminal path-to-point approximation property. Replacing them by a SLiCE class with the same property gives the same construction. □

C Proof of expressivity gap example

We introduce the setting of the example in Section 2.3 in slightly more detail: Denote by $\mathcal{H}$ the set of binary sequences of length n with no consecutive ones:

$$\mathcal{H}_n := \big\{c = (c_1, \dots, c_n) \in \{0,1\}^n : c_i c_{i+1} = 0 \text{ for all } 1 \leq i \leq n-1\big\}. \tag{38}$$

Then we sample n independently and identically distributed binary Bernoulli Random variables Z_i

$$Z_1, \dots, Z_n \overset{\text{i.i.d.}}{\sim} \text{Bernoulli}(p), \qquad p \in [0,1],$$

and define the target map $C = (C_1, \dots, C_n)$ as the recursion

$$C_1 = Z_1, \qquad C_k = Z_k(1 - C_{k-1}) \quad (2 \leq k \leq n). \tag{39}$$

By construction, $C \in \mathcal{H}_n$ for every realisation and we write

$$\mu_{n,p} := C_\sharp\big(\text{Bernoulli}(p)^{\otimes n}\big)$$

for the induced law on $\mathcal{H}_n$.

We now prove Proposition 2.3. Throughout this proof, sequence space is equipped with

$$d_\infty(c, \tilde{c}) = \max_{1 \leq k \leq n} |c_k - \tilde{c}_k|.$$

For the dense selective upper bound we prove a uniform pathwise error bound over all $Z \in \{0,1\}^n$, which implies the stated W_∞ bound by coupling each generated sequence with the target sequence obtained from the same input Z.

Dense selective. First, we prove the dense selective upper bound. Let

$$J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}, \qquad e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Choose $0 < \eta < \min\{\varepsilon, 1/2\}$, and define affine maps by

$$A(z) = (1 - z)(\log \eta) I_2 + z\pi J, \qquad \beta(z) = z e_1.$$

Set $h_0 = 0$, $w = e_1$, and $b = 0$. Then

$$\exp(A(0)) = \eta I_2, \qquad \exp(A(1)) = -I_2.$$

Writing $x_k = e_1^\top h_k$, the output satisfies

$$x_k = \begin{cases} \eta x_{k-1}, & Z_k = 0, \\ 1 - x_{k-1}, & Z_k = 1. \end{cases}$$

The target satisfies the same recursion on one-input steps, while on zero-input steps it satisfies $C_k = 0$. Since $x_0 = 0$ and $0 < \eta < 1$, induction gives $x_k \in [0, 1]$ for every k.

Let $E_k = x_k - C_k$. If $Z_k = 0$, then $C_k = 0$ and

$$|E_k| = |x_k| = \eta x_{k-1} \le \eta.$$

If $Z_k = 1$, then

$$E_k = (1 - x_{k-1}) - (1 - C_{k-1}) = -E_{k-1}.$$

Since $E_1 = 0$, it follows that $|E_k| \le \eta$ for every k and every binary input Z. Therefore

$$\max_{Z \in \{0,1\}^n} \max_{1 \le k \le n} |\widehat{C}_k - C_k| \le \eta < \varepsilon.$$

Because $\eta < 1/2$, thresholding $\widehat{C}_k = x_k$ at $1/2$ recovers C_k exactly.

Dense non-selective. Consider a dense non-selective exact-flow SSM. It has the form

$$h_k = M h_{k-1} + \beta(Z_k), \qquad \widehat{C}_k = w^\top h_k + b,$$

with fixed M, affine β, and h_0 fixed independently of the input. Unrolling gives

$$h_k = M^k h_0 + \sum_{j=1}^{k} M^{k-j} \beta(Z_j).$$

Hence $\widehat{C}_1$ is affine in Z_1, and $\widehat{C}_2$ is affine in (Z_1, Z_2). Writing

$$Y_{z_1 z_2} = \big(\widehat{C}_1(z_1, z_2), \widehat{C}_2(z_1, z_2)\big), \qquad (z_1, z_2) \in \{0, 1\}^2,$$

we therefore have the parallelogram identity

$$Y_{00} + Y_{11} - Y_{10} - Y_{01} = 0.$$

It remains to turn this pointwise affine obstruction into a distributional W_∞ obstruction. The projection onto the first two coordinates is 1-Lipschitz, so a W_∞ approximation of the full length-n law with error r would imply a W_∞ approximation of the projected two-step laws with error at most r.

For $p = 1/2$, the target two-step law is the uniform law on the labelled multiset

$$T_{00} = (0, 0), \qquad T_{01} = (0, 1), \qquad T_{10} = (1, 0), \qquad T_{11} = (1, 0).$$

Suppose, for contradiction, that the projected generated law is within W_∞ distance $r < 1/4$ of this target law. The open d_∞-balls of radius r around the three distinct target atoms $(0, 0)$, $(0, 1)$, and $(1, 0)$ are disjoint. Hence the four generated atoms can be assigned, counting multiplicity, to the target multiset above so that

$$d_\infty(Y_{z_1 z_2}, \widetilde{T}_{z_1 z_2}) < r$$

for some relabelling $\{\widetilde{T}_{00}, \widetilde{T}_{01}, \widetilde{T}_{10}, \widetilde{T}_{11}\}$ of the multiset $\{(0, 0), (0, 1), (1, 0), (1, 0)\}$.

