Reservoir Computing for Temporal Sequence Learning: Overcoming the Hyperparameter Tuning Bottleneck via Free-Probability Kernels

Sara Malacarne[1], Andrea Ceni[2], Claudio Gallicchio[2]

[1]Telenor Research & Innovation, Oslo (Norway)
[2]Department of Computer Science, University of Pisa, Pisa (Italy)

Abstract. Reservoir computing is an attractive paradigm for temporal sequence learning: the recurrent dynamics are kept fixed and only a lightweight readout is trained, making the approach both efficient and broadly applicable. A key practical bottleneck, however, is hyperparameter selection, which typically requires instantiating and evaluating many finite-width reservoirs across a candidate grid — a cost that grows with grid size and reservoir width. We address this bottleneck with a zero-rollout selector grounded in free-probability theory. Our approach targets reservoirs with linear recurrent dynamics and a coordinate-wise nonlinear readout — a setting that preserves the universal approximation power of echo state networks and has recently been shown empirically competitive with fully nonlinear recurrences. At large reservoir width, the post-nonlinearity kernel governing temporal feature encoding converges to a deterministic limit, computable analytically from free-probability moments of the recurrent matrix and a short labelled pilot sequence. Candidate hyperparameter configurations are ranked by kernel ridge regression on this pilot data, with no reservoir instantiated or rolled out during search; the selected configuration transfers across reservoir widths without rerunning selection. Across ten temporal benchmarks spanning nonlinear sequence processing, memory tasks, and chaotic time series forecasting, corrected pairwise tests detect no significant difference between the proposed selector and simulation-based competitors requiring up to 194 400 reservoir rollouts. A screening protocol reduces the rollout budget to 3.9% of full-grid search at nearly identical performance. On a real multivariate telco traffic-forecasting task, the zero-rollout selector matches simulation-based competitors.

Keywords: Time-series · Reservoir computing · Echo state networks · Free probability · Kernel methods · Hyperparameter selection

1 Introduction

Temporal sequence learning often requires models that can retain task-relevant information over multiple time scales while remaining efficient to train and tune. In these settings, performance depends not only on the learning algorithm but

also on the ability to select hyperparameters that control memory, stability, and sensitivity to recent versus distant inputs. This selection problem is especially important for recurrent temporal models, where a configuration that is effective on one task or time scale may be suboptimal on another. Reservoir computing (RC) [13,18] provides an efficient approach to temporal sequence learning: a fixed recurrent dynamical system maps the input history into a high-dimensional temporal representation, and only a lightweight readout is trained. The canonical instance is the Echo State Network (ESN) [10]. Because recurrent weights are not trained, RC is computationally attractive for time-series tasks; however, its performance depends strongly on hyperparameters such as the recurrent gain, input scale, and leakage rate. In standard practice, these hyperparameters are selected by repeatedly instantiating finite-width reservoirs and evaluating them on validation sequences. This can require thousands of reservoir rollouts and must often be repeated when the reservoir width, task, or operating regime changes.

We address this temporal model-selection bottleneck with an offline, zero-rollout procedure. Given a short labelled input-output pilot sequence from the downstream temporal task, the proposed selector ranks candidate operating points by deterministic computation, without instantiating or rolling out finite-width reservoirs during search. The pilot data describe the task; they are not measurements collected from candidate reservoir configurations. The method can be used either as a fully zero-rollout selector, when no simulation or calibration budget is available, or as a pre-screen that reduces a large search grid to a short list before task-direct validation. The key idea is to replace empirical finite-reservoir evaluation with a deterministic large-width kernel. For reservoirs with leaky linear recurrence and a coordinate-wise nonlinear readout, the post-nonlinearity Gram matrix over time steps converges to a deterministic limit computable from free-probability (FP) moments of the recurrent matrix. This kernel captures how temporal information injected at different lags remains correlated after propagation through the reservoir. Hyperparameter selection can therefore be carried out directly on pilot-time kernel matrices, at a cost independent of deployment width.

Contributions.

1. We derive a deterministic large-width kernel for reservoirs with leaky linear recurrence and nonlinear readouts, using FP moments of the recurrent matrix.
2. We use this kernel for task-informed, simulation-free hyperparameter selection on labelled temporal pilot data. Candidate configurations are ranked without finite-reservoir rollouts, and the selected operating point can be reused across deployment widths.
3. We evaluate the selector on a broad temporal benchmark suite covering nonlinear sequence processing, memory, chaotic forecasting, and real multivariate traffic forecasting. The results show that zero-rollout selection remains competitive with simulation-based search, while FP-based screening recovers nearly full-grid performance with a small fraction of the rollout budget.

2 Background

RC [13,18] is a recurrent neural network paradigm in which only a lightweight output layer is trained, while the recurrent dynamics are fixed. The canonical software instance is the Echo State Network (ESN) [10]: a large random recurrent matrix W_r drives a reservoir state $x_t \in \mathbb{R}^n$, and a linear readout $\hat{y}_t = W_{\text{out}}^\top x_t$ is fitted by ridge regression. The echo-state property, i.e., the stable dependence of the reservoir state x_t on the input history rather than on initial conditions, is the key requirement for learning. In practice, whether it holds, and how well the reservoir encodes task-relevant structure, depends on three hyperparameters: the recurrent gain σ_r, the input scale σ_{in}, and the leakage rate α. These are typically chosen either by task-based search – grid or random search [1], Bayesian optimisation [14], gradient-based [19] or evolutionary [15] methods – in which assessing each candidate requires instantiating a finite reservoir, generating its trajectory and fitting a readout; or by dynamical proxies such as echo-state-property and spectral-radius conditions [20,12] and memory capacity [9], which are informative about stability and memory but are not designed to rank candidates against a labelled downstream objective. Our Direct$_n$ and Memory$_n$ baselines represent these two families.

We study the leaky linear reservoir with a coordinate-wise nonlinear readout:

$$x_{t+1} = Ax_t + \alpha W_{\text{in}} u_t, \quad A = (1-\alpha)I + \alpha W_r, \quad z_t = \psi(x_t), \quad \hat{y}_t = W_{\text{out}}^\top z_t. \tag{1}$$

Placing the nonlinearity ψ in the readout rather than the recurrence mirrors the design of linear state-space sequence models such as S4 and LRU [6,17], and preserves universal approximation power [5]. Recent empirical work confirms that this architecture is competitive with fully nonlinear ESNs across a broad range of tasks [11]. The linear recurrence is also what makes the theory tractable: with ψ outside the loop, the reservoir state x_t is a linear function of past inputs and of W_r, so the large-width state covariance reduces to normalised traces of polynomials in W_r and $W_r^\top$.

