Results in Engineering 29 (2026) 109716

Contents lists available at ScienceDirect

Results in Engineering

journal homepage: www.sciencedirect.com/journal/results-in-engineering

Research paper

Differentiable neural search architecture with zero-cost metrics for insulator fault prediction

Laio Oriel Seman [a,*], William Gouvêa Buratto [b,c], Gabriel Villarrubia Gonzalez [c], Valderi Reis Quietinho Leithardt [c,d,e], Ademir Nied [b], Stefano Frizzo Stefenon [b,e,f]

[a] Department of Automation and Systems Engineering, Federal University of Santa Catarina (UFSC), 88040-900, Florianopolis, SC, Brazil
[b] Systems Control Research Group, Department of Electrical Engineering, Santa Catarina State University (UDESC), 89219-710, Joinville, SC, Brazil
[c] Faculty of Science, Expert Systems and Applications Lab, University of Salamanca, 37008, Salamanca, Spain
[d] Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR, 1649-026, Lisboa, Portugal
[e] Center of Technology and Systems (UNINOVA-CTS) and Associated Lab of Intelligent Systems (LASI), NOVA University Lisbon, 2829-516, Lisbon, Portugal
[f] Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, Rua Conselheiro Emídio Navarro 1, 1959-007, Lisboa, Portugal

ARTICLE INFO

Keywords:
Differentiable neural architecture
Forecasting
Neural network architectures
Predictive maintenance

ABSTRACT

Reliable monitoring of high-voltage insulators is critical for maintaining the stability of electrical power systems, particularly under environmental contamination that can lead to flashover. Traditional inspection techniques struggle to anticipate degradation dynamics, while data-driven models often rely on fixed neural architectures that inadequately capture the complex temporal patterns in leakage current signals. This work proposes a Differentiable Neural Architecture Search (DARTS) framework, based on zero-cost metrics, tailored for time series forecasting in insulator monitoring. The method based on DARTS integrates a mixed encoder-decoder design with learnable selection over long short-term memory, gated recurrent units, and transformer components, coupled with a cross-attention bridge featuring temporal bias and gating mechanisms. To ensure efficient architecture exploration, the search leverages metrics such as SynFlow and Jacobian covariance for early candidate screening, followed by a bilevel optimization stage with entropy and diversity regularization. Experiments on real-world leakage current data demonstrate that the discovered architectures outperform manually designed baselines, offering improved forecasting performance.

1. Introduction

The reliability of electrical power systems fundamentally depends on the integrity of their insulating components, particularly high-voltage insulators that are continuously exposed to environmental contamination [1]. Insulators installed in outdoor environments are vulnerable to the accumulation of contaminants on their surface, including industrial waste, coastal salinity, and dust from unpaved roads, which progressively increase surface conductivity and leakage current until flashover occurs [2].

Contaminated insulators constitute an anomaly that can compromise the integrity of the grid insulation; while not directly a fault, they can lead to a disruptive fault when very high levels of leakage current occur [3]. The urgency of this research problem is underscored by the growing frequency and impact of power outages associated with insulation degradation in transmission and distribution networks [4]. Insulator-related faults represent a non-negligible share of line outages worldwide, particularly in coastal, industrial, and arid regions, where contamination severity is intensified by environmental [5] and climatic conditions [6]. Such outages often propagate beyond local failures, leading to widespread service interruptions, reduced power quality, and cascading operational challenges for grid operators [7].

The economic consequences are substantial, encompassing direct costs related to emergency maintenance and component replacement, as well as indirect losses due to interrupted industrial production, service downtime, and penalties associated with reliability indices. As electrical grids expand and operate closer to their stability limits [8], even short-duration outages caused by insulator flashovers can result in significant financial losses and reduced public confidence in power supply reliability [9]. Consequently, developing predictive methods capable of anticipating contamination-induced insulation breakdown is not only a technical challenge but also an urgent practical necessity for improving grid resilience, reducing operational costs, and

* Corresponding author.
E-mail addresses: laio.seman@ufsc.br (L.O. Seman), william.buratto@edu.udesc.br (W.G. Buratto), gvg@usal.es (G.V. Gonzalez), valderi.leithardt@iscte-iul.pt (V.R.Q. Leithardt), ademir.nied@udesc.br (A. Nied), stefano.stefenon@isel.pt (S.F. Stefenon).

https://doi.org/10.1016/j.rineng.2026.109716
Received 10 January 2026; Received in revised form 9 February 2026; Accepted 22 February 2026
Available online 25 February 2026

supporting the transition toward more reliable and sustainable power systems [10].