For any such relabelling,

$$\left\| \widetilde{T}_{00} + \widetilde{T}_{11} - \widetilde{T}_{10} - \widetilde{T}_{01} \right\|_\infty \ge 1.$$

Indeed, the two plus-labelled targets and the two minus-labelled targets form a partition of the multiset $\{(0, 0), (0, 1), (1, 0), (1, 0)\}$, and their sums cannot agree. Using the parallelogram identity for the generated atoms,

$$\begin{aligned} 1 &\le \left\| \widetilde{T}_{00} + \widetilde{T}_{11} - \widetilde{T}_{10} - \widetilde{T}_{01} \right\|_\infty \\ &= \left\| (\widetilde{T}_{00} - Y_{00}) + (\widetilde{T}_{11} - Y_{11}) - (\widetilde{T}_{10} - Y_{10}) - (\widetilde{T}_{01} - Y_{01}) \right\|_\infty \\ &< 4r, \end{aligned}$$

which contradicts $r < 1/4$. Therefore

$$W_\infty(\widehat{\mu}_{n,1/2}, \mu_{n,1/2}) \ge \frac{1}{4}.$$

Diagonal selective. Finally consider a width d diagonal selective exact-flow SSM. It can be written as

$$h_k = D(Z_k)h_{k-1} + \beta(Z_k), \qquad D(z) = \mathrm{diag}(\lambda_1(z), \dots, \lambda_d(z)), \qquad \lambda_i(z) > 0.$$

For $p = 1$, the input is the all-ones sequence almost surely, so both the target law and the generated law are Dirac measures. Thus a W_∞ error strictly smaller than $1/2$ is equivalent to pointwise approximation of the all-ones target sequence with error strictly smaller than $1/2$. On the all-ones input, this becomes

$$h_k = D(1)h_{k-1} + \beta(1).$$

For a coordinate with diagonal value $\lambda \neq 1$, the scalar recursion contributes a constant plus a multiple of λ^k. For a coordinate with $\lambda = 1$, it contributes a constant plus a multiple of k. Therefore every linear readout on the all-ones input has the form

$$\widehat{C}_k = a_0 + a_1 k + \sum_{\lambda \in \Lambda} a_\lambda \lambda^k, \qquad \lambda > 0, \qquad \lambda \neq 1, \qquad |\Lambda| + \mathbf{1}_{\{a_1 \neq 0\}} \leq d,$$

where Λ is the set of distinct non-unit diagonal values that contribute to the readout.

Subtracting $1/2$ preserves this form. Writing $\alpha = \log \lambda$, we have

$$\widehat{C}_k - \frac{1}{2} = a_0 + a_1 k + \sum_{\alpha \in \Gamma} a_\alpha e^{\alpha k}, \qquad \alpha \in \mathbb{R} \setminus \{0\}, \qquad |\Gamma| + \mathbf{1}_{\{a_1 \neq 0\}} \leq d.$$

Let

$$f(t) = a_0 + a_1 t + \sum_{\alpha \in \Gamma} a_\alpha e^{\alpha t}.$$

This belongs to the real exponential-polynomial space generated by $1, t$, and $e^{\alpha t}$ for $\alpha \in \Gamma$. This space is an extended complete Chebyshev system, and by the standard zero-counting property of such systems, if f is not identically zero, then f has at most

$$|\Gamma| + \mathbf{1}_{\{a_1 \neq 0\}}$$

zeros on any interval [4]. Since $\widehat{C}_k - 1/2 = f(k)$, each sign change of the nonzero sampled sequence $\widehat{C}_1 - 1/2, \dots, \widehat{C}_n - 1/2$ gives a zero of f in the corresponding interval $(k, k+1)$. Hence $\widehat{C}_k - 1/2$ has at most d sign changes.

On the all-ones input, the target is

$$C_k = \begin{cases} 1, & k \text{ odd}, \\ 0, & k \text{ even}. \end{cases}$$

Thus $C_k - 1/2$ changes sign at every step. If the diagonal selective model approximated the target with error strictly less than $1/2$, then $\widehat{C}_k - 1/2$ would be nonzero and would have the same sign as $C_k - 1/2$ for every k. It would therefore have $n - 1$ sign changes. This contradicts the bound of at most d sign changes whenever $n - 1 > d$. Hence no width d diagonal selective exact-flow SSM can approximate the target uniformly with error strictly less than $1/2$ when $n \geq d + 2$.

This proves

$$W_\infty(\widehat{\mu}_{n,1}, \mu_{n,1}) \geq \frac{1}{2}$$

for width d real diagonal selective exact-flow SSMs of the stated form whenever $n \geq d + 2$.

D G-SLiCE as an augmented path-space flow

Let $G_\theta : \mathcal{X}(d_X) \to \mathcal{Y}(d_y)$ be a generative SLiCE. Introduce an augmented flow-time state

$$Z^{(s)} = (U^{(s)}, Y^{(s)}), \qquad U^{(0)} = X, \qquad Y^{(0)} = 0, \tag{40}$$

and define

$$\widetilde{F}_\theta(s, U, Y) = \begin{pmatrix} 0 \\ G_\theta(U) \end{pmatrix}. \tag{41}$$

Table 7: Ablation on the diagonal vs (diagonal-)dense versions.

