$\boxed{\text{Milestone 3}}$

Model Architecture for Echo State Networks with Deep Baseline Comparators

Group 3 – DS and AI Lab

October 17, 2025

1 Scope & Deliverables

Objective. Select and justify model architectures for multi-horizon daily return forecasting with state-space flavor, centred on an Echo State Network (ESN) and three deep baselines (LSTM, Transformer, Temporal ConvNet). We document: (i) formal model definitions and training criteria; (ii) hyperparameter controls and expected behavior; (iii) how architectures integrate with our leakage-safe, walk-forward pipeline from Milestone 2; and (iv) an evaluation protocol to be executed in Milestone 4.

2 Problem Framing (State-Space View)

Let $u_t \in \mathbb{R}^d$ denote standardized exogenous features at day t (our engineered u_t: returns, vol, MAs, RSI, volume-z, weekday), and let targets be forward log-returns y_{t+h} for $h \in \{1, 5, 20\}$. A generic nonlinear state-space model writes

$$x_t = f(x_{t-1}, u_t), \qquad \hat{y}_{t+h} = g_h(x_t),$$

where $x_t \in \mathbb{R}^H$ is a latent state compressing recent history. Our ESN instantiates f with a fixed random reservoir and g_h with a trainable linear readout, yielding efficient training and stable temporal memory. Baselines (LSTM, Transformer, TCN) provide learnable alternatives to realize f end-to-end.

3 Proposed Model: Echo State Network (ESN)

3.1 Architecture

We employ a leaky-integrator ESN with tanh nonlinearity:

$$x_t = (1-a)\,x_{t-1} \;+\; a\,\tanh\big(W_{\text{in}}\,[1; u_t] \;+\; W\,x_{t-1}\big), \tag{1}$$

$$\hat{y}_{t+h} = \begin{bmatrix} 1 \\ x_t \end{bmatrix}^{\top} W_{\text{out},h}. \tag{2}$$

Here $a \in (0, 1]$ is the leak rate; $W \in \mathbb{R}^{H\times H}$ is a sparse random reservoir scaled to spectral radius $\rho(W) \approx \gamma \in (0, 1)$ to promote the *echo-state* property; $W_{\text{in}} \in \mathbb{R}^{H\times(d+1)}$ projects bias+inputs; $W_{\text{out},h} \in \mathbb{R}^{(H+1)}$ is a horizon-specific linear readout.

Training. We *do not* train W nor W_{in}. Given a training sequence, we roll (1) to collect states $\{x_t\}$, discard an initial *washout* of w steps, and solve, per horizon, a ridge-regularized least squares:

$$W^{\star}_{\text{out},h} \;=\; \arg\min_{W} \,\big\|HW - Y_h\big\|_2^2 + \alpha\|W\|_2^2, \qquad H = \begin{bmatrix} \mathbf{1} & X \end{bmatrix},$$

with X the post-washout state matrix. Closed-form: $W = (H^{\top}H + \alpha I)^{-1}H^{\top}Y_h$.

Key controls.

- **Spectral radius** γ (memory depth / stability), **leak** a (state smoothing), **hidden size** H (capacity), **density** (sparsity and compute), **washout** w (transient removal), **ridge** α (readout bias–variance).
- **State clipping** (optional) improves numerical robustness under stress events.

3.2 Why ESN for Finance?

Pros. (i) Low-variance training: only the readout is fit; (ii) nonlinearity with controlled memory via γ, a; (iii) fast CV over many folds/horizons; (iv) compatible with state-space interpretation (latent x_t). **Trade-offs.** Reservoir is random (requires seeds/averaging); no task-specific internal training (mitigated by hyperparameter search and ensembling).

3.3 Default Hyperparameters (starting grid)

Param	H	γ	a	density	washout	α
Values	400/600/800	0.8/0.9/0.95	0.3/0.6/1.0	0.05/0.1	100/150	0.1/1/3/10

4 Deep Baselines (Comparators)

All deep baselines operate on standardized features and consume *left-padded sliding windows* of length L so that each day has a valid sequence context and yields an aligned prediction.

4.1 Ridge Readout (Linear Lower Bound)

A regularized linear model on z_t features: $\hat{y}_{t+h} = \boldsymbol{w}_h^\top z_t + b_h$ with L2 penalty $\lambda \|\boldsymbol{w}_h\|_2^2$. Purpose: sanity check and performance floor.

4.2 LSTM Regressor

Sequence-to-one mapping with final hidden state readout. Architecture: 1–2 LSTM layers (hidden 128 by default) with MSE loss and Adam. Input: $(L, d) \to \mathbb{R}^o$ with $o \in \{1, 3\}$ targets. Expected strength: learn smooth temporal filters and gating; weakness: small data / nonstationarity may cause overfit without regularization.

4.3 Transformer Encoder

Linear projection to d_{model}, sinusoidal positional encoding, N encoder layers, final token readout. Good at long-range dependencies if L is moderate; regularization via dropout and weight decay. Risk: data scale vs. parameter count.

4.4 Temporal ConvNet (TCN)

Causal, dilated 1-D convolutions with residual connections (TCN blocks) and last-time readout. Efficient receptive field growth, strong inductive bias for local motifs; less parameter-hungry than Transformer.

