DA527-final project presentation

-Soumya Savarn, Ishan Chandra Gupta, Saptarshi Mukherjee

Title, URL of paper

Title: Nanowire Neural Networks for time-series processing

URL: https://openreview.net/pdf?id=TM9xxxLhtO

Motivation

- **Nanowire networks naturally act like recurrent systems** because their memristive junctions store and update state over time.

- **Neuromorphic computing benefits from physical dynamics**, and nanowires physically implement RNN-like nonlinear state transitions.

- **They form high-dimensional nonlinear reservoirs**, making them ideal for fast, low-training-cost sequence processing (RC / ESN).

- **Hybrid NW-ESN/NW-RNN models outperform classical RNNs**, giving scalable, efficient, and high-accuracy time-series processing.

Nanowire Memristor Physics

- Nanowire networks are meshes of tiny metallic wires whose junctions behave like synapses.
- Their structure closely resembles brain connectivity, enabling learning and memory-like behavior.
- Each junction acts as a memristive element whose conductance changes with past electrical activity.
- These dynamics mirror ion-flow mechanisms in biological neurons, similar to the Hodgkin–Huxley model.
- As a result, nanowire networks naturally function as brain-inspired computational substrates.

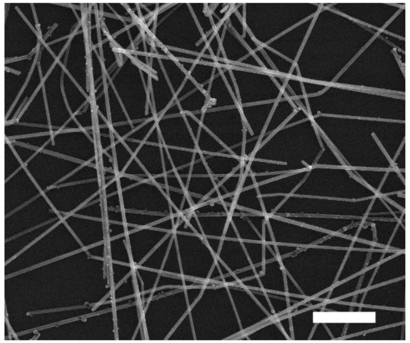
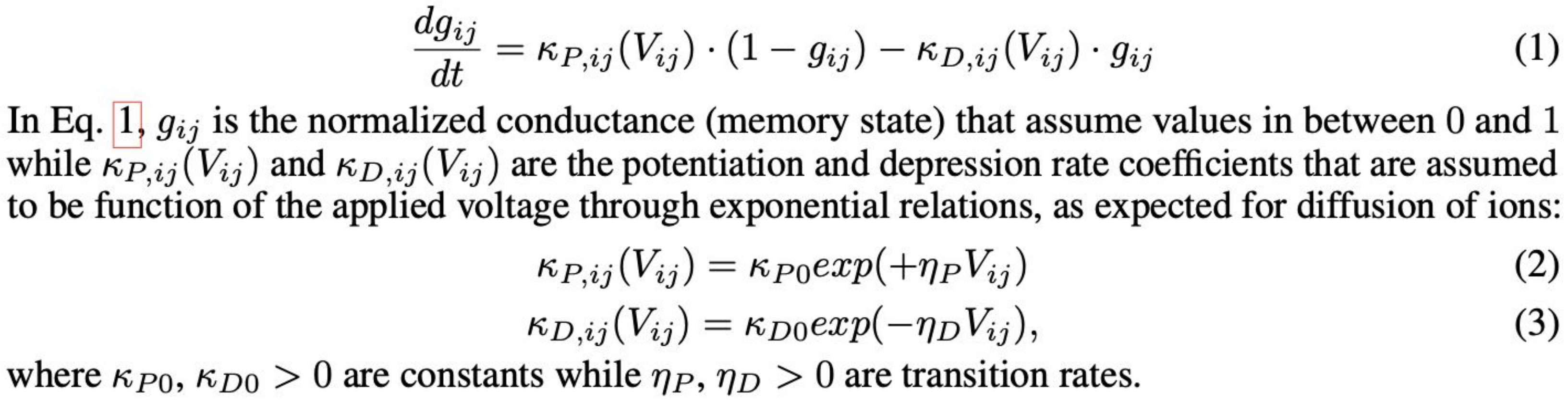


Figure 1: Image of a nanoscale self-assembled Ag nanowire network obtained through scanning electron microscopy (scale bar, 2 μm). This represents a memristive network, since each junction between nanowires is a memristive element and the current flow relies on the history of applied voltage and current.