Variant	Electricity	Exchange	KDDCup	M4 Hourly	Solar	Traffic	Uber	Wiki2000
$d = 16$, diagonal $b = 1$	**0.0446**	0.0082	0.2871	0.0236	**0.3260**	0.0830	**0.1537**	0.2669
$d = 16$, dense $b = 16$	0.0453	**0.0079**	**0.2866**	**0.0225**	0.3476	**0.0824**	0.1558	**0.2331**
$d = 128$, diagonal $b = 1$	**0.0435**	0.0078	0.2652	**0.0298**	**0.3585**	**0.0799**	0.1504	0.2293
$d = 128$, dense $b = 16$	0.0442	**0.0073**	**0.2593**	0.0319	0.3855	0.0804	**0.1482**	**0.2171**

Then the augmented path-space flow satisfies

$$\frac{d}{ds}\begin{pmatrix} U^{(s)} \\ Y^{(s)} \end{pmatrix} = \begin{pmatrix} 0 \\ G_\theta(U^{(s)}) \end{pmatrix}. \tag{42}$$

Hence $U^{(s)} = X$ for all $s \in [0, 1]$, and

$$Y^{(1)} = G_\theta(X). \tag{43}$$

Thus every direct G-SLiCE generator is the terminal projection of an augmented path-space flow. Under the mild closure assumption that the chosen SLiCE family can ignore auxiliary channels and append zero-output channels, the augmented flow formulation inherits Corollary 2.1.

E Additional results

E.1 Ablation on importance of dense blocks

SLiCEs are more expressive than TSFlow's S4 backbone in two key ways. First, S4 is a non-selective state space model (SSM), meaning that its transition matrices are not state-dependent. Second, S4 only uses diagonal transition matrices with an additive rank-1 correction. To study the impact of these design choices on time series forecasting, we repeat the probabilistic forecasting experiments from Section 4.1 for two hidden dimensions $d \in \{16, 128\}$ and compare a purely diagonal parameterisation ($b = 1$) against a dense block of size $b = 16$. All other hyperparameters are kept fixed as described in Appendix G. Results are reported in Table 7.

Across all configurations, the dense variant outperforms the diagonal one in 9 out of 16 cases. Moreover, for 5 out of 8 datasets, the best-performing configuration uses a dense block (cf. Table 14), indicating that increased expressivity is often beneficial.

To quantify this effect more precisely, we compare the relative improvement of SLiCE over TSFlow-Cond. (OU) based on the results in Table 1, conditioning on whether the dense block is selected. For dataset i, we define the relative improvement in CRPS as

$$r_i = \frac{\text{TSFlow}_i - \text{SLiCE}_i}{|\text{TSFlow}_i|}.$$

We find that, on datasets where SLiCE uses only diagonal blocks, the average relative improvement is 1.75%, whereas it increases to 6.43% on datasets where the dense block is selected. This corresponds to an absolute difference of 4.68 percentage points.

Treating dense block usage as a binary variable, the Pearson correlation between this indicator and the relative improvement is $\rho = 0.21$, indicating a positive association between selecting the dense block and achieving larger gains. The magnitude of this effect should be interpreted with caution given the small number of datasets.

E.2 Additional results for irregular sampling generalisation

Table 8 shows that G-SLiCE remains stable across changes in the sampling irregularity at test time, while TSFlow exhibits substantially larger variance and several degradation regimes, particularly for small k_{train} and under regular-grid evaluation; representative forecasts for $k_{\text{train}} = 1$ are shown in Figure 5a.

Table 8: Cross-irregular generalisation on the ETT 15-minute dataset. Columns indicate the training irregularity k_{train}, and rows indicate the evaluation irregularity k_{test}. We report CRPS as mean $\pm$ std. dev. over random seeds. The final row per model reports evaluation on the regular grid, i.e. $k_{\text{test}} \to \infty$.

		k_{train}					
Model	Evaluation	1	3	10	25	50	100
G-SLiCE	$k_{\text{test}} = 1$	0.13 ± 0.01	0.13 ± 0.01	0.16 ± 0.03	0.14 ± 0.02	0.21 ± 0.15	0.62 ± 0.95
	$k_{\text{test}} = 3$	0.13 ± 0.02	0.12 ± 0.01	0.13 ± 0.02	0.12 ± 0.01	0.14 ± 0.07	0.28 ± 0.28
	$k_{\text{test}} = 10$	0.15 ± 0.03	0.13 ± 0.01	0.13 ± 0.01	0.12 ± 0.01	0.13 ± 0.02	0.15 ± 0.01
	$k_{\text{test}} = 25$	0.15 ± 0.03	0.13 ± 0.01	0.13 ± 0.01	0.12 ± 0.02	0.12 ± 0.01	0.14 ± 0.02
	$k_{\text{test}} = 50$	0.15 ± 0.03	0.13 ± 0.01	0.13 ± 0.01	0.12 ± 0.02	0.13 ± 0.01	0.13 ± 0.02
	$k_{\text{test}} = 100$	0.15 ± 0.03	0.13 ± 0.01	0.13 ± 0.01	0.12 ± 0.01	0.13 ± 0.01	0.13 ± 0.02
	$k_{\text{test}} \to \infty$	0.15 ± 0.03	0.14 ± 0.01	0.14 ± 0.01	0.12 ± 0.01	0.13 ± 0.01	0.13 ± 0.02
TSFlow	$k_{\text{test}} = 1$	0.20 ± 0.07	0.66 ± 1.04	0.15 ± 0.05	0.20 ± 0.10	0.26 ± 0.18	0.19 ± 0.05
	$k_{\text{test}} = 3$	0.17 ± 0.06	0.62 ± 0.93	0.13 ± 0.02	0.15 ± 0.05	0.22 ± 0.14	0.17 ± 0.04
	$k_{\text{test}} = 10$	0.18 ± 0.06	0.37 ± 0.39	0.13 ± 0.03	0.17 ± 0.07	0.23 ± 0.16	0.19 ± 0.05
	$k_{\text{test}} = 25$	0.19 ± 0.07	0.34 ± 0.32	0.14 ± 0.03	0.17 ± 0.07	0.23 ± 0.16	0.19 ± 0.05
	$k_{\text{test}} = 50$	0.19 ± 0.07	0.35 ± 0.35	0.14 ± 0.03	0.17 ± 0.06	0.25 ± 0.18	0.20 ± 0.06
	$k_{\text{test}} = 100$	0.19 ± 0.07	0.36 ± 0.33	0.15 ± 0.04	0.18 ± 0.07	0.27 ± 0.21	0.21 ± 0.06
	$k_{\text{test}} \to \infty$	0.20 ± 0.07	0.40 ± 0.40	0.17 ± 0.06	0.19 ± 0.08	0.29 ± 0.22	0.22 ± 0.06