This is precisely the quantity that FP [16] handles. Because the recurrence is linear, the reservoir state can be written as a weighted sum of past inputs propagated through powers of $A = (1-\alpha)I + \alpha W_r$. The covariance between states at two time steps is therefore determined by coefficients of the form $\frac{1}{n}\operatorname{Tr}\big(A^k (A^\ell)^\top\big)$, which measure how information injected at lags k and ℓ remains correlated after passing through the reservoir. At large width, these coefficients converge to deterministic values. FP provides a way to compute them directly from the chosen recurrent-matrix model, without instantiating a finite reservoir. For Ginibre matrices, the coefficients are explicit; the same construction also applies to Haar-orthogonal, cyclic-shift, Gaussian skew-symmetric, and circulant matrices, with matrix-specific coefficients where required. The post-nonlinearity reservoir kernel – the Gram matrix of readout features across time steps – can then be evaluated directly from the input sequence and the hyperparameters $(\sigma_r, \sigma_{\text{in}}, \alpha)$, with no reservoir instantiated.

Deterministic large-width descriptions of this kind are established: infinite-width ESNs as recurrent kernel machines [8], random-matrix analyses of large

linear ESNs [2], nonlinear recurrent kernels used directly for prediction [3], and universal Volterra reservoir kernels [4]. That work uses the kernel to characterise the reservoir or to predict with it; it does not ask whether such a kernel can *replace finite-reservoir rollouts during hyperparameter search*. That is the step taken here, and to our knowledge this use of the deterministic post-nonlinearity kernel for task-informed selection is new.

3 Free-Probability Kernel for Temporal Model Selection

Sections 3.1 and 3.2 are expository and fix notation; the new results begin in Section 3.3, with Theorem 1, Proposition 1, and Lemma 1.

3.1 Leaky linear reservoir

Throughout the paper we instantiate the reservoir model (1) with a scaled real Ginibre recurrent matrix W_r and an independent input matrix W_{in} with i.i.d. Gaussian entries:

$$W_r = \frac{\sigma_r}{\sqrt{n}} G_r, \quad W_{\text{in}} = \frac{\sigma_{\text{in}}}{\sqrt{d}} G_{\text{in}}, \quad (G_r)_{ij}, (G_{\text{in}})_{ij} \sim \mathcal{N}(0,1), \quad G_r \perp G_{\text{in}}. \tag{2}$$

The $1/\sqrt{n}$ scaling keeps W_r's spectrum $O(1)$; the $1/\sqrt{d}$ scaling keeps each coordinate of $W_{\text{in}} u_t$ at variance $O(1)$ when $\|u_t\|^2 = O(d)$. The kernel construction below requires two properties of the recurrent matrix: asymptotic boundedness in operator norm and coordinate-level diagonal self-averaging of $A_n^k (A_n^\ell)^\top$. Scaled Ginibre matrices satisfy these properties with probability tending to one. Haar-orthogonal and cyclic-shift recurrence satisfy the corresponding finite-width norm bound deterministically and yield the same coefficients (6). Other structured recurrent matrices can be handled when the same assumptions hold, with matrix-specific coefficients where required. The experiments use Ginibre recurrence throughout. For Ginibre W_r, the large-width eigenvalue cloud of A is a disk centred at $1-\alpha$ with radius $\alpha\sigma_r$. Hence, for $\alpha > 0$, the asymptotic linear echo-state condition is $\sigma_r < 1$.

3.2 Readout and kernel notation

In (1), the matrices W_r and W_{in} are drawn once as in (2) and held fixed; only W_{out} is fitted. The readout features are $z_t = \psi(x_t)$, with $\psi : \mathbb{R} \to \mathbb{R}$ applied coordinate-wise; experiments use $\psi = \tanh$, and the theory covers any bounded C^1 activation with bounded derivative. Stacking the training and validation feature vectors gives matrices $Z_T \in \mathbb{R}^{T_{\text{train}} \times n}$ and $Z_V \in \mathbb{R}^{T_{\text{val}} \times n}$. The width-normalised kernel blocks are

$$K_{\psi,n,TT} = \tfrac{1}{n} Z_T Z_T^\top \in \mathbb{R}^{T_{\text{train}} \times T_{\text{train}}}, \qquad K_{\psi,n,VT} = \tfrac{1}{n} Z_V Z_T^\top \in \mathbb{R}^{T_{\text{val}} \times T_{\text{train}}},$$

both indexed by pilot time steps, not by reservoir coordinates. Ridge regression admits the kernel form [7]:

$$\widehat{Y}_V = K_{\psi,n,VT}(K_{\psi,n,TT} + \lambda I)^{-1} Y_T,$$

so hyperparameter selection requires only these pilot-time matrices — which the FP theory replaces with deterministic large-width limits computable without instantiating a reservoir.

3.3 Free-probabilistic kernel construction

For a fixed context length $L < \infty$, the truncated reservoir state is

$$x_t^{(L)} := \alpha \sum_{k=0}^{L} A^k W_{\text{in}} u_{t-1-k}, \tag{3}$$

and the empirical nonlinear kernel entry at width n is

$$\widehat{K}_{\psi,n}^{(L)}(t,s) := \frac{1}{n} \psi(x_t^{(L)})^\top \psi(x_s^{(L)}). \tag{4}$$

Our main result shows that this finite-reservoir quantity converges to a deterministic large-width limit.