Considering the impacts of power outages, this paper focuses on predicting failures in electrical power grids by monitoring the increase in leakage current from contaminated insulators. Thus, the main goal of this analysis is to predict when contamination becomes high enough to cause an electrical breakdown, thereby allowing proactive measures to be taken to prevent power outages. Traditional monitoring approaches rely on periodic inspections by specialized personnel, but these methods struggle to predict failure progression and determine optimal maintenance schedules, particularly when visual indicators are absent or subtle [11].

Leakage current measurement is a reliable indicator for assessing insulator condition and predicting impending failures [12]. Unlike visual inspection methods that may miss critical degradation stages, leakage current provides a quantitative assessment of surface conductivity changes that precede flashover events [13]. However, the temporal patterns in leakage current exhibit complex, non-linear characteristics with significant noise components arising from environmental variations, measurement artifacts, and the inherent non-linearity of electrical discharge phenomena [14]. These complexities present substantial challenges for traditional time series forecasting approaches, which often rely on fixed architectural assumptions that may not optimally capture the diverse temporal dependencies present in electrical insulation degradation processes.

Recent advances in deep learning have demonstrated promising results for time series forecasting in electrical power systems, with Long Short-Term Memory (LSTM) networks and attention mechanisms [15] showing particular effectiveness for capturing long-term dependencies in temporal data [16]. However, the manual design of neural network architectures for time series forecasting remains a labor-intensive process requiring extensive domain expertise and empirical tuning. Different aspects of insulator degradation may benefit from different architectural components-for instance, high-frequency noise may require specialized filtering operations, while long-term degradation trends may benefit from different temporal modeling approaches.

Neural Architecture Search (NAS) offers a promising solution to this challenge by automating the discovery of optimal network architectures for specific forecasting tasks. Traditional NAS approaches, however, suffer from prohibitive computational costs that scale exponentially with search space size, making them impractical for many real-world applications. Differentiable Architecture Search (DARTS) addresses this limitation by reformulating the discrete architecture search problem as a continuous optimization problem, enabling efficient gradient-based optimization of architectural parameters [17].

Despite these advances, existing DARTS frameworks face several limitations when applied to time series forecasting, particularly for critical infrastructure monitoring applications like insulator fault prediction. First, standard DARTS implementations often converge to trivial solutions dominated by identity operations, limiting architectural diversity and exploration of meaningful design choices [18]. Second, traditional encoder-decoder architectures employ fixed architectural paradigms that may not optimally adapt to the varying temporal characteristics of different forecasting problems [19]. Third, existing attention mechanisms typically rely on content-based similarity measures that may not capture the specific temporal relationships and patterns inherent in time series data [20].

This work addresses these limitations through several key contributions. We introduce an enhanced differentiable architecture search framework specifically designed for time series forecasting that incorporates mixed encoder-decoder architectures with learnable component selection. The framework enables dynamic architecture selection at the sequence modeling level, allowing the model to automatically discover optimal combinations of LSTM [21], Gated Recurrent Units (GRU) [22], and Transformer components [23] based on data characteristics. We develop a novel cross-attention bridge mechanism with learnable temporal bias that captures sequence-dependent relationships specific to time series forecasting, extending beyond standard content-based attention mechanisms. Additionally, we implement comprehensive regularization techniques, including entropy regularization, identity operation penalties to ensure exploration of diverse and meaningful architectural solutions.

We demonstrate the effectiveness of our approach on insulator leakage current prediction, a critical application where accurate forecasting can prevent catastrophic power system failures. The proposed framework automatically discovers architectures that achieve superior forecasting performance compared to manually designed networks, while providing insights into the temporal modeling strategies that prove most effective for electrical insulation degradation patterns. The zero-cost evaluation metrics enable efficient architecture screening without full training, significantly reducing computational overhead while maintaining high-quality architecture discovery.