Memristive Dynamics Equation

The state equation for the synaptic properties of the NW under electrical stimulation is expressed as the following potentiation–depression rate balance equation Miranda et al. [2020].



$$\frac{dg_{ij}}{dt} = \kappa_{P,ij}(V_{ij}) \cdot (1 - g_{ij}) - \kappa_{D,ij}(V_{ij}) \cdot g_{ij} \tag{1}$$

In Eq. 1, g_{ij} is the normalized conductance (memory state) that assume values in between 0 and 1 while $\kappa_{P,ij}(V_{ij})$ and $\kappa_{D,ij}(V_{ij})$ are the potentiation and depression rate coefficients that are assumed to be function of the applied voltage through exponential relations, as expected for diffusion of ions:

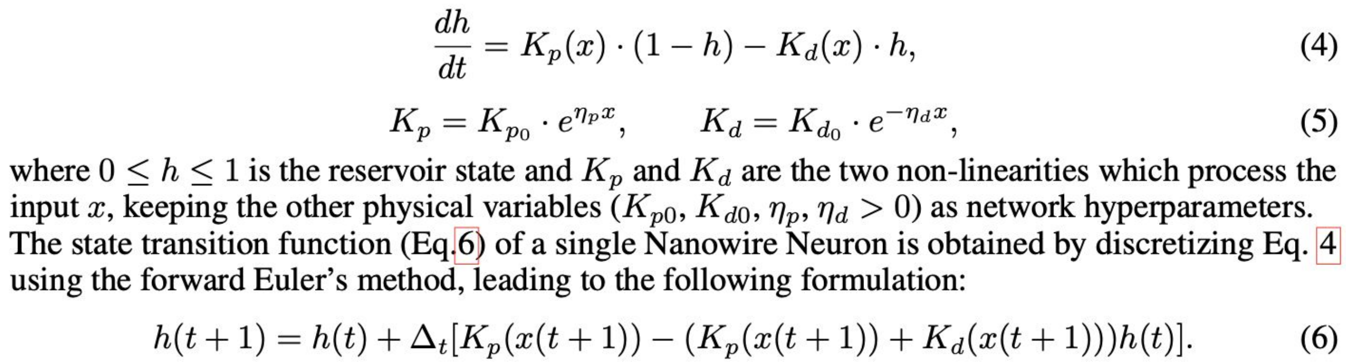
$$\kappa_{P,ij}(V_{ij}) = \kappa_{P0} exp(+\eta_P V_{ij}) \tag{2}$$

$$\kappa_{D,ij}(V_{ij}) = \kappa_{D0} exp(-\eta_D V_{ij}), \tag{3}$$

where κ_{P0}, $\kappa_{D0} > 0$ are constants while η_P, $\eta_D > 0$ are transition rates.

Computational Nanowire Model

The authors develop Nanowire Neural Networks for time-series processing where the physical memory state g becomes the nanowire network state. To achieve this, they rewrite Eq. 1, 2 and 3 as:



$$\frac{dh}{dt} = K_p(x) \cdot (1 - h) - K_d(x) \cdot h, \tag{4}$$

$$K_p = K_{p_0} \cdot e^{\eta_p x}, \qquad K_d = K_{d_0} \cdot e^{-\eta_d x}, \tag{5}$$

where $0 \leq h \leq 1$ is the reservoir state and K_p and K_d are the two non-linearities which process the input x, keeping the other physical variables (K_{p0}, K_{d0}, η_p, $\eta_d > 0$) as network hyperparameters. The state transition function (Eq.6) of a single Nanowire Neuron is obtained by discretizing Eq. 4 using the forward Euler's method, leading to the following formulation:

$$h(t+1) = h(t) + \Delta_t[K_p(x(t+1)) - (K_p(x(t+1)) + K_d(x(t+1)))h(t)]. \tag{6}$$

h belongs to [0, 1] is the internal memory state of a nanowire neuron.

Kp increases conductance(potentiation)

Kd decreases conductance(depression)

Nanowire Neural Network Architecture

Equation 6 is then used to construct the fully connected Nanowire Neural Network (Fig.2).