Theorem 1 (Deterministic nonlinear reservoir kernel). *Let $\psi \in C_b^1(\mathbb{R})$. For fixed L, t, s and deterministic bounded input sequence $(u_r)_r$, as $n \to \infty$,*

$$\widehat{K}_{\psi,n}^{(L)}(t,s) \xrightarrow{P} K_\psi^{(L)}(t,s) := \mathbb{E}[\psi(g_t)\psi(g_s)]\,, \tag{5}$$

where $(g_t, g_s) \sim \mathcal{N}\big(0, \Sigma^{(L)}(t,s)\big)$ and $\Sigma^{(L)}(t,s)$ is the 2×2 covariance matrix defined in (10)*. The convergence is in probability with respect to the joint randomness of W_r and W_{in}.*

The limit is a Gaussian expectation, fully determined by $\Sigma^{(L)}(t,s)$. We now show how to compute it. Because the recurrence is linear, the covariance between states at time steps t and s depends on how strongly inputs injected at different lags remain correlated after propagation through powers of A. This is captured by the recurrent memory coefficients

$$\tau_{k,\ell} := \sum_{j=0}^{\min(k,\ell)} \binom{k}{j}\binom{\ell}{j} (1-\alpha)^{k+\ell-2j} (\alpha\sigma_r)^{2j}, \tag{6}$$

which, for Ginibre recurrence, are the large-width limits of the mixed normalised traces $\frac{1}{n}\operatorname{Tr}\big(A^k (A^\ell)^\top\big)$, as stated below.

Proposition 1 (Ginibre recurrent memory coefficients). *Let $A_n = (1-\alpha)I + \alpha W_r^{(n)}$. For every fixed $k, \ell \geq 0$,*

$$\frac{1}{n}\operatorname{Tr}\big(A_n^k (A_n^\ell)^\top\big) \xrightarrow{L^1} \tau_{k,\ell}, \tag{7}$$

where $\tau_{k,\ell}$ is given by (6)*.*

Proof (sketch). Expand $A_n^k(A_n^\ell)^\top$ by the binomial theorem. In the large-width Ginibre $*$-moment limit, only terms containing equal powers of G_r and $G_r^\top$ survive, yielding (6).

Lemma 1 (Diagonal self-averaging). *Let $A_n = (1-\alpha)I + \alpha W_r^{(n)}$. For every fixed $L < \infty$,*

$$\max_{0\le k,\ell\le L} \frac{1}{n}\sum_{i=1}^{n} \left|\left[A_n^k(A_n^\ell)^\top\right]_{ii} - \tau_{k,\ell}\right|^2 \xrightarrow{P} 0. \tag{8}$$

Proof (sketch). Fix k, ℓ. By permutation symmetry of the Ginibre law, the diagonal entries of $A_n^k(A_n^\ell)^\top$ are identically distributed. Proposition 1 identifies their limiting mean through the normalised trace. On events where the operator norm of $W_r^{(n)}$ is bounded, a Gaussian Poincaré estimate gives a vanishing variance bound for each fixed-degree diagonal polynomial. Markov's inequality then yields empirical L^2 convergence for each (k, ℓ). Since L is fixed, the maximum over $0 \le k, \ell \le L$ follows from a finite union bound.

Given the coefficients $\tau_{k,\ell}$, define the deterministic state covariance

$$Q^{(L)}(t,s) := \alpha^2 \frac{\sigma_{\text{in}}^2}{d} \sum_{k,\ell=0}^{L} \tau_{k,\ell}\, u_{t-1-k}^\top u_{s-1-\ell}, \tag{9}$$

and the 2×2 covariance matrix entering Theorem 1,

$$\Sigma^{(L)}(t,s) := \begin{pmatrix} Q^{(L)}(t,t) & Q^{(L)}(t,s) \\ Q^{(L)}(t,s) & Q^{(L)}(s,s) \end{pmatrix}. \tag{10}$$

Proof idea for Theorem 1. Conditionally on A, the coordinate covariance of $(x_{t,i}^{(L)}, x_{s,i}^{(L)})$ is

$$C_{n,i}^{(L)}(t,s) = \alpha^2 \frac{\sigma_{\text{in}}^2}{d} \sum_{k,\ell=0}^{L} \left[A_n^k(A_n^\ell)^\top\right]_{ii} u_{t-1-k}^\top u_{s-1-\ell}.$$

Lemma 1 implies that these coordinate covariances converge in empirical L^2 to the common deterministic limit $Q^{(L)}(t,s)$, and analogously for the diagonal entries. Since $\psi \in C_b^1$, the Gaussian expectations converge to (5). A Gaussian Poincaré estimate controls the remaining fluctuations of the empirical kernel around its conditional mean. The Gaussian expectation in (5) can be evaluated by two-dimensional quadrature. In practice we use the erf surrogate $\tanh(x) \approx \operatorname{erf}(\sqrt{\pi}x/2)$, for which the kernel admits the closed-form arcsin expression

$$K_{\text{erf}}^{(L)}(t,s) = \frac{2}{\pi} \arcsin\left(\frac{\pi Q^{(L)}(t,s)}{\sqrt{\left(2 + \pi Q^{(L)}(t,t)\right)\left(2 + \pi Q^{(L)}(s,s)\right)}} \right). \tag{11}$$

Finite-reservoir deployment uses coordinate-wise tanh features.

3.4 Zero-rollout selection on temporal pilot data

The previous subsection gives the theoretical ingredient needed for selection. For any candidate $\theta = (\sigma_r, \sigma_{\text{in}}, \alpha)$, the deterministic kernel $K_\psi^{(L)}$ replaces the empirical reservoir kernel, requiring only a short labelled pilot sequence from the downstream task.

Algorithm 1: Zero-rollout FP hyperparameter selector

Input: Pilot sequence (u_t, y_t), split into train/validation; context window L; candidate grid Θ; ridge grid Λ

Output: Selected hyperparameters $\theta^\star$ and ranking regulariser $\lambda^\star_{\text{sel}}$

for *each* $\theta = (\sigma_r, \sigma_{\text{in}}, \alpha) \in \Theta$ *satisfying the stability guard* **do**
  Compute $\tau_{k,\ell}(\theta)$ via (6) for $0 \le k, \ell \le L$; Build $Q^{(L)}$ via (9) and the train/validation kernel blocks via (11); **for** *each* $\lambda \in \Lambda$ **do**
    Compute $\text{NMSE}_{\text{val}}(\theta, \lambda)$ by kernel ridge regression;
  end
end
return $(\theta^\star, \lambda^\star_{\text{sel}}) = \arg\min_{\theta,\lambda} \text{NMSE}_{\text{val}}(\theta, \lambda)$;

FP selector. All matrices in Algorithm 1 are indexed by pilot time steps rather than reservoir coordinates: no $n \times n$ recurrent matrix is formed and no finite reservoir is rolled out. After precomputing input inner products, a direct implementation constructs each pilot-time kernel using $O(T_{\text{pilot}}^2 L^2)$ lag-pair operations and $O(T_{\text{pilot}}^2)$ memory. Kernel ridge regression is performed on the $T_{\text{train}} \times T_{\text{train}}$ pilot block. The selection cost therefore depends on the pilot length, context window, candidate grid, and ridge grid, but not on the deployment width n. The regulariser $\lambda^\star_{\text{sel}}$ is used only for ranking; at deployment, λ is re-selected on the deployment training block.