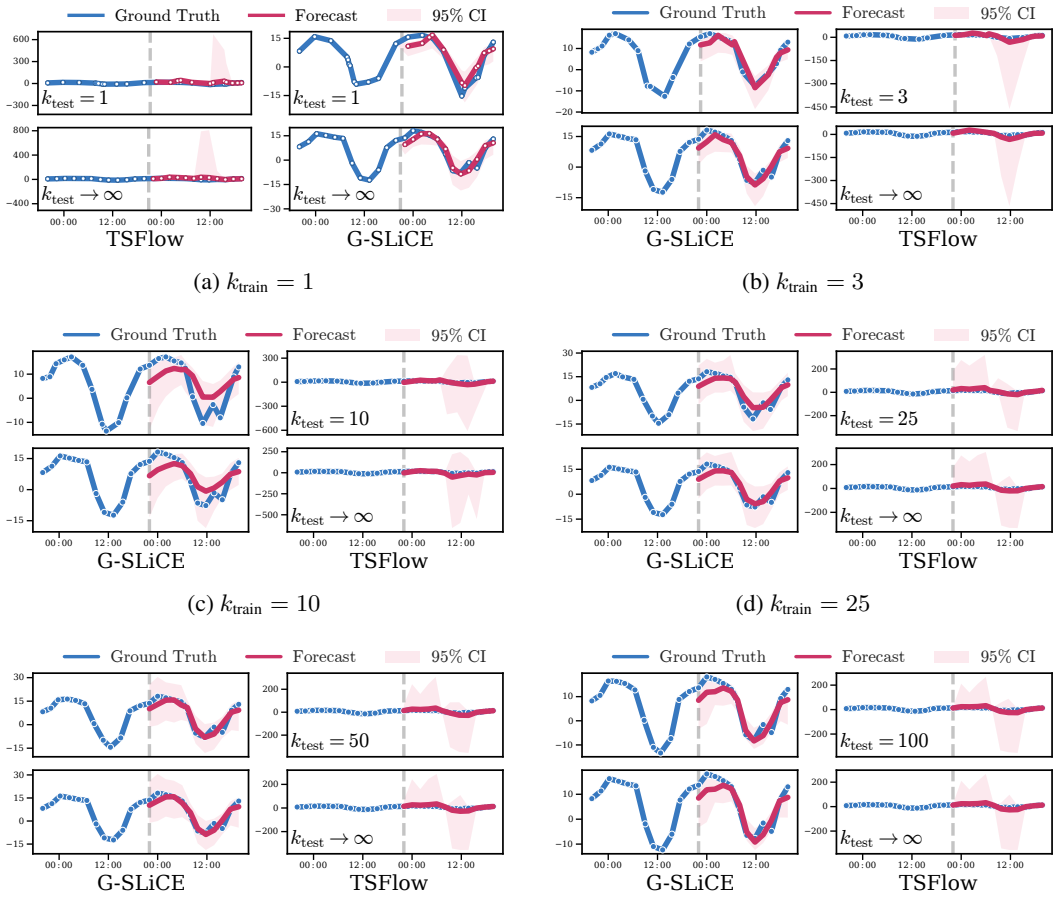
F Computational cost: runtime and memory

Training and generation. Conditional flow-matching training does not require solving the dynamics in Eq. (12) in full. Instead, one flow time is sampled randomly and the vector field is evaluated once per loss evaluation. During generation, a K-step Euler solver requires K vector-field evaluations, giving a rollout cost that scales linearly with K when the remaining settings are fixed. TSFlow shares this repeated-evaluation structure, so the comparison depends on the cost of its S4 backbone relative to the selected SLiCE backbone. Tables 9 and 10 report runtime and peak training-memory measurements, respectively.

Inference. We compare TSFlow with the eight dataset-specific G-SLiCE configurations. For the rollout measurements, we use a batch size of 64, 32 Euler steps, and context and prediction windows of 24 time steps each, on an H100 GPU. G-SLiCE is faster on six of the eight datasets: its rollout latency ranges from 49.17 to 174.06 ms, compared with 171.15 ms for TSFlow. The relative performance ranges from a $3.48\times$ speedup on Solar to a $1.02\times$ slowdown on Wiki2000. Although the more expressive G-SLiCE configurations have higher theoretical complexity, their operations can be accelerated using `torch.compile`, whereas this is generally not available for the TSFlow implementation used in the comparison. The measured latencies therefore reflect both the transition structure and the available implementation optimisations.

Training time. Training cost depends strongly on the selected transition structure. The diagonal G-SLiCE configurations used for Solar and Exchange train $1.67\times$ and $1.45\times$ faster than TSFlow, respectively. More expressive dense-block configurations are slower because training requires backpropagation through structured transition operators, matrix exponentials, and parallel associative scans. The measured slowdown reaches $3.64\times$ on Wiki2000. This slowdown is therefore not intrinsic to every G-SLiCE model, but reflects the cost of additional state-mixing capacity in the evaluated configurations.