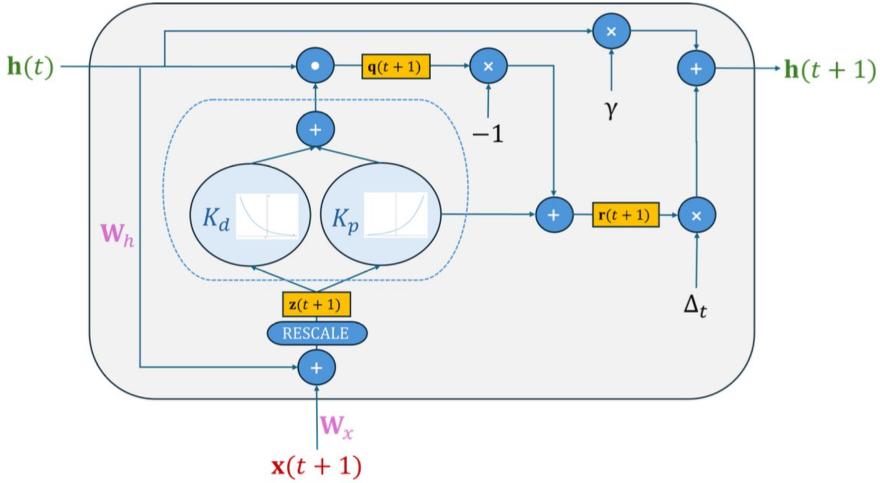


Figure 2: Nanowire Network computational graph where: $\mathbf{h}(t)$ and $\mathbf{h}(t+1)$, in green, are the current and the next hidden state vectors, respectively, $\mathbf{x}(t+1)$, in red, is the external input vector, $\mathbf{W}_h$ is the recurrent weight matrix (or recurrent kernel), $\mathbf{W}_x$, in magenta, is the input weight matrix (or kernel) and Δ_t is the discretization time (non-linearity operations and their trajectories are marked by a dashed line).

The dynamics of the Nanowire Neural Network are described by the following set of equations:

$$\mathbf{z}(t+1) = \text{RESCALE}\big(\mathbf{W}_h\mathbf{h}(t) + \mathbf{W}_x\mathbf{x}(t+1) + \mathbf{b}\big), \tag{7}$$
$$\mathbf{q}(t+1) = diag(K_p(\mathbf{z}(t+1) + K_d(\mathbf{z}(t+1))\mathbf{h}(t), \tag{8}$$
$$\mathbf{r}(t+1) = K_p(\mathbf{z}(t+1)) - \mathbf{q}(t+1), \tag{9}$$
$$\mathbf{h}(t+1) = \mathbf{r}(t+1)\Delta_t + \gamma\mathbf{h}(t), \tag{10}$$

where $\mathbf{h}(t)$ indicates the state of the recurrent layer, $\mathbf{x}(t)$ is the external input signal, $\mathbf{W}_h$ is the recurrent kernel, $\mathbf{W}_x$ is the (input-to-recurrent) kernel, and $\mathbf{b}$ is a bias vector.

The nanowire-based model introduces key physical hyperparameters (Kp0,Kd0,ηp,ηd,Δt,γ), where γ<1 ensures stable Echo State–like dynamics.

Input values shift the system's fixed point, so the affine transformation in Eq. 7 is rescaled to keep states in a useful operating range.

This RESCALE heuristic improves stability and learning performance before the nonlinear nanowire functions are applied.

The resulting architecture can be used both as a reservoir model (NW-ESN) and as a fully trainable RNN (NW-RNN).

NW-ESN: Reservoir model

NW-ESN is the reservoir-based Nanowire Neural Network Model in which both the kernel (Wx) and the recurrent kernel (Wh) matrices are initialized under stability constraints and left untrained. As usual in RC applications, we rescale the kernel Wx, the bias b, and recurrent kernel Wh, with the input scaling (ω), the bias scaling (β), and the spectral radius (ρ), respectively, and we use them as hyperparameters of the model.

- The input scaling controls how strongly the external input influences the reservoir state, with a large ω amplifying the input, yielding non linear responses at the risk of instability while small ω increases the importance of the internal network dynamics at risk of underfitting.
- Bias scaling(β) helps shift the internal state into a useful operating region—important because nanowire nonlinearities depend sensitively on the input range.
- Spectral radius(ρ) governs the stability and memory depth of the recurrent matrix Wh. It is defined as the largest absolute eigenvalue of Wh. Intuitively, a $\rho < 1$ implies fading memory while ρ close to 1 allows for longer memory with larger dynamics, although it does run the risk of instability.

NW-RNN: fully trainable model

- **NW-RNN** extends the nanowire model into a fully trainable recurrent network.
- Both the input kernel Wx and recurrent kernel Wh are optimized using **backpropagation through time (BPTT)**, just like in standard RNNs.
- The nanowire nonlinearities Kp, Kd, and the state update equation remain fixed, providing rich, physics-inspired dynamics.
- Training adjusts how inputs and past states influence the nanowire dynamics, enabling task-specific adaptation.
- This combines the expressiveness of RNNs with the stability and biological realism of nanowire-based state transitions.