Stability guard. For scaled real Ginibre recurrence, the large-width eigenvalue cloud of $A = (1-\alpha)I + \alpha W_r$ is a disk centred at $1-\alpha$ with radius $\alpha\sigma_r$. Define

$$\rho_{\text{FP}}(\alpha, \sigma_r) := (1-\alpha) + \alpha\sigma_r.$$

The asymptotic stability condition is $\rho_{\text{FP}} < 1$, equivalently $\sigma_r < 1$ for $\alpha > 0$. Since a large-width selector cannot detect realization-specific instabilities of finite Ginibre matrices close to this boundary, we apply the fixed conservative guard

$$\rho_{\text{FP}}(\alpha, \sigma_r) \le 1 - \delta, \qquad \delta = 0.05. \tag{12}$$

The margin is a practical robustness safeguard rather than a claim of optimality, and is kept fixed across deployment widths. The additional margin is not needed for recurrent matrices whose finite-width spectral boundary is controlled exactly. For scaled Haar-orthogonal recurrence, $W_r = \sigma_r O_n$ with $O_n^\top O_n = I$, the eigenvalues of A are $(1-\alpha) + \alpha\sigma_r e^{i\varphi_j}$ at every width, so that $\rho(A) \le \rho_{\text{FP}}$ deterministically. The same observation applies to cyclic-shift recurrence.

Table 1. Synthetic task protocols. Deployment train/test lengths are counted after washout.

Task	Washout	Deploy train	Test	Input, target, and task parameter
NARMA10	100	800	800	$u_t \sim U(0, 0.5)$; NARMA target, order $k = 10$
NARMA20	100	1200	1200	$u_t \sim U(0, 0.5)$; NARMA target, order $k = 20$
NARMA30	100	1200	1200	$u_t \sim U(0, 0.5)$; NARMA target, order $k = 30$
NARMA50	100	1600	1600	$u_t \sim U(0, 0.5)$; NARMA target, order $k = 50$
MC	100	2000	2000	$u_t \sim U(-1, 1)$; delayed targets at lags 1:200
Inubushi	100	1000	1000	$u_t \sim U(-1, 1)$; $y_t = \sin(2u_{t-5})$
Lorenz63	200	2000	2000	scalar Lorenz63 forecast, horizon $h = 25$
SF-MG30	200	2000	2000	Mackey–Glass ($\tau = 30$) forecast, horizon $h = 10$
MG84	200	1000	1000	Mackey–Glass ($\tau = 17$) forecast, horizon $h = 84$
Lorenz96	200	2000	2000	5D Lorenz96 forecast, horizon $h = 25$

4 Experiments

4.1 Setup

Tasks. We evaluate ten sequence benchmarks: four nonlinear temporal-processing tasks (NARMA-k for $k \in \{10, 20, 30, 50\}$), linear memory capacity (MC), one nonlinear delayed-input task (Inubushi), and four chaotic forecasting tasks (Lorenz63, Santa Fe–style Mackey–Glass with $\tau = 30$, Mackey–Glass with horizon $h = 84$, and 5D Lorenz96). Task protocols are summarised in Table 1.

Selectors. **FP**: the proposed zero-rollout selector (Algorithm 1). Candidate points are ranked by kernel NMSE on short task-specific labelled pilot sequences, without instantiating a finite reservoir. We use three independently generated pilot sequences and assign each candidate its largest validation NMSE across the three draws; FP selects the candidate with the smallest worst-case NMSE. **Memory**$_n$: evaluates each grid point at width n on a generic scalar input $u_t \sim U(-1, 1)$. For each candidate θ, a ridge readout is trained to reconstruct $D = 2n$ delayed inputs $u_{t-1}, \ldots, u_{t-D}$ from reservoir features z_t. The selection score follows Jaeger's memory capacity criterion [9]:

$$\mathrm{MemScore}(\theta) = \frac{1}{D} \sum_{d=1}^{D} \max\left(0,\ 1 - \frac{\sum_t (u_{t-d} - \hat{u}_{t-d})^2}{\sum_t (u_{t-d} - \bar{u}_d)^2}\right), \tag{13}$$

where $\hat{u}_{t-d}$ is the ridge prediction of u_{t-d} from z_t and $\bar{u}_d = \sum_t (u_{t-d}/T)$ is its sample mean. Each term is the coefficient of determination for lag d, clipped to zero. Scores are averaged over three independent reservoir seeds. Task-agnostic: the same proxy is used for every downstream task. **Direct**$_n$: evaluates the downstream task at width n and ranks each candidate by its mean validation score across three independent reservoir seeds. **ESN Direct**$_n$: same as Direct$_n$, but using a standard nonlinear ESN recurrence $x_{t+1} = (1-\alpha)x_t + \alpha \tanh(W_r x_t + W_{\mathrm{in}} u_t)$ instead of the linear recurrence.

Grid and deployment. The fine grid contains 6480 candidate points in $(\sigma_r, \sigma_{\mathrm{in}}, \alpha)$. Each empirical selector evaluates every candidate at $n_{\mathrm{select}} = 500$; the reservoir is instantiated three times with independent random seeds per candidate, and

the selection score is the mean validation score across the three realisations. Under the per-task selection accounting used in the matched protocol, this gives 194 400 finite-reservoir selection rollouts per empirical selector across the ten-task suite.[1] Memory is task-agnostic and can in principle be reused across tasks; the 194 400 figure reflects the matched per-task protocol.

FP ranks the same grid using zero finite-reservoir rollouts. For each task, it uses three independently generated pilot sequences of length $T_{\text{pilot}} = 500$, each split into 333 training and 167 validation time steps, with context window $L = 50$ (eq. (3)).

After selection, $\theta^\star$ is fixed and evaluated with fresh deployment seeds. We use three deployment seeds for the intermediate-width trend at $n \in \{1\,000, 3\,000, 5\,000, 10\,000\}$ and 15 deployment seeds at $n = 20000$, where the critical-difference analysis is performed; reported deployment scores are averaged over these seeds. Throughout, we use the holdout-ridge protocol with $\lambda \in \{10^{-6}, 10^{-5}, \ldots, 10^{2}\}$. Performance is reported as $1 - \text{NRMSE}$; higher is better.