Peak training memory. The parallel associative scan increases peak training-memory consumption, particularly for the larger dense-block configurations. Relative to TSFlow, the measured peak-memory ratio ranges from $0.95\times$ on Solar to $16.83\times$ on Wiki2000. The overhead depends on block size, model width and depth, bidirectionality, and sequence length. Solar is the least expensive configuration and uses approximately the same memory as TSFlow. Exchange requires moderately more memory, with a ratio of $1.43\times$. The more expressive Uber and Wiki2000 configurations require substantially more memory, with ratios of $16.52\times$ and $16.83\times$, respectively.



(a) $k_{\text{train}} = 1$ (b) $k_{\text{train}} = 3$

(c) $k_{\text{train}} = 10$ (d) $k_{\text{train}} = 25$

(e) $k_{\text{train}} = 50$ (f) $k_{\text{train}} = 100$

Figure 5: Example forecasts for G-SLiCEs (left) and TSFlow (right) trained with $k_{\text{train}} \in \{1, 3, 10, 25, 50, 100\}$ and evaluated for $k_{\text{test}} = k_{\text{train}}$ (top) or $k_{\text{test}} \to \infty$ (bottom). Curves show ground truth and predictive means; shaded regions indicate 95% confidence intervals.

Expressivity–cost trade-off. These results show that computational efficiency depends on the chosen transition. The diagonal, less expressive G-SLiCE configurations used for Solar and Exchange are faster than TSFlow during both inference and training, with comparable or moderately greater memory consumption. More expressive configurations, such as those used for Uber and Wiki2000, retain inference latency comparable to TSFlow but are slower to train and substantially more memory-intensive. G-SLiCE thus preserves parallel-in-time evaluation within the backbone and often reduces rollout latency, while greater transition expressivity increases training time and memory consumption in these experiments. More efficient structured-scan and matrix-exponential kernels could reduce this overhead.

G Experimental details

G.1 Expressivity Gap: Hard-core Example

We empirically test the hard-core example from Section 2.3. For each sequence length $n \in \{8, 32, 128, 512\}$, we sample

$$(Z_1, \ldots, Z_n) \sim \text{Bernoulli}(p)^{\otimes n}$$

and train each model to approximate the target map C defined in Equation 10. We compare three architectures: G-SLiCE, a diagonal selective SSM, and a dense non-selective SSM. All models are trained with pointwise MSE loss on 1000 training samples, with 200 validation samples and 200 test

Table 9: Rollout latency with 32 Euler steps and reported training time for TSFlow and G-SLiCE. Rollout measurements use an H100 GPU, a batch size of 64, and context and prediction windows of 24 time steps each. Relative speeds describe G-SLiCE compared with TSFlow; lower absolute times are better.

Dataset	Rollout TSFlow (ms)	Rollout G-SLiCE (ms)	Relative rollout speed	Train TSFlow (s)	Train G-SLiCE (s)	Relative training speed
Electricity	171.15	117.97	1.45× faster	2.22	4.98	2.24× slower
Exchange	171.15	67.72	2.53× faster	2.37	1.63	1.45× faster
KDD	171.15	115.98	1.48× faster	2.55	8.00	3.14× slower
M4-Hourly	171.15	136.77	1.25× faster	2.08	4.18	2.01× slower
Solar	171.15	49.17	3.48× faster	2.25	1.35	1.67× faster
Traffic	171.15	83.88	2.04× faster	2.21	2.46	1.11× slower
Uber	171.15	173.55	1.01× slower	2.18	6.55	3.00× slower
Wiki2000	171.15	174.06	1.02× slower	2.14	7.78	3.64× slower

Table 10: Observed peak GPU memory consumption during training. The peak ratio is G-SLiCE memory divided by TSFlow memory; a value below one indicates lower memory consumption for G-SLiCE.

Dataset	TSFlow peak GPU memory (MiB)	G-SLiCE peak GPU memory (MiB)	Peak ratio
Electricity	596.9	6073.6	10.17×
Exchange	730.6	1045.9	1.43×
KDD	1123.4	12314.9	10.96×
M4-Hourly	1112.4	8048.2	7.23×
Solar	596.9	567.9	0.95×
Traffic	596.9	3081.1	5.16×
Uber	596.9	9863.0	16.52×
Wiki2000	730.0	12287.8	16.83×

samples. We train for 50 epochs using Adam optimizer, learning rate 10^{-2}, batch size 256, and no weight decay. Results are averaged over 3 random seeds. G-SLiCE uses hidden dimension $d = 2$, while the diagonal selective SSM and dense non-selective SSM use hidden dimension $d = 8$.

Table 11 reports two metrics. Exact accuracy is the fraction of test sequences for which the model output matches the full target sequence C. The validity ratio is the fraction of generated sequences lying in $\mathcal{H}_n$, i.e. satisfying the no-consecutive-ones constraint. The results agree with the theoretical separation: G-SLiCE learns the hard-core map at all tested sequence lengths, whereas the diagonal selective SSM and dense non-selective SSM fail to recover the exact map beyond the shortest cases. The diagonal selective SSM can still produce valid sequences, but this validity is often degenerate (predicts all-zero sequences) rather than target-tracking. The dense non-selective SSM fits the shortest setting but deteriorates rapidly as n grows.