Our Implementation Framework

Implemented the **Nanowire Neural Network (NW-NN)** architecture as described in the paper.

Developed both variants:

- **NW-ESN**: Fixed reservoir; trained only the readout layer.
- **NW-RNN**: Fully trainable input and recurrent weights using BPTT.

Implemented custom **nanowire state updates** using:

- Potentiation and depression dynamics
- Discretized ODE updates (forward Euler)
- Rescaling operations for stability

Created modular PyTorch classes for:

- Input transformation
- Hidden-state evolution
- Output prediction

Training Pipeline & Experimental Setup

Preprocessed time-series datasets (FordA, SyntheticControl, ECG5000, Earthquakes, FloodModeling1/2).

Used fixed hidden size (**N = 100**) for all models to ensure fair comparison.

For NW-ESN:

- Random initialization of Wx and Wh
- Spectral radius scaling + input scaling
- Linear readout trained with ridge regression

For NW-RNN:

- Trained using **Adam** optimizer
- Backpropagation through time with truncated sequence windows

Performed grid search over nanowire hyperparameters.

Evaluated models on each dataset using accuracy (classification) and MSE (regression).

Datasets (classification)

***FordA*:**

Description: The FordA dataset consists of engine sensor readings collected from an automotive system under normal and abnormal operating conditions. The task is to classify each univariate time series as “*healthy* or *faulty*” . The sequences are relatively long (around 500 time steps) and contain realistic noise patterns, making this dataset a challenging benchmark for time-series classification models.
Task: Binary classification

***ECG 5000*:**

Description: ECG 5000 contains electrocardiogram (ECG) recordings representing different heartbeat morphologies. It is a 5-class classification dataset derived from the PhysioNet BIDMC database. Each sequence is about 140 samples long and reflects subtle variations in heartbeat shapes. The dataset is also moderately imbalanced, with the “normal” heartbeat class dominating, making it a useful benchmark for models tested on medical time-series data and class imbalance settings.
Task: 5-class heartbeat classification

Datasets (classification)

***Synthetic Control*:**

Description: Synthetic Control is a classic benchmark dataset composed of artificially generated control-process signals. It includes six distinct types of temporal patterns, each representing a different class. The sequences are short (length 60) and noise-free, making this dataset ideal for assessing the baseline behavior of classification models and understanding their ability to separate clean, well-defined temporal structures.
Task: 6-class pattern classification

***Earthquakes*:**

Description: The Earthquakes dataset contains seismic sensor recordings used to distinguish between earthquake events and non-earthquake signals. It poses a binary classification problem with significant class imbalance, as true earthquake events are relatively rare. The sequences, roughly 512 samples long, include real-world noise and variability, making this dataset valuable for evaluating a model's sensitivity to rare event detection and noisy observational time-series.
Task: 5-class heartbeat classification

Datasets (regression)

***FloodModeling1*:**

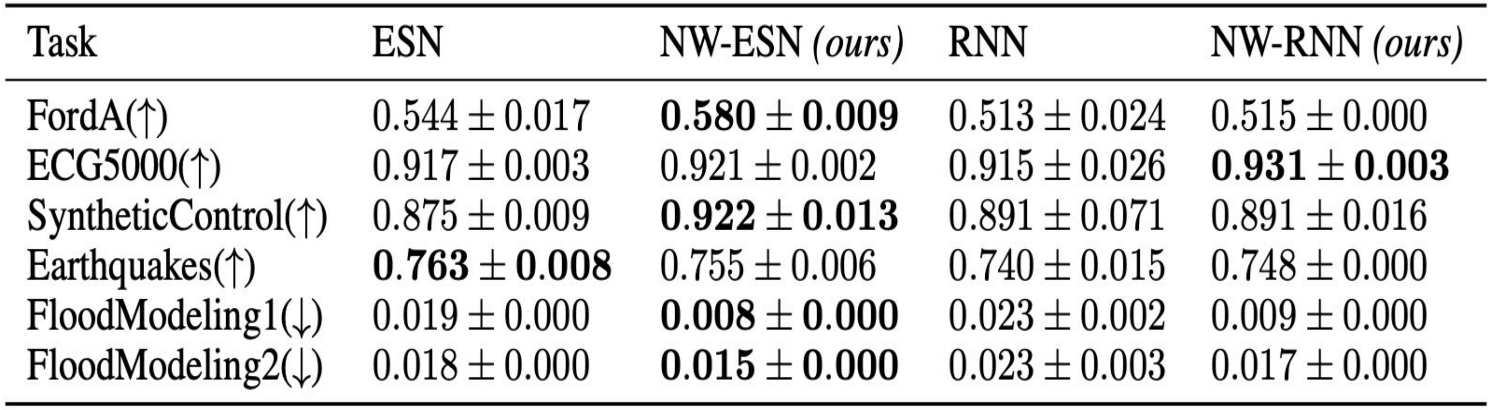
Description: FloodModeling1 is a real-world multivariate time-series regression dataset designed to predict water-level dynamics in a river basin. It contains several hydrological input signals such as rainfall, soil moisture, flow rate, and other environmental variables recorded over time. The task involves forecasting a continuous target variable—typically water level or discharge—making it a suitable benchmark for evaluating a model’s ability to learn long-term dependencies, nonlinear environmental dynamics, and complex multivariate relationships.