Based on that, the contributions of this work are:

- Proposes a differentiable architecture search framework for time series forecasting in insulator fault prediction.
- Introduces a mixed encoder-decoder design combining LSTM, GRU, and Transformer components with learnable architecture selection.
- Develops a novel cross-attention bridge mechanism with learnable temporal bias and adaptive gating for sequence modeling.
- Utilizes zero-cost evaluation metrics for early architecture screening, reducing training cost.
- Implements entropy and diversity regularization to promote architectural diversity and avoid trivial solutions.
- Employs a multi-fidelity architecture search strategy combining zero-cost metrics and bilevel optimization for efficient model discovery.
- Achieves superior forecasting performance on real-world insulator leakage current datasets.
- Demonstrates the ability to adaptively model degradation processes in electrical power systems with reduced computational overhead.

The remainder of this paper is organized as follows: After introducing the critical need for reliable insulator fault prediction and the limitations of traditional monitoring and forecasting approaches, the paper presents the proposed differentiable architecture search framework in detail. Section 2 surveys related work on machine learning techniques for insulator monitoring. Section 3 introduces the enhanced DARTS-based architecture, including its cell structure, mixed encoder-decoder components, and cross-attention mechanisms, followed by the multi-fidelity search strategy and regularization methods. Section 4 reports the experimental results, evaluating architecture search efficiency and forecasting performance. Finally, Section 5 concludes with the contributions and implications of the proposed approach.

2. Related works

The reliable prediction of faults in engineering systems is a central challenge in modern monitoring and maintenance strategies, as early detection of degradation can significantly reduce operational risks and unexpected failures [24]. In many applications, this task depends on identifying informative precursors embedded in noisy and nonstationary data, which requires robust data-driven modeling approaches capable of capturing subtle temporal and structural patterns [25]. At the same time, the increasing complexity of learning models and the diversity of operating conditions highlight the importance of systematic methods for model design and optimization, rather than ad hoc or purely manual solutions [26]. Together, these aspects motivate research efforts that combine domain-specific knowledge with advanced learning frameworks to improve predictive accuracy, generalization, explainability, and adaptability across a wide range of real-world fault prediction problems [27].

2.1. Insulator fault detection

Several studies have addressed the problem of insulator fault prediction by focusing on the monitoring and analysis of leakage current as a key indicator of degradation [28]. Early works emphasized threshold-based and signal processing techniques to detect abnormal leakage current patterns associated with contamination, aging, and environmental stress, demonstrating their relevance for preventive maintenance in power systems [29]. More recent research has explored advanced monitoring frameworks that combine sensing technologies with data-driven methods to improve sensitivity and robustness under varying operating conditions [30]. Investigations on polymeric and ceramic insulators have shown that leakage current characteristics are strongly correlated with surface pollution and moisture, reinforcing their suitability as fault precursors [31].

Traditional fault detection techniques include thermal imaging [32], ultraviolet radiation [33], and acoustic analysis [34], which can be performed using an Unmanned Aerial Vehicle (UAV) [35]. However, these are often reactive, requiring field inspections and lacking temporal forecasting. Consequently, machine learning and signal processing-based approaches have gained prominence. Sopelsa Neto et al. [36] evaluated forecasting methods such as LSTM, Group Method of Data Handling (GMDH), adaptive-network-based fuzzy inference system, and ensemble learning techniques, including bagging, boosting, and stacking, applying wavelet transforms to enhance signal preprocessing. Their findings indicated that wavelet-enhanced hybrid models outperformed classical models in predicting the evolution of leakage current.

Several other works applied similar attention-based architectures. Beyond architecture, the use of filters has been fundamental in denoising signals. Stefenon et al. [37] proposed a method combining the Christiano-Fitzgerald random walk filter with GMDH. This hybrid model achieved superior noise suppression and forecasting accuracy compared to standard LSTM and GMDH methods. To further improve long-sequence modeling, attention-based mechanisms were integrated into forecasting architectures. Klaar et al. [38] introduced an optimized Empirical Wavelet Transform (EWT)-sequence-to-sequence-LSTM model with attention, which demonstrated increased forecasting precision when combined with EWT and hyperparameter optimization using Optuna.

Composite insulators are widely used in power systems for their low weight, high strength, and hydrophobicity, but polymer aging poses challenges for their condition diagnosis. Jin et al. [39] proposed using a knowledge graph to enhance diagnostic efficiency by structuring expert knowledge. Two methods being the lite bidirectional encoder representations from transformer (ALBERT), bidirectional LSTM (BiLSTM), conditional random field (CRF) model, and the ALBERT-bidirectional GRU (BiGRU)-attention, were used for entity and relation extraction, achieving F1-scores of 92.49% and 92.36%. The resulting knowledge graph, with 820 triplets, integrates data on conditions, methods, and faults, supporting AI-driven diagnosis.