For the visual diagnostics, we train the models on sequences of length 128. We then evaluate the learned maps on two representative length-8 binary prefixes,

$$Z^{(1)} = (1, 1, 1, 1, 1, 1, 1, 0), \qquad Z^{(2)} = (1, 1, 1, 1, 0, 1, 0, 1),$$

and compare the corresponding outputs with the hard-core target. We also estimate the pushforward distributions by drawing 10^6 Bernoulli input sequences of length 128 and plotting the empirical distributions of their cumulative sums. Figure 2 shows the dense non-selective SSM comparison used in the main text, while Figures 6 and 7 provide the corresponding comparisons for both the dense non-selective SSM and the diagonal selective SSM. Here, G-SLiCEs uses as hidden dimension of $d = 2$, while the diagonal selective and dense non-selective SSM use $d = 4$.

G.2 Time Series Forecasting: Datasets

For the conditional and unconditional time series forecasting experiments, we use eight datasets from the GluonTS benchmark collection Alexandrov et al. [5, 6], following the standard train/validation/test splits provided with each dataset.

Table 11: Validity ratio and exact accuracy on the hard-core target map for the diagonal selective SSM, dense non-selective SSM, and G-SLiCE, evaluated at sequence lengths $n \in \{8, 32, 128, 512\}$. The validity ratio is the fraction of generated sequences in $\mathcal{H}_n$; exact accuracy is the fraction of sequences matching the full target map C.

	Validity ratio				Exact accuracy			
Model	8	32	128	512	8	32	128	512
Diag. selective	0.77	0.91	0.61	0.67	0.02	0.00	0.00	0.00
Dense non-selective	1.00	0.75	0.29	0.00	1.00	0.00	0.00	0.00
G-SLiCEs	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

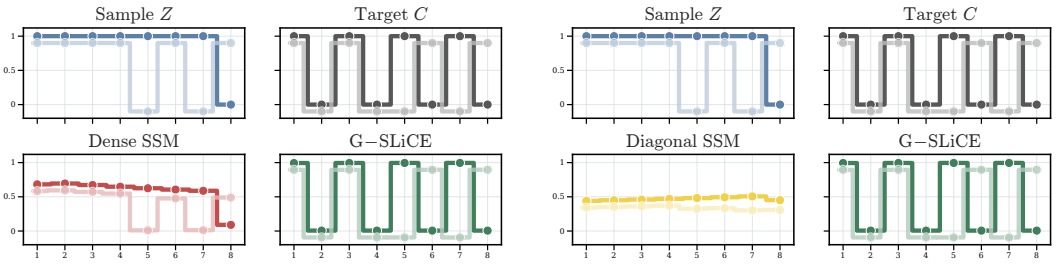


Figure 6: Sample-path diagnostics for the hard-core expressivity example. Each block compares a representative Bernoulli input sequence Z, the corresponding hard-core target C, the output of a restricted SSM baseline, and the output of G-SLiCE. Left: dense non-selective SSM. Right: diagonal selective SSM. Both restricted SSMs fail to reproduce the state-dependent target map, while G-SLiCE tracks C on the displayed examples.

The datasets cover a range of value domains, including positive real-valued series ($\mathbb{R}^+$), series constrained to the open interval $(0, 1)$, and count-valued series ($\mathbb{N}$). They are observed at three different sampling frequencies: 15 minutes (15min), hourly (H), and daily (D). For hourly datasets, we consider forecasting horizons of 24 or 48 hours, while for daily datasets we forecast 30 days ahead. An overview of all datasets and their main characteristics is given in Table 12.

In the experiments of Sections 4.1 and 4.2, we use the Electricity [19], Exchange [39], KDDCup [30], M4 [43], Solar [39], Traffic [30], UberTLC [23], and Wikipedia [27] datasets. For the *grid generalisation* (§ 4.3) experiments, we additionally include the small ETT datasets at 15-minute and 1-hour resolutions [72].

G.3 Training protocol and hyperparameters

Data processing. We use the preprocessing protocol of Kollovieh et al. [38] for all forecasting experiments. Each dataset consists of long univariate or multivariate time series. Training instances are obtained by extracting windows containing a context region and a prediction region. For a dataset with prediction length Δ_{pred}, the prediction region has length Δ_{pred}, as reported in Table 12. The context region has the same length.

For conditional forecasting, the observed context values are used to define a conditional Gaussian-process prior. Concretely, for each training instance, we condition the Gaussian process on the context region and sample a path on the full context-plus-prediction grid. This sampled path is used as the stochastic input to the flow model, while the true future values in the prediction region define the target trajectory for conditional flow matching.

The model input is augmented with deterministic feature channels. First, we append a binary observation mask indicating which grid points correspond to observed context values and which grid points are sampled or forecasted. Second, when lag features are enabled, we append historical values from a longer context window. For hourly datasets, the lagged features at time t are the values from $1, 2, \ldots, 7, 14, 21, 28$ days before t. For daily datasets, the lagged features are the values

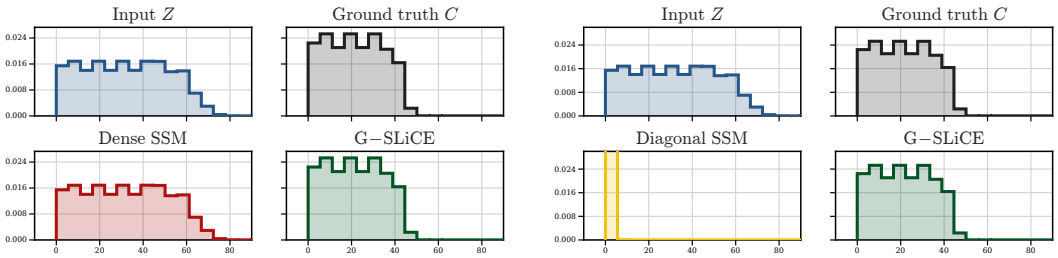


Figure 7: Empirical pushforward distributions for the hard-core expressivity example. Each panel shows the distribution of cumulative sums over 10^6 generated sequences of length 128. Left: dense non-selective SSM compared with the input law, ground-truth hard-core law, and G-SLiCE. Right: the same comparison for the diagonal selective SSM. G-SLiCE matches the ground-truth pushforward distribution, whereas the restricted SSM baselines produce distorted or degenerate distributions.