Fairness policy. Same inputs, same folds, train-only scaling, identical loss (MSE), and fixed validation split strategy per fold. Hyperparameters remain modest (CPU-friendly) for reproducibility.

5 Data & Inputs (from Milestone 2)

We reuse standardized features u_t comprising returns/volatility/trend/RSI/volume/weekday and targets $\{\texttt{target\_h1}, \texttt{target\_h5}, \texttt{target\_h20}\}$. Splits are *walk-forward* with Train=2520 days ($\approx$10y), Test=252 days ($\approx$1y), Step=252 days; scalers fit on train only and saved per fold.

6 Integration & Modularity

Repository. Models are drop-in under `src/models/` with a registry:

- `EchoStateNetwork` (`esn.py`): state update (1), ridge readout (2).
- `LSTMRegressor` (`lstm.py`), `TransformerRegressor` (`transformer.py`), `TCNRegressor` (`tcn.py`).
- `registry.py`: `"ridge"`, `"esn"`, `"lstm"`, `"transformer"`, `"tcn"` → classes.

Pipeline. `main.ipynb` orchestrates: download → features → splits → materialize → `run_baseline(model, fold, horizon)`. Visuals live in `src/viz/plots.py` to compare RMSE/MAE/R^2/DirAcc and cumulative PnL.

7 Training & Inference Protocol

Common settings

- **Inputs.** Per-fold standardized features $\{z_t\}$; for deep models, left-padded windows of length L.
- **Targets.** $h \in \{1, 5, 20\}$ (trained per horizon).
- **Validation.** Chronological tail of train (e.g., 10%) for early selection/checkpoint.
- **Metrics.** RMSE, MAE, R^2, directional accuracy on test; plus simple sign-based P&L (1 bp cost) as *toy* diagnostic.

ESN specifics

Algorithm 1 ESN Training (per fold, per horizon)

1: Roll reservoir on train inputs using Eq. (1); collect states $\{x_t\}$.
2: Discard first w states (washout); form $H = [\mathbf{1}\ X]$.
3: Solve ridge: $W_{\text{out}} = (H^\top H + \alpha I)^{-1} H^\top Y$.
4: Save W_{out}, last train state x_{last}.
5: **Predict:** initialize with x_{last}, roll test inputs and emit $\hat{y}$ via Eq. (2).

8 Ablations & Hyperparameter Strategy

ESN. Sweep $(H, \gamma, a, \alpha, \text{density}, w)$ on a coarse grid; fix seed or average 3 seeds to reduce variance. **Deep models.** Small, compute-aware sweeps: `seq_len` $\in \{16, 32, 64\}$; LSTM hidden $\{64, 128\}$; Transformer layers $\{1, 2\}$, heads $\{2, 4\}$; TCN channels $(32, 32)$ or $(64, 64)$; moderate epochs (e.g., 10–20) with early selection by validation loss. Report fold-wise means and 95% CIs.

9 Risks, Assumptions & Mitigations

- **Weak signals.** Daily returns have low predictability. *Mitigation:* volatility-normalized targets, classification (direction), and cross-asset features.
- **Overfitting (deep).** *Mitigation:* parameter budgets, dropout/weight decay, validation splits, and multiple folds.
- **Reservoir sensitivity.** *Mitigation:* scale $\rho(W)$ via power iteration; seed averaging; washout.
- **Leakage risk.** Controlled by train-only scaling, strict walk-forward, and per-fold artifacts.

10 What is Done vs. Next (Milestone 4)

Completed (M3).

- Implemented ESN with leaky reservoir, spectral radius control, ridge readout, washout, optional state clipping; drop-in API.

- Added deep baselines (LSTM, Transformer, TCN) with left-padded sequence builders; unified `.fit/.predict` API.
- Registry-based orchestration; visualization utilities for metrics and backtests.

Next (M4).

- Run models across all folds/horizons; aggregate metrics and conduct significance tests (e.g., Diebold–Mariano) vs. ridge baseline.
- Hyperparameter sweeps per model (coarse-to-fine); ablation on ESN (H, γ, a).
- Extend features with cross-asset inputs (e.g., VIX for SPY) and compare.

Appendix A: Default Hyperparameters (code-aligned)

Model	Key Params (defaults)	File
Ridge	α=1.0	`src/models/ridge_readout.py`
ESN	H=500, γ=0.9, a=1.0, density=0.1, w=100, α=1.0	`src/models/esn.py`
LSTM	L=32, hidden=128, layers=1, epochs=10	`src/models/lstm.py`
Transformer	L=32, d_{model}=128, heads=4, layers=2, epochs=10	`src/models/transformer.py`
TCN	L=32, channels=(64, 64), k=3, epochs=10	`src/models/tcn.py`

Appendix B: Evaluation Metrics

We report RMSE, MAE, R^2, and directional accuracy on the test window per fold and horizon. A simple sign-based backtest with per-trade cost (default 1 bp) is included as a *diagnostic*:

$$\text{position}_t = \text{sign}(\hat{y}_t), \quad \text{pnl}_t = \text{position}_t \cdot y_t - \text{cost} \cdot |\Delta\text{position}_t|.$$

We summarize average daily P&L, volatility, Sharpe (daily $\to$ annualized by $\sqrt{252}$), hit ratio, and turnover. This is not a trading claim.

Ethics/Disclaimer. Academic project. No financial advice. Any backtest here is illustrative and uses simplified assumptions.