The increasing occurrence of decay-like fractures in composite insulator core rods has raised concerns about transmission line reliability. Yet, quantitative assessment methods for evaluating degradation levels remain limited. Fang et al. [40] proposed an ultrasonic guided wave-based approach to assess the degradation degree of core rods collected from the field. Both simulations and experiments were performed, identifying the 35 kHz T(0,1) and 55 kHz L(0,1) wave modes as effective for detection. Results showed that the peak-to-peak amplitude of received signals decreased with greater decay. Given that, this work offers a practical reference for evaluating decay-like degradation in composite insulator core rods.

Insulator failures, often caused by flashovers, pose serious threats to power system reliability. Lutfi et al. [41] proposed a deep learning-based method combining ultrasonic sensing with a convolutional neural network to detect and classify ceramic insulator defects. Their model was trained on ultrasonic data from a defective two-disk setup in the lab and tested in both a three-disk setup and a real 138/13.8kV substation. It achieved accuracies of 99% and 96% in lab settings, and 91% in the field. Using gradient-weighted class activation mapping, the model's attention to high-frequency spectral features was revealed, highlighting its potential for accurate and nonintrusive insulator condition monitoring.

Novel techniques for insulator state diagnostics include spectral analysis of leakage current [42], wavelet preprocessing [43], using transformers [44], digital fault recorders [45], knowledge transfer with attention mechanism network [46], considering infrared thermal cameras [47], and COMSOL-based simulations for electric field stress evaluation [48]. These approaches further validate the correlation between leakage current trends and flashover probability. These works highlight a trajectory from classic signal acquisition toward hybrid, filter-enhanced, deep learning-based forecasting models, establishing a strong foundation for the development of robust predictive maintenance tools in power systems.

Considering this problem, Differentiable Neural Architecture Search (DNAS) has emerged as a powerful paradigm for automating the design of neural network architectures by enabling gradient-based optimization over the architecture space [49]. Instead of relying on discrete and computationally expensive trial and error strategies, this approach relaxes architectural choices into continuous parameters, allowing efficient joint optimization of both network weights and structure [50]. As a result, Differentiable Neural Search has been successfully applied in domains such as computer vision, natural language processing, and time series forecasting, where it has demonstrated competitive or superior performance compared to manually designed models while significantly reducing search cost [51].

2.2. Differentiable neural architecture search applications

DNAS has been successfully applied to various domains, including time series forecasting, which aligns closely with the challenges in fault prediction from nonstationary signals like leakage current. The foundational work, DARTS, introduced a continuous relaxation of the architecture representation, allowing gradient-based optimization to efficiently search for high-performing networks in vision and other tasks [52]. This method significantly reduced search times compared to earlier reinforcement learning or evolutionary approaches.

In high-dimensional time series forecasting, DNAS has been adapted to handle multivariate data, such as stock prices. A key study proposed a DNAS framework that automates the design of neural models for such data, achieving superior accuracy on real-world datasets [53]. Recent advancements include hierarchical NAS approaches for time series, where a broad search space encompasses various forecasting models. This method optimizes architectures hierarchically, leading to improved generalization and efficiency in forecasting tasks [54].

For financial time series forecasting, chain-structured NAS has been explored, comparing optimization strategies like Bayesian optimization, hyperband, and reinforcement learning on architectures such as Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), and Recurrent Neural Networks (RNNs). This work highlights the effectiveness of simple, chain-based search spaces for small datasets, with Bayesian methods often yielding the best results despite high variance in financial data [55]. Directly relevant to fault diagnosis, DNAS has been applied to domain adaptation problems in mechanical systems. By automating architecture design while addressing distribution shifts between source and target domains, this approach enhances fault detection accuracy under varying conditions [56].

For power systems, differentiable NAS has been applied to optimize Transformer architectures for power system fault detection, reducing the need for manual design of attention and feedforward layers [57]. Transformers are viewed as a suitable alternative to deep CNN attention and RNN-based models, as they can reduce memory demands while better capturing long-term dependencies in electrical signals. Given the

variety of possible attention mechanisms and feedforward structures, manual design becomes inefficient, motivating the use of differentiable architecture search strategies such as DARTS to enable end-to-end optimization with reduced search cost. DNAS emerges as a viable solution to automatically construct deep learning architectures by exploring structured search spaces while relaxing discrete design choices into continuous ones [58].