4.2 Temporal benchmark results

The synthetic suite evaluates three complementary uses of the FP selector: zero-rollout selection when no simulation budget is available, width-portable deployment as the reservoir grows, and FP-based pre-screening when a small rollout budget can be spent. Direct_{500} attains the highest mean deployment score across the ten-task suite, reaching 0.777 at $n = 20\,000$ (Table 2). This is expected: it directly evaluates finite-reservoir downstream validation performance during selection. FP reaches 0.761 while using zero finite-reservoir search rollouts. The task-level comparison in Figure 1 shows that FP is best or tied-best on NARMA30, NARMA50, SF-MG30, and Lorenz96, and remains close to the best selector on several other tasks. The largest gaps occur on Inubushi, Lorenz63, and MG84, where direct finite-reservoir selection offers a clearer advantage. Memory_{500} is strongest on MC, where its delayed-input reconstruction proxy is closely aligned with the downstream objective. FP also exceeds ESN Direct_{500} on average at $n = 20\,000$ (0.761 versus 0.751), showing that the analytically tractable linear-recurrence setting remains competitive in this suite. Full task-level scores are in Table 6.

Critical-difference analysis. Figure 2 gives the rank-based comparison at $n = 20\,000$. FP has the best mean rank (2.05), followed by ESN Direct_{500} (2.20), Direct_{500} (2.35), and Memory_{500} (3.40). After Holm correction, no pairwise difference is statistically significant at $\alpha = 0.05$. The supported conclusion is that FP remains competitive with simulation-based selection while eliminating finite-reservoir search rollouts.

[1] We also evaluated empirical selection at $n_{\text{select}} = 100$. The selected hyperparameters gave lower deployment performance at larger widths, so we report $n_{\text{select}} = 500$ only.

Table 2. Average deployment score under the matched holdout-ridge protocol across the ten benchmark tasks. Scores are $1 - \mathrm{NRMSE}$; $\pm$ denotes task-level standard deviation. Theory evals and RC sims are selection-stage costs. Best deployment scores are bolded and second-best scores are underlined.

Method	$n = 1\,000 \uparrow$	$n = 3\,000 \uparrow$	$n = 5\,000$	$n = 10\,000 \uparrow$	$n = 20\,000\uparrow$
FP	0.675±0.160	<u>0.727±0.158</u>	<u>0.741±0.157</u>	<u>0.755±0.155</u>	<u>0.761±0.154</u>
Memory_{500}	0.633±0.149	0.677±0.133	0.690±0.138	0.705±0.142	0.714±0.144
Direct_{500}	**0.710±0.175**	**0.751±0.167**	**0.758±0.164**	**0.771±0.159**	**0.777±0.156**
ESN Direct_{500}	<u>0.699±0.196</u>	0.720±0.179	0.725±0.178	0.747±0.167	0.751±0.160

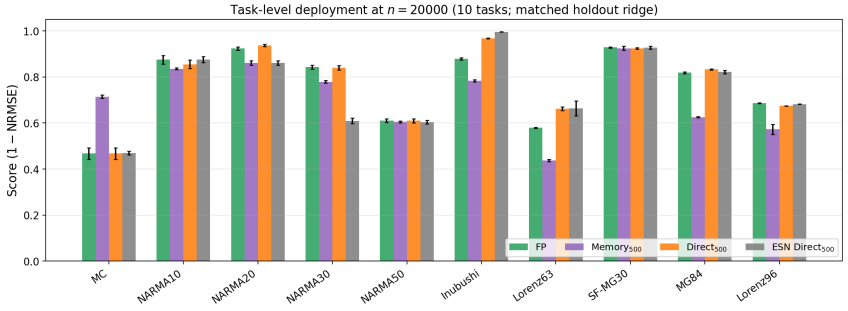

Fig. 1. Deployment score at $n = 20\,000$ across the ten-task synthetic suite. Scores are $1 - \mathrm{NRMSE}$; higher is better.

Computational scaling and width portability. The predictive comparison above is obtained under sharply different selection costs. Figure 3 reports full-grid selection time as the selection width increases. FP constructs pilot-time kernel matrices and therefore remains independent of n_{select}, whereas the empirical selectors require finite-reservoir rollouts and become increasingly expensive with width. At deployment, the same FP operating point is reused without rerunning the selector: its mean score increases from 0.675 at $n = 1000$ to 0.761 at $n =$ 20,000, while the gap to Direct_{500} narrows from 0.035 to 0.016 (Table 2).

FP screening with a rollout budget. The zero-rollout selector can also serve as a pre-screen when a small simulation budget is available. FP first ranks the full grid without finite-reservoir rollouts; Direct_{500} is then evaluated only on the top K candidates. With $K = 250$, the screened method reaches 0.772 versus 0.777 for full-grid Direct_{500} at $n = 20\,000$, using 7 500 rather than 194 400 selection rollouts (3.9% of the full-grid budget). This closes 65.6% of the score gap between pure FP and full-grid Direct_{500}. The screened protocol provides an intermediate operating mode between zero-rollout selection and exhaustive task-direct search, and motivates the FPTop5+Direct_{500} telco experiment below.

Selection-length sensitivity. Table 4 and Figure 4 report an auxiliary coarse-grid sweep over the selection-sequence length. FP is already close to its plateau at $T_{\mathrm{select}} = 250$ and reaches 0.750 by $T_{\mathrm{select}} = 500$, remaining stable as the pilot sequence grows. Direct selection benefits from longer sequences, while Memory_{500}

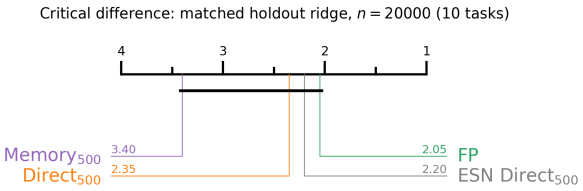

Fig. 2. Average ranks across the ten synthetic tasks at $n = 20\,000$. Lower is better. Horizontal cliques indicate no significant pairwise difference under Wilcoxon signed-rank tests with Holm correction ($\alpha = 0.05$).