***FloodModeling2*:**

Description: FloodModeling2 is an extended and more complex variant of the first flood dataset. It includes a richer set of environmental variables and longer temporal sequences, capturing more detailed hydrological behavior across different time periods and weather conditions. Like FloodModeling1, the task is continuous-value prediction, but the increased complexity—including more varied patterns, delays, and dependencies—makes this dataset particularly useful for stress-testing the temporal modeling capacity of recurrent and reservoir-based architectures.

Baselines

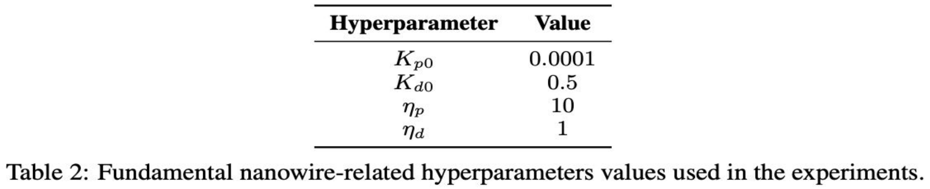
The results achieved demonstrate the competitive performance of nanowire-based models (NW-ESN and NW-RNN) in time-series tasks compared to standard methods such as ESN and RNN.



Task	ESN	NW-ESN *(ours)*	RNN	NW-RNN *(ours)*
FordA(↑)	0.544 ± 0.017	$\mathbf{0.580 \pm 0.009}$	0.513 ± 0.024	0.515 ± 0.000
ECG5000(↑)	0.917 ± 0.003	0.921 ± 0.002	0.915 ± 0.026	$\mathbf{0.931 \pm 0.003}$
SyntheticControl(↑)	0.875 ± 0.009	$\mathbf{0.922 \pm 0.013}$	0.891 ± 0.071	0.891 ± 0.016
Earthquakes(↑)	$\mathbf{0.763 \pm 0.008}$	0.755 ± 0.006	0.740 ± 0.015	0.748 ± 0.000
FloodModeling1(↓)	0.019 ± 0.000	$\mathbf{0.008 \pm 0.000}$	0.023 ± 0.002	0.009 ± 0.000
FloodModeling2(↓)	0.018 ± 0.000	$\mathbf{0.015 \pm 0.000}$	0.023 ± 0.003	0.017 ± 0.000

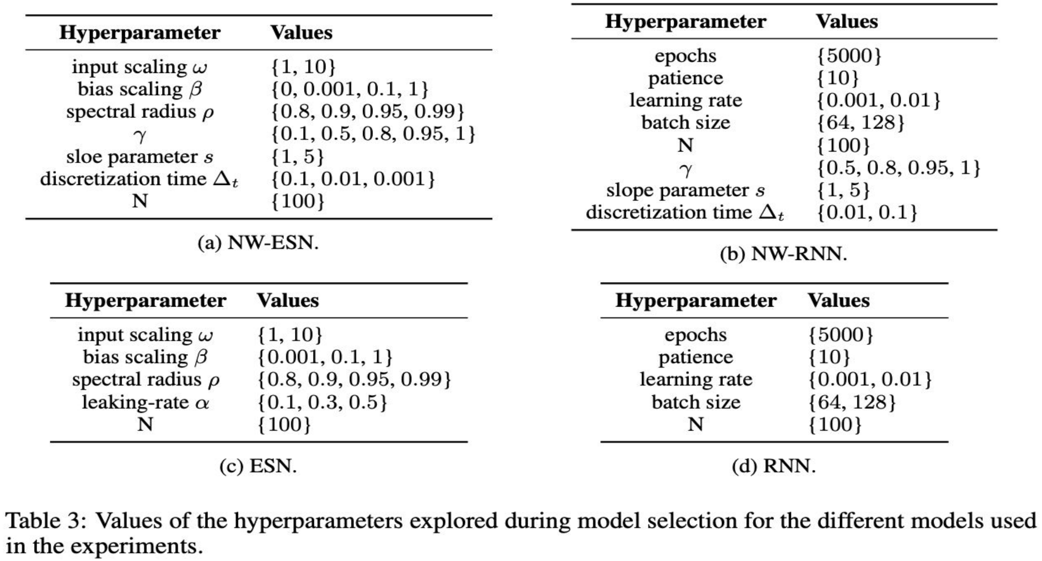
Table 4: Experimental results. Best results are highlighted in bold.