3. Differentiable architecture search for time series forecasting

Neural architecture search automates the design of neural network architectures, eliminating manual trial-and-error approaches. Traditional neural architecture search methods require training numerous candidate architectures to convergence, imposing computational costs that scale exponentially with search space size. This work presents an enhanced differentiable architecture search framework for time series forecasting that integrates mixed encoder-decoder architectures with cross-attention bridging mechanisms and comprehensive regularization techniques to ensure architectural diversity and prevent premature convergence.

Let $\mathcal{A}$ represent the space of all possible neural architectures for time series forecasting tasks. Each architecture $a \in \mathcal{A}$ is characterized by its operation set $\mathcal{O}_a$, connectivity patterns $\mathcal{C}_a$, and hyperparameter configuration $\mathcal{H}_a$. The objective is to identify the optimal architecture $a^* \in \mathcal{A}$ that minimizes the validation loss function:

$$a^* = \arg\min_{a \in \mathcal{A}} \mathcal{L}_{\text{val}}(a; \mathcal{D}_{\text{val}}, \theta_a^*) \tag{1}$$

where $\mathcal{D}_{\text{val}}$ denotes the validation dataset and θ_a^* represents the optimal network parameters obtained through training on the training dataset $\mathcal{D}_{\text{train}}$:

$$\theta_a^* = \arg\min_{\theta} \mathcal{L}_{\text{train}}(a; \mathcal{D}_{\text{train}}, \theta) \tag{2}$$

3.1. DARTS cell architecture

The architecture is composed of repeated cells, each represented as a directed acyclic graph $G = (V, E)$, where nodes correspond to latent feature states and edges represent candidate operations. Given N nodes, the graph is defined by the vertex set $V = \{v_0, v_1, \ldots, v_{N-1}\}$ and the directed edge set $E = \{(v_i, v_j) \mid 0 \le i < j < N\}$. Each node v_j aggregates incoming transformations from its predecessors according to

$$x^{(j)} = \text{Aggregate}\left(\left\{o^{(i,j)}(x^{(i)}) \cdot \sigma(\gamma_{i,j}) \mid i < j\right\}\right), \tag{3}$$

where $o^{(i,j)}$ denotes the operation assigned to edge (i, j) and $\gamma_{i,j}$ is a learnable scalar controlling edge importance.

Each operation is selected from a hierarchical search space. The operation set $\mathcal{O}$ is partitioned into groups $\{\mathcal{O}_g\}_{g=1}^{G}$, where each group shares a semantic category (e.g., temporal, frequency, multiscale). The operation applied at edge (i, j) is given by

$$o^{(i,j)}(x) = \sum_{g=1}^{G} \pi_g^{(i,j)} \sum_{k \in \mathcal{O}_g} \pi_{g,k}^{(i,j)} \cdot \text{op}_{g,k}(x), \tag{4}$$

where $\pi_g^{(i,j)}$ and $\pi_{g,k}^{(i,j)}$ are softmax- or Gumbel-softmax-based probabilities parameterized by learnable logits and temperature τ.

The aggregation function in (3) can take different forms depending on the configuration:

- Mean aggregation:

$$\text{Aggregate}_{\text{mean}}(\mathcal{I}_j) = \frac{1}{|\mathcal{I}_j|} \sum_{z \in \mathcal{I}_j} z \tag{5}$$

- Weighted aggregation:

$$\text{Aggregate}_{\text{weighted}}(\mathcal{I}_j) = \sum_{i=1}^{|\mathcal{I}_j|} \beta_i^{(j)} z_i, \quad \beta_i^{(j)} = \text{softmax}(\delta_i^{(j)}) \tag{6}$$

- Attention-based aggregation:

$$\text{Aggregate}_{\text{attn}}(\mathcal{I}_j) = \sum_i a_i z_i, \quad a_i = \text{softmax}\left(\frac{q^\top z_i}{\sqrt{d}}\right), \quad q = \frac{1}{|\mathcal{I}_j|} \sum_{z \in \mathcal{I}_j} z \tag{7}$$