Table 12: Overview of the GluonTS datasets and their statistics used in the experiments in Section 4.

Dataset	Train Size	Test Size	Domain	Freq.	Median Seq. Length	Prediction Length
Electricity[a] [19]	370	2590	$\mathbb{R}^+$	H	5833	24
Exchange[b] [39]	8	40	$\mathbb{R}^+$	D	6071	30
KDDCup[c] [30]	270	270	$\mathbb{N}$	H	10850	48
M4 (H)[d] [43]	414	414	$\mathbb{N}$	H	960	48
Solar[e] [39]	137	959	$\mathbb{R}^+$	H	7009	24
Traffic[f] [30]	963	6741	(0,1)	H	4001	24
UberTLC[g] [23]	262	262	$\mathbb{N}$	H	4320	24
Wikipedia[h] [27]	2000	10000	$\mathbb{N}$	D	792	30
ETT Small (15min)[i] [72]	14	14	$\mathbb{R}$	15min	69656	96
ETT Small (H)[i] [72]	14	14	$\mathbb{R}$	H	17396	48

[a] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[b] https://github.com/laiguokun/multivariate-time-series-data
[c] https://zenodo.org/record/4656756
[d] https://github.com/Mcompetitions/M4-methods/tree/master/Dataset
[e] https://www.nrel.gov/grid/solar-power-data.html
[f] https://zenodo.org/record/4656132
[g] https://github.com/fivethirtyeight/uber-tlc-foil-response
[h] https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets
[i] https://github.com/zhouhaoyi/ETDataset

from $1, 2, \ldots, 7$ days before t. These lag channels provide the model with short-term and seasonal historical information without changing the prediction target.

All deterministic augmentations are concatenated channel-wise to the sampled Gaussian-process path before being passed to the vector-field network. The same preprocessing, context construction, Gaussian-process conditioning, masking, lag-feature construction, and train/validation/test splits are used for G-SLiCE and the TSFlow baselines. Additionally, we append the analytical mean of the fitted GP to the features.

Model/Training parameters. Conditional forecasting models are trained for 400 epochs, while unconditional generation models are trained for 1000 epochs. Each epoch consists of 128 batches with batch size 64. We use the Adam optimiser, clip gradients at norm 0.5, and maintain an exponential moving average of the model parameters with decay rate 0.999.

For conditional experiments, the Gaussian-process prior uses an Ornstein–Uhlenbeck kernel

$$k_{\mathrm{OU}}(t, t') = \exp\left(-\frac{|t - t'|}{\ell}\right),$$

with length scale $\ell = 1$. The Gaussian process is conditioned on the observed context window and sampled on the model input grid. The sampled trajectories are then interpolated and augmented with the deterministic features used by the backbone.

Table 13: Hyperparameter search space for G-SLiCE.

Hyperparameter	Range
Bidirectional	{True, False}
Use lags	{True, False}
Learning rate	$\{0.0001, 0.001\}$
No. of residual blocks	$\{3, 5\}$
Hidden dim	$\{16, 64, 128\}$
Block size	$\{1, 16\}$

Table 14: Dataset-specific G-SLiCE hyperparameters selected by validation CRPS.

Hyperparameter	Electricity	Exchange	KDDCup	M4 Hourly	Solar	Traffic	Uber	Wiki2000
Bidirectional	True	True	False	True	True	True	True	True
Use lags	True	True	True	False	True	True	True	False
Learning rate	0.001	0.0001	0.001	0.001	0.0001	0.001	0.0001	0.0001
No. of residual blocks	5	3	3	5	3	5	5	5
Hidden dim	128	128	128	16	16	64	128	128
Block size	1	16	16	16	1	1	16	16

For G-SLiCE, we tune only the architecture-specific hyperparameters and learning rate. The remaining training protocol is kept fixed across datasets and experiments to isolate the effect of replacing the S4-style backbone by a SLiCE backbone. We perform a grid search over bidirectionality, use of lag features, learning rate, number of residual SLiCE blocks, hidden dimension, and dense-block size. The search ranges are shown in Table 13. For each dataset, the best configuration is selected by validation CRPS. The selected hyperparameters are reported in Table 14 and are used for the corresponding forecasting, unconditional generation, grid-generalisation, and irregular-sampling experiments.

G.4 Metrics

Wasserstein distance. The *Wasserstein Distance* between two probability measures μ and ν is a metric that measures the minimum cost of transforming one distribution into another. Given two probability measures μ and ν on the metric space $\mathcal{X}$ and a distance metric $d(x, y)$ between two points $x, y \in \mathcal{X}$, a *coupling* γ of μ and ν is a joint probability measure on $\mathcal{X} \times \mathcal{X}$ whose marginals recover μ and ν, i.e. $\gamma(A \times \mathcal{X}) = \mu(A)$ and $\gamma(\mathcal{X} \times B) = \nu(B)$ for all measurable $A, B \subseteq \mathcal{X}$. Denoting the set of all such couplings $\Gamma(\mu, \nu)$, the Wasserstein distance is defined as

$$W_p(\mu, \nu) = \left(\inf_{\gamma \in \Gamma(\mu,\nu)} \int_{\mathcal{X} \times \mathcal{X}} d(x, y)^p \, d\gamma(x, y) \right)^{1/p}.$$

In our experiments we compute the 2-Wasserstein distance for which we use the implementation provided in [24, 25]. To approximate the optimal transport plan γ^* we use 10^7 iterations.