Hyperparameters



Hyperparameter	Value
K_{p0}	0.0001
K_{d0}	0.5
η_p	10
η_d	1

Table 2: Fundamental nanowire-related hyperparameters values used in the experiments.



Hyperparameter	Values
input scaling ω	{1, 10}
bias scaling β	{0, 0.001, 0.1, 1}
spectral radius ρ	{0.8, 0.9, 0.95, 0.99}
γ	{0.1, 0.5, 0.8, 0.95, 1}
sloe parameter s	{1, 5}
discretization time Δ_t	{0.1, 0.01, 0.001}
N	{100}

(a) NW-ESN.

Hyperparameter	Values
epochs	{5000}
patience	{10}
learning rate	{0.001, 0.01}
batch size	{64, 128}
N	{100}
γ	{0.5, 0.8, 0.95, 1}
slope parameter s	{1, 5}
discretization time Δ_t	{0.01, 0.1}

(b) NW-RNN.

Hyperparameter	Values
input scaling ω	{1, 10}
bias scaling β	{0.001, 0.1, 1}
spectral radius ρ	{0.8, 0.9, 0.95, 0.99}
leaking-rate α	{0.1, 0.3, 0.5}
N	{100}

(c) ESN.

Hyperparameter	Values
epochs	{5000}
patience	{10}
learning rate	{0.001, 0.01}
batch size	{64, 128}
N	{100}

(d) RNN.

Table 3: Values of the hyperparameters explored during model selection for the different models used in the experiments.

Classification Results

The following table demonstrate the performance of nanowire-based models (NW-ESN and NW-RNN) in time-series tasks compared to standard methods such as ESN and RNN as achieved by our code.

Task	ESN	NW-ESN	RNN	NW-RNN
FordA	0.494	0.558	0.486	**0.692**
ECG5000	0.838	0.820	**0.932**	0.928
SyntheticControl	0.450	0.730	**0.963**	0.947
Earthquakes	0.712	**0.763**	0.741	0.748

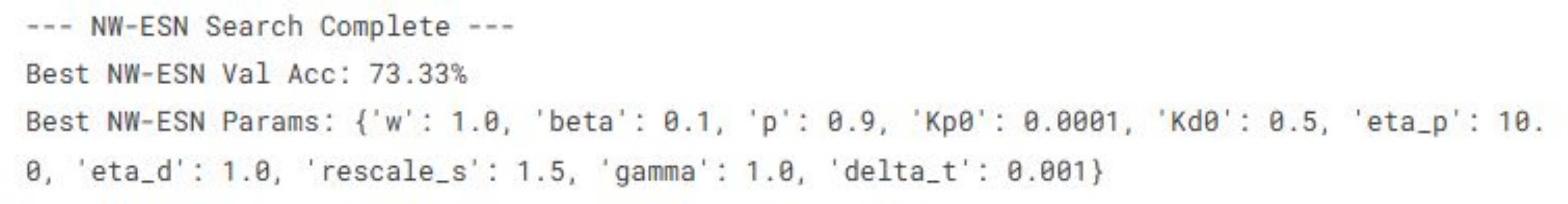
Regression Results

The following table demonstrate the performance of nanowire-based models (NW-ESN and NW-RNN) in time-series tasks compared to standard methods such as ESN and RNN as achieved by our code.