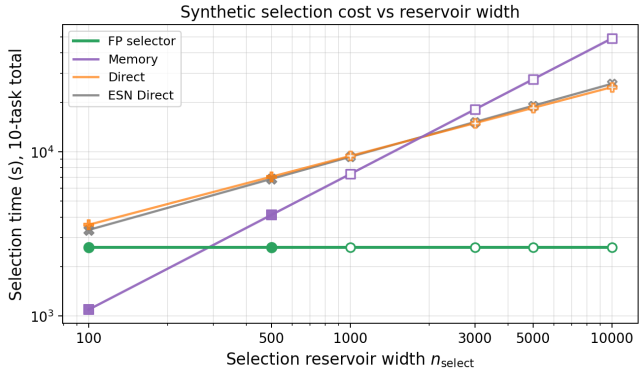

Fig. 3. Full-grid selection time versus selection width n_{select}. Lower is better. FP remains independent of reservoir width because it constructs pilot-time kernel matrices rather than rolling out finite reservoirs. Filled markers denote measured runs; remaining points are power-law extrapolations.

remains unchanged because it uses a task-agnostic proxy sequence. FP is therefore efficient not only in finite-reservoir rollouts but also in the length of the labelled pilot sequence required for selection.

4.3 Real-data traffic-forecasting case study

We evaluate the selector on hourly traffic data from ten randomly sampled sectors of an operational cellular network. The input combines two traffic Key Performance Indicators (KPIs), corresponding to data and voice usage, with calendar covariates encoding hour-of-day and weekday as sine–cosine pairs to capture periodicity; the target is the joint k-step forecast of both KPIs for $k = 1, 6$ and 12. Each sector uses 841 training rows, 360 validation rows, and 932 test rows, with context window $L = 96$ hours. The compact hyperparameter grid contains 48 candidates, so exhaustive empirical selection using 3 seeds requires 144 finite-reservoir rollouts. Because the supervised series is short and the outputs are correlated, we use RidgeCV for readout regularisation in this experiment.

Table 5 reports deployment results at $n = 20\,000$. FPTop5+Direct$_{500}$ and Direct$_{500}$ attain essentially identical average scores (0.601), while the screened

Table 3. FP-screened synthetic sweep at $n = 20\,000$. K: number of top-ranked FP candidates passed to Direct$_{500}$. Budget: percentage of full-grid selection rollouts. Gain recovered: percentage of the score gap from pure FP to full-grid Direct$_{500}$ closed by screening. Scores are task-level mean±std across ten tasks.

Method	K	Budget	Mean score ↑	Gain recovered
FP	–	0.0%	0.761±0.154	0.0%
FP-screened Direct$_{500}$	25	0.4%	0.761±0.156	0.8%
FP-screened Direct$_{500}$	50	0.8%	0.766±0.154	27.0%
FP-screened Direct$_{500}$	100	1.5%	0.765±0.154	24.3%
FP-screened Direct$_{500}$	250	3.9%	0.772±0.159	65.6%
Full-grid Direct$_{500}$	6,480	100.0%	**0.777±0.156**	100.0%

Table 4. Selection-length sensitivity in an auxiliary coarse-grid synthetic sweep. Scores are deployment $1 - \mathrm{NRMSE}$ at $n = 5000$, averaged over tasks; ± denotes task-level standard deviation. The selection length T_{select} is the train/validation sequence length used only during hyperparameter selection. Best entries are bolded and second-best entries are underlined.

T_{select}	FP	Memory$_{500}$ ↑	Direct$_{500}$ ↑	ESN Direct$_{500}$ ↑
150	0.662 ± 0.209	$\mathbf{0.690 \pm 0.146}$	0.629 ± 0.241	0.662 ± 0.270
250	$\mathbf{0.744 \pm 0.194}$	0.690 ± 0.146	0.619 ± 0.245	0.612 ± 0.261
500	$\mathbf{0.750 \pm 0.196}$	0.690 ± 0.146	0.716 ± 0.176	0.687 ± 0.247
1000	0.750 ± 0.195	0.690 ± 0.146	$\mathbf{0.757 \pm 0.211}$	0.710 ± 0.223
2000	0.741 ± 0.197	0.690 ± 0.146	$\mathbf{0.760 \pm 0.200}$	0.734 ± 0.222

variant uses only 15 finite-reservoir selection rollouts instead of 144. Pure FP remains close to the direct methods, reaching 0.586 on average. Memory$_{500}$ provides an informative contrast: its delayed-KPI reconstruction proxy is task-agnostic and has much larger sector-to-sector variation, averaging 0.382 ± 0.507. On the synthetic MC task, delayed-input reconstruction is itself the downstream objective; here it is only a proxy, so generic memory optimisation need not identify a good operating point for multivariate forecasting.

To check whether the aggregate result is driven by the easiest sectors, we stratify post hoc by FPTop5+Direct$_{500}$ score. On the three highest-scoring sectors, FP, Direct$_{500}$, and FPTop5+Direct$_{500}$ are effectively tied at 0.752. On the three lowest-scoring sectors, pure FP attains the highest average score (0.493), ahead of FPTop5+Direct$_{500}$ (0.468) and Direct$_{500}$ (0.449). This split is descriptive rather than inferential.

5 Discussion

What the kernel changes. The large-width kernel relocates selection cost from the reservoir to the pilot sequence. Empirical selectors instantiate a reservoir for every grid point; FP instantiates none. This has two concrete consequences. First, the selection cost is independent of reservoir width: the same $\theta^\star$ computed at pilot length $T_{\mathrm{pilot}} = 500$ deploys at any n, and its score improves as n grows because the selector targets the large-width limit. Second, cost scales with pilot length rather than with reservoir width. Table 4 shows that FP plateaus at

Table 5. Real-data bivariate traffic forecasting at deployment width $n = 20\,000$ on ten randomly sampled sectors. Inputs are hourly data traffic and voice traffic; targets are both KPIs at horizons 1:12. RidgeCV selects the readout regularisation. Score columns report $1-\text{NRMSE}$ mean±std across sectors. The last column reports the mean within-sector deployment-seed std over three seeds.