Each node includes a residual connection parameterized by a learnable scalar ρ_j, such that the final node output is computed as

$$x_{\text{res}}^{(j)} = \sigma(\rho_j) \cdot x^{(j)} + (1 - \sigma(\rho_j)) \cdot x^{(j-1)}, \tag{8}$$

where σ denotes the sigmoid function. The cell input undergoes a projection and normalization step:

$$x^{(0)} = \text{GELU}(\text{RMSNorm}(W_{\text{in}} x_{\text{input}})), \tag{9}$$

and the final output of the cell is obtained by blending the last node with the input:

$$x_{\text{output}} = \sigma(\rho_0) \cdot x^{(N-1)} + (1 - \sigma(\rho_0)) \cdot x^{(0)}, \tag{10}$$

here, RMSNorm stands for Root Mean Square Layer Normalization.

A progressive search strategy is employed, in which the operation space is expanded over discrete stages (basic, intermediate, advanced), controlling the complexity of available operations over time. The effective number of architecture parameters grows as

$$|\Theta_{\text{arch}}| = \sum_{(i,j) \in E} |\mathcal{O}_{\text{active}}^{(i,j)}|, \tag{11}$$

with $\mathcal{O}_{\text{active}}^{(i,j)}$ denoting the active set of operations at each edge during the current stage.

In the DARTS cell, a forward pass is used as implemented in Algorithm 1. The algorithm defines an advanced encoder-decoder structure fed by differentiable architecture search to predict future time sequences. It begins by incorporating the input time series with positional information and processing it through a stack of hierarchical DARTS cells, each defined by architecture parameters. The outputs of these cells are adaptively fused through a learnable gate mechanism, producing a refined latent representation. This representation is then passed through a mixed encoder that generates hidden and contextual states, used to initialize the decoder.

The autoregressive decoder uses a cross-attention bridge to dynamically integrate encoder information at each time step. It uses a combination of residual connections and a controlled teacher forcing mechanism to balance learning from ground truth and self-generated predictions. This design allows the model to flexibly adapt its structure through learned architecture weights, enabling efficient and accurate predictions across diverse time series domains.

3.2. Operation space and memory-efficient design

The operation space comprises ten specialized components designed for time series modeling, where all operations inherit from a memory-efficient base class with lazy initialization. This design pattern defers parameter allocation until first use, significantly reducing memory overhead during neural architecture search. The lazy initialization mechanism is formalized as:

$$\text{MemoryEfficientOp}(x) = \begin{cases} \text{lazy\_init}(x); \text{forward}(x) & \text{if not initialized} \\ \text{forward}(x) & \text{otherwise} \end{cases} \tag{12}$$

where $x \in \mathbb{R}^{B \times L \times d_{\text{in}}}$ represents the input tensor with batch size B, sequence length L, and input dimension d_{in}. The operation search space $\mathcal{O}$ contains ten distinct operations, each targeting specific aspects of temporal pattern modeling. All operations transform input tensors from $\mathbb{R}^{B \times L \times d_{\text{in}}}$ to $\mathbb{R}^{B \times L \times d_{\text{out}}}$, where d_{out} denotes the output dimension.

Fig. 1 presents the DARTSCell diagram, where different operations can be used to build the complete architecture. The proposed method searches for the optimal combinations of each operation, ensuring the

Algorithm 1: DARTS forward pass for time series forecasting.