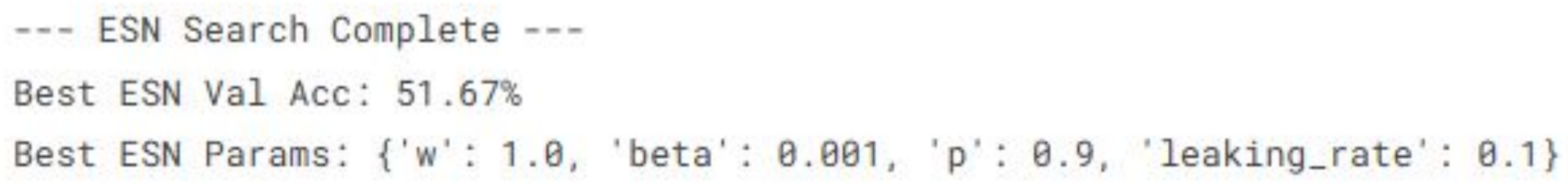
Task	ESN	NW-ESN	RNN	NW-RNN
FloodModelling1	0.1038	**0.0177**	0.0180	0.0184
FloodModelling2	0.0913	0.0959	0.0182	**0.0143**

For classification tasks, performance is evaluated using accuracy, while for regression tasks, we report the root mean squared error.

Ablations-Hyperparameter search result



```
--- NW-ESN Search Complete ---
Best NW-ESN Val Acc: 73.33%
Best NW-ESN Params: {'w': 1.0, 'beta': 0.1, 'p': 0.9, 'Kp0': 0.0001, 'Kd0': 0.5, 'eta_p': 10.
0, 'eta_d': 1.0, 'rescale_s': 1.5, 'gamma': 1.0, 'delta_t': 0.001}
```



```
--- ESN Search Complete ---
Best ESN Val Acc: 51.67%
Best ESN Params: {'w': 1.0, 'beta': 0.001, 'p': 0.9, 'leaking_rate': 0.1}
```

Insights

Rich Intrinsic Dynamics: Nanowire networks naturally exhibit nonlinear, high-dimensional temporal dynamics. When used as a reservoir or recurrent layer, these dynamics generate diverse internal trajectories that make time-dependent patterns easier to separate.

Memory + Nonlinearity Balance: The potentiation and depression processes in the nanowire model provide a balance between short-term memory (state retention) and nonlinear transformation. This is ideal for tasks requiring both temporal tracking and feature extraction.

Stability Through Physical Constraints: The nanowire ODEs include natural saturation and rescaling terms, which prevent exploding or vanishing states. This leads to more stable training, especially compared to standard RNNs at similar hidden sizes.

Implicit Feature Expansion: The dynamic state updates act as a continuous feature expansion mechanism, mapping simple input sequences into a more expressive state space—similar to kernel methods but implemented through physical-inspired recurrent dynamics.

Limitations

Sensitive Hyperparameters: Nanowire dynamics depend on several physical-inspired parameters (potentiation, depression rates, scaling factors). Small changes can destabilize training or degrade performance.

ODE-Based Updates Are Costly: The state evolution requires numerical integration at every timestep, which is computationally heavier than standard RNN updates and slows training on long sequences.

Performance Not Always Superior: On simpler datasets or short sequences, NW models may not outperform LSTMs/GRUs or even ESNs, especially when hyperparameters are not well tuned.

Scalability Challenges: Hidden dimensions are typically small (e.g., 50–100). Scaling to hundreds or thousands of units may cause numerical instability or increased training time.

Conclusion

In this work, we implemented both variants of Nanowire Neural Networks—**NW-ESN** and **NW-RNN**—and evaluated them on a diverse collection of datasets. Our experiments show that nanowire-inspired dynamics can generate rich temporal representations and capture meaningful patterns in sequential data. Despite being based on simple physical principles, these models can perform competitively with traditional recurrent architectures. Their inherent stability, caused by built-in saturation and rescaling behavior, makes them an interesting alternative for time-series modeling.

Future exploration can focus on identifying and understanding bottlenecks in the nanowire update equations to determine where information is being distorted or lost. Scaling the architectures to larger reservoirs and experimenting with more advanced numerical integration methods could further improve performance on long sequences. Additionally, incorporating structured sparsity or connectivity patterns may help mimic real nanowire networks more closely. Expanding the models to handle multivariate or irregularly sampled time-series data, and exploring hybrid architectures that blend nanowire dynamics with attention mechanisms or transformers, represent promising directions for extending this line of research.