Method	Sims	Avg.↑	$h = 1$ ↑	$h = 6$ ↑	$h = 12$ ↑	Seed std
FP	0	0.586±0.149	0.599±0.190	0.588±0.140	0.581±0.152	0.003
Memory_{500}	144	0.382±0.507	0.265±0.885	0.373±0.529	0.428±0.374	0.009
Direct_{500}	144	0.601±0.131	**0.663±0.107**	**0.594±0.130**	0.574±0.168	0.005
$\text{FPTop5+Direct}_{500}$	15	**0.601±0.124**	0.649±0.118	0.593±0.127	**0.591±0.131**	0.001

$T_{\text{select}} = 500$, while task-direct selection continues to improve; FP is therefore data-efficient as well as simulation-efficient. The screening protocol combines both properties: FP ranks the full grid offline, and task-direct evaluation is restricted to the top K candidates. At $K = 250$, 3.9% of the rollout budget recovers 65.6% of the full-grid gain.

Recurrent structure as part of the model specification. The theory requires two properties of the recurrent matrix: asymptotic operator-norm control and coordinate level diagonal self-averaging of $\frac{1}{n}\operatorname{Tr}\big(A_n^k(A_n^\ell)^\top\big)$ (Lemma 1). Ginibre matrices satisfy both with probability tending to one; Haar-orthogonal and cyclic-shift matrices satisfy the norm bound deterministically and give the same closed-form coefficients (6), so the selector is unchanged. Moreover, for these matrices the eigenvalues of A satisfy $\rho(A) \le \rho_{\text{FP}}$ deterministically at every finite width, so the conservative stability guard (12) is not required and the full grid can be searched without a margin. Gaussian skew-symmetric and circulant matrices fit the same framework with matrix-specific coefficients. In each case the recurrent-matrix model is specified once and used analytically by the selector.

Scope and extensions. Our experimental variable is the selection procedure, so the deployed model is held fixed across selectors; ESN Direct_{500} is the closest architecture reference, and an external non-reservoir baseline such as a fully trained RNN, which would place the absolute scores in a wider context, is left to future work. The exact theory requires linear recurrent dynamics. For a nonlinear ESN, the state is no longer a linear combination of propagated inputs, and the large-width covariance involves time-varying activation factors that depend on the input trajectory. A frozen-diagonal or mean-field approximation would replace the exact trace moments with approximate ones, at the cost of the exactness guarantee. Extending the framework to nonlinear recurrences, deep or stacked reservoir layers, and structured state-space models are natural directions for future work.

6 Conclusion

We introduced a zero-rollout hyperparameter selector for reservoir-based temporal sequence learning, grounded in free-probability moments of the recurrent

matrix. The post-nonlinearity reservoir kernel – the Gram matrix of temporal features – converges at large width to a deterministic limit computable from a short labelled pilot sequence alone. This moves hyperparameter search entirely offline: no reservoir is instantiated during selection, the cost is independent of reservoir width, and the selected configuration transfers across widths without rerunning the search.

Across ten temporal benchmarks covering nonlinear sequence processing, memory, and chaotic forecasting, corrected pairwise tests detect no significant difference between FP and simulation-based selectors requiring up to 194 400 finite-reservoir rollouts. FP pre-screening reaches 0.772 versus 0.777 for full-grid Direct_{500} using 3.9% of the rollout budget. On a ten-sector real-data traffic-forecasting task, FPTop5+Direct_{500} matches Direct_{500} using 15 rather than 144 simulations per sector, while the task-agnostic Memory baseline collapses – showing that pilot-informed ranking is essential for multivariate forecasting.

Free-probability kernels thus provide an effective and theoretically grounded tool for efficient temporal model selection. Future work will extend the construction to nonlinear recurrences and broader classes of structured sequence models.

Acknowledgments. This work was supported by the following: Norwegian Research Council SFI NorwAI (309834); the European Union's HORIZON Research and Innovation Programme under grant agreement No 101120657, project ENFIELD (European Lighthouse to Manifest Trustworthy and Green AI); by EMERGE, a project funded by the European Innovation Council (prj. code 101070918); and by NEURONE, a project funded by the European Union - Next Generation EU, M4C1 CUP I53D23003600006, under program PRIN 2022 (prj. code 20229JRTZA).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**, 281–305 (2012)
2. Couillet, R., Wainrib, G., Ali, H.T., Sevi, H.: A random matrix approach to echo-state neural networks. In: International Conference on Machine Learning (ICML). pp. 517–525 (2016)
3. Dong, J., Ohana, R., Rafayelyan, M., Krzakala, F.: Reservoir computing meets recurrent kernels and structured transforms. In: Advances in Neural Information Processing Systems. vol. 33, pp. 16785–16796 (2020)
4. Gonon, L., Grigoryeva, L., Ortega, J.P.: Reservoir kernels and Volterra series. IEEE Transactions on Neural Networks and Learning Systems (2025)
5. Grigoryeva, L., Ortega, J.P.: Universal discrete-time reservoir computers with stochastic inputs and linear reservoirs using non-homogeneous state-affine systems. Journal of Machine Learning Research **19**(24), 1–40 (2018)
6. Gu, A., Goel, K., Re, C.: Efficiently Modeling Long Sequences with Structured State Spaces. In: International Conference on Learning Representations (2022), `https://openreview.net/forum?id=uYLFoz1vlAC`

7. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer series in statistics New-York, 2nd edn. (2009)
8. Hermans, M., Schrauwen, B.: Recurrent kernel machines: Computing with infinite echo state networks. Neural Computation **24**(1), 104–133 (2012)
9. Jaeger, H.: Short term memory in echo state networks (2001)
10. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German national research center for information technology gmd technical report **148**(34), 13 (2001)
11. Lagomarsini, G., Ceni, A., Gallicchio, C.: Benchmarking nonlinear readouts in linear reservoir networks. In: International Conference on Artificial Neural Networks. pp. 176–187. Springer (2025)
12. Lukoševičius, M.: A practical guide to applying echo state networks. In: Neural Networks: Tricks of the Trade, pp. 659–686. Springer, 2 edn. (2012)
13. Lukoševičius, M., Jaeger, H.: Reservoir computing approaches to recurrent neural network training. Computer Science Review **3**(3), 127–149 (2009)
14. Maat, J.R., Gianniotis, N., Protopapas, P.: Efficient optimization of echo state networks for time series datasets. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–7. IEEE (2018)
15. Matzner, F.: Hyperparameter tuning in echo state networks. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 404–412 (2022)
16. Mingo, J.A., Speicher, R.: Free probability and random matrices, vol. 35. Springer (2017)
17. Orvieto, A., Smith, S.L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., De, S.: Resurrecting recurrent neural networks for long sequences. In: International conference on machine learning. pp. 26670–26698. PMLR (2023)
18. Tanaka, G., Yamane, T., Héroux, J.B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., Hirose, A.: Recent advances in physical reservoir computing: A review. Neural Networks **115**, 100–123 (2019)
19. Thiede, L.A., Parlitz, U.: Gradient based hyperparameter optimization in echo state networks. Neural Networks **115**, 23–29 (2019)
20. Yildiz, I.B., Jaeger, H., Kiebel, S.J.: Re-visiting the echo state property. Neural Networks **35**, 1–9 (2012)

A Full Task-Level Deployment Scores

Table 6 gives per-task scores for the four primary selectors, including ESN Direct_{500}, at three representative large widths ($n \in \{5\,000, 10\,000, 20\,000\}$).