CRPS. The *Continuous Ranked Probability Score (CRPS)* [29] is a proper scoring rule for evaluating probabilistic forecasts of a real-valued target. Let F denote the cumulative distribution function (CDF) of a predictive distribution and let $y \in \mathbb{R}$ be the observed outcome. The CRPS is defined as

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} \big(F(z) - \mathbb{1}\{z \geq y\}\big)^2 \, dz,$$

where $\mathbb{1}\{\cdot\}$ denotes the indicator function. This can be interpreted as the squared L^2 distance between the predictive CDF F and the CDF of a point mass at y. The CRPS is a strictly proper scoring rule, meaning that it is minimised in expectation if and only if F coincides with the true data-generating distribution. In contrast to point-wise losses, it evaluates the full predictive distribution and therefore captures both accuracy and calibration.

Using the GluonTS library Alexandrov et al. [5, 6], we approximate the CRPS via a quantile-based representation. For each forecast horizon step t and each quantile level $q \in \mathcal{Q} = \{0.1, \ldots, 0.9\}$, we compute the pinball (quantile) loss

$$\rho_q(y_t, \hat{y}_{t,q}) = \max\big(q(y_t - \hat{y}_{t,q}),\ (q-1)(y_t - \hat{y}_{t,q})\big).$$

GluonTS aggregates this loss over the forecast horizon and normalises by the total absolute target mass:

$$\mathrm{wQL}_q = \frac{2 \sum_t \rho_q(y_t, \hat{y}_{t,q})}{\sum_t |y_t|},$$

The CRPS is then approximated by averaging over the quantile grid:

$$\mathrm{CRPS} \approx \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \mathrm{wQL}_q.$$

In our experiments, the predictive distribution is represented by 100 samples from the model, from which the empirical quantiles $\hat{y}_{t,q}$ are estimated.

LPS. The *Linear predictive Score*, as introduced by [37], aims to measure the consistency between generated synthetic and real samples. To calculate it, we fit a ridge regression model on 10,000 synthetically generated samples to predict vectors of future values $y_f \in \mathbb{R}^{L_f}$ from a past sequence $y_p \in \mathbb{R}^{L_p}$. The LPS is the CRPS of this model on a test set of real samples. For the linear ridge regression model, we use the implementation provided by [48] .

G.5 Construction of irregular sampling grid

In this Section, we outline how we irregularly subsample a regular base grid $\mathcal{G} = \{0, \delta, 2\delta, \ldots, T\}$, on $[0, T]$. We first sample a continuous irregular grid using the Gamma renewal process. For a desired number of observed points N, we draw $N - 1$ independent increments

$$\Delta\tau_i \sim \mathrm{Gamma}(k, \theta), \qquad i = 1, \ldots, N-1,$$

and normalise them to span the full window:

$$\widetilde{\Delta\tau}_i = \frac{\Delta\tau_i}{\sum_{j=1}^{N-1} \Delta\tau_j} T, \qquad \tau_i = \sum_{j=1}^{i} \widetilde{\Delta\tau}_j,$$

with $\tau_0 = 0$. This gives continuous target times

$$0 = \tau_0 < \tau_1 < \cdots < \tau_{N-1} = T.$$

Since the data are only observed on the base grid $\mathcal{G}$, we map each continuous target time to its nearest base-grid index:

$$m_i = \mathrm{round}\left(\frac{\tau_i}{\delta}\right), \qquad t_i = m_i \delta.$$

We then remove duplicate indices introduced by rounding and, if necessary, resample the Gamma increments until exactly N distinct grid indices are obtained:

$$0 \le m_0 < m_1 < \cdots < m_{N-1} \le M.$$

The final irregular observation grid is therefore

$$\mathcal{G}_{\mathrm{irr}} = \{m_0\delta, m_1\delta, \ldots, m_{N-1}\delta\} \subseteq \mathcal{G}.$$

This construction preserves the irregular spacing induced by the Gamma renewal process while ensuring that every selected observation corresponds to an actual timestamp in the ETTSmall15min data.

The shape parameter k again controls the regularity of the selected grid: $k = 1$ produces bursty, highly irregular subsets, while larger k yields subsets closer to uniform subsampling of the base grid.

G.6 Compute resources

All experiments were run on a compute cluster with NVIDIA A100 (80 GB) and H200 (141 GB) GPUs. Each node was equipped with a 64-core AMD EPYC 9334 CPU clocked at 3.90 GHz and 256 GB of RAM.

G.7 Statistical tests

Because Biloš et al. [11] has a missing entry, it was excluded from the Friedman test. On the remaining models, the Friedman test indicated significant differences in CRPS (statistic of 54.785, p-value of 1.977×10^{-7}). G-SLiCEs achieved the best average rank (1.812), followed by TSFlow (2.312). For the pairwise comparison between G-SLiCE and TSFlow, the one-sided Wilcoxon signed-rank test on log-ratios gave a statistic of 6.0 and p-value 0.055. When Wiki2000 was excluded from this Wilcoxon calculation, the test gave a statistic of 0 and a p-value 0.008.