Table 6: Selected deployment widths. Values are $1 - \text{NRMSE}$ mean±std over seeds. Best score per task and width is bolded; second-best is underlined.

Task	Selector	$n = 5\,000 \uparrow$	$n = 10\,000 \uparrow$	$n = 20\,000 \uparrow$
MC	FP	0.423±0.001	0.457±0.006	0.468±0.025
	Memory_{500}	**0.609±0.003**	**0.676±0.012**	**0.714±0.007**
	Direct_{500}	0.423±0.001	0.457±0.006	0.468±0.025
	ESN Direct_{500}	0.388±0.008	0.433±0.014	0.470±0.008
NARMA10	FP	0.854±0.004	0.871±0.010	0.875±0.019
	Memory_{500}	0.795±0.009	0.827±0.006	0.836±0.005
	Direct_{500}	0.841±0.006	0.844±0.009	0.855±0.018
	ESN Direct_{500}	**0.867±0.009**	**0.874±0.011**	**0.876±0.013**
NARMA20	FP	0.886±0.006	0.915±0.009	0.924±0.007

Task	Selector	$n = 5\,000 \uparrow$	$n = 10\,000 \uparrow$	$n = 20\,000 \uparrow$
	Memory_{500}	0.800±0.014	0.843±0.016	0.861±0.010
	Direct_{500}	**0.915±0.005**	**0.931±0.007**	**0.937±0.005**
	ESN Direct_{500}	0.842±0.005	0.855±0.003	0.862±0.010
NARMA30	FP	**0.789±0.006**	**0.834±0.004**	**0.843±0.009**
	Memory_{500}	0.721±0.013	0.768±0.004	0.779±0.006
	Direct_{500}	0.781±0.008	0.829±0.003	0.840±0.009
	ESN Direct_{500}	0.608±0.008	0.609±0.010	0.609±0.012
NARMA50	FP	**0.606±0.004**	0.605±0.005	**0.611±0.009**
	Memory_{500}	0.605±0.005	**0.607±0.008**	0.605±0.004
	Direct_{500}	**0.606±0.005**	**0.607±0.005**	0.610±0.009
	ESN Direct_{500}	0.596±0.005	0.600±0.004	0.604±0.008
Inubushi	FP	0.860±0.006	0.870±0.007	0.879±0.005
	Memory_{500}	0.785±0.005	0.784±0.006	0.783±0.005
	Direct_{500}	0.964±0.002	0.966±0.003	0.968±0.002
	ESN Direct_{500}	**0.993±0.001**	**0.995±0.000**	**0.996±0.000**
Lorenz63	FP	0.570±0.002	0.575±0.002	0.579±0.002
	Memory_{500}	0.423±0.021	0.436±0.018	0.438±0.005
	Direct_{500}	**0.648±0.002**	0.652±0.005	0.663±0.008
	ESN Direct_{500}	0.578±0.039	**0.678±0.026**	**0.664±0.033**
SF-MG30	FP	0.932±0.003	**0.928±0.002**	**0.929±0.002**
	Memory_{500}	0.931±0.009	0.925±0.002	0.925±0.009
	Direct_{500}	**0.933±0.002**	**0.928±0.006**	0.924±0.003
	ESN Direct_{500}	0.907±0.008	0.923±0.004	0.928±0.007
MG84	FP	0.816±0.003	0.812±0.001	0.819±0.004
	Memory_{500}	0.653±0.067	0.605±0.003	0.625±0.002
	Direct_{500}	**0.825±0.002**	0.830±0.001	**0.833±0.003**
	ESN Direct_{500}	0.813±0.003	**0.831±0.008**	0.822±0.008
Lorenz96	FP	**0.677±0.001**	**0.686±0.001**	**0.687±0.002**
	Memory_{500}	0.576±0.024	0.580±0.005	0.573±0.021
	Direct_{500}	0.647±0.001	0.663±0.000	0.675±0.001
	ESN Direct_{500}	0.653±0.001	0.669±0.000	0.682±0.001

B Selection Time vs. Pilot Sequence Length

Figures 4–5 show selection performance and wall-clock time as a function of pilot sequence length T_{select}, at $n_{\text{select}} = 500$.

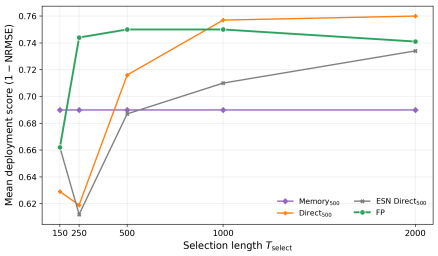

Fig. 4. Mean deployment score at $n = 5000$ as a function of T_{select}. Higher is better. FP reaches its plateau by $T_{\text{select}} = 250$–500; Direct_{500} and ESN Direct_{500} improve with longer sequences; Memory_{500} is flat.

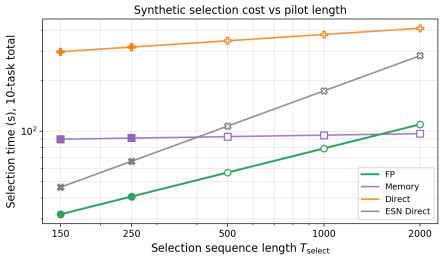

Fig. 5. Mean selection time per task averaged over all ten tasks, as a function of T_{select} ($n_{\text{select}} = 500$). Filled markers are measured runs. Lower is better.