Input : Historical sequence $x \in \mathbb{R}^{B\times L\times d}$; Future targets $y \in \mathbb{R}^{B\times H\times d}$ (optional)
Input : Teacher forcing ratio $\rho \in [0,1]$; Architecture parameters $\alpha = \{\alpha^{(\ell)}, \alpha_{\text{enc}}, \alpha_{\text{dec}}\}$
Output: Forecasted sequence $\hat{y} \in \mathbb{R}^{B\times H\times d}$

```
// Input Embedding and Positional Encoding
x_emb ← InputEmbedding(x) + PositionalEncoding(L, d);
// Hierarchical DARTS Cell Processing
h^(0) ← x_emb, F ← ∅ ;    // Initialize features and cell outputs
for ℓ = 1 to N_cells do
    h^(ℓ) ← DARTSCell(h^(ℓ−1), α^(ℓ)) ;          // Differentiable cell operation
    f^(ℓ) ← CellProjection_ℓ(h^(ℓ)) · Scale_ℓ(α^(ℓ)) ;          // Weighted projection
    F ← F ∪ {f^(ℓ)};
    h^(ℓ) ← LayerNorm(h^(ℓ) + Residual(h^(ℓ−1))) ;          // Residual connection
end
// Adaptive Feature Fusion
w_cell ← Softmax(α_cell/τ) ;          // Temperature-controlled cell weights
f_combined ← Σ_{ℓ=1}^{N_cells} w_{cell,ℓ} · f^(ℓ);
g ← σ(GateFusion([f_combined; x_emb])) ;          // Learnable gating
z ← g ⊙ f_combined + (1 − g) ⊙ x_emb ;          // Gated combination
// Mixed Encoder with Architecture Selection
(h_enc, c_enc, s_enc) ← MixedEncoder(z, α_enc);
// Autoregressive Decoding with Cross-Attention Bridge
Ŷ ← [], h_dec ← s_enc, u_0 ← x[:, −1 :, :] ;          // Initialize decoder state
for t = 1 to H do
    // Mixed decoder processing
    (o_t, h_dec) ← MixedDecoder(u_{t−1}, c_enc, h_dec, α_dec);
    // Cross-attention bridge for information flow
    õ_t ← AttentionBridge(o_t, h_enc, c_enc);
    // Output processing with residual connections
    r_t ← LayerNorm(õ_t + o_t) ;          // Attention residual
    p_t ← MLP(r_t) + r_t ;          // Feed-forward residual
    ŷ_t ← OutputProjection(p_t) ;          // Final prediction
    Ŷ ← Ŷ ∪ {ŷ_t};
    // Teacher forcing decision for next input
    if training and y ≠ None and Bernoulli(ρ) = 1 then
        u_t ← y[:, t − 1 : t, :] ;          // Ground truth input
    else
        u_t ← ŷ_t ;          // Predicted input
    end
end
return Concat(Ŷ, dim = 1) ;          // Stack predictions along time dimension
```

best design. Here, Identity, Temporal Convolution (TimeConv), Temporal Convolutional Network (TCN), Residual Multi-Layer Perceptron (ResidualMLP), Frequency-Domain Fourier, Multi-Scale Wavelet, Enhanced ConvMixer, Gated Residual Network (GRN), Multi-Scale Convolution (MultiScaleConv), Pyramid Convolution (PyramidConv), and Normalization operations are considered.

The operations of the DARTSCell diagram have the following goals:

Identity Operation: The Identity operation provides a parameter-efficient skip connection that preserves temporal structure while enabling dimension transformation when necessary:

$$\text{Identity}(x) = \begin{cases} x & \text{if } d_{\text{in}} = d_{\text{out}} \\ W_{\text{proj}} x & \text{otherwise} \end{cases} \tag{13}$$

where $W_{\text{proj}} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ is a learnable projection matrix. This operation serves as a crucial baseline and prevents architecture collapse by maintaining gradient flow paths throughout the search process.

Temporal Convolution (TimeConv): The TimeConv operation employs depthwise separable convolutions optimized for temporal sequence processing. The operation decomposes standard convolution into channel-wise and cross-channel components:

$$\text{TimeConv}(x) = \text{RMSNorm}(\text{Pointwise}(\text{Depthwise}(x^T))^T + \text{Residual}(x)) \tag{14}$$

$$\text{Depthwise}(x) = \text{Conv1D}_{\text{groups}=d_{\text{in}}}(x, K = 3, \text{padding} = 2) \tag{15}$$

$$\text{Pointwise}(x) = \text{Conv1D}_{K=1}(x) \tag{16}$$

where K denotes the kernel size, groups $= d_{\text{in}}$ indicates depthwise convolution with each input channel processed independently, and causal padding ensures temporal ordering preservation. The depthwise convolution captures local temporal patterns within each feature channel independently, while pointwise convolution facilitates information exchange across channels.

Temporal Convolutional Network (TCN): TCN extends TimeConv with dilated convolutions to capture hierarchical temporal dependencies across multiple time scales:

$$\text{TCN}(x) = \text{RMSNorm}(\text{Pointwise}(\text{Depthwise}_{\text{dil}}(x^T, d))^T + \text{Residual}(x)) \tag{17}$$

$$\text{Depthwise}_{\text{dil}}(x, d) = \text{Conv1D}_{\text{groups}=d_{\text{in}}}(x, K = 3, \text{dilation} = d) \tag{18}$$

where the dilation rate $d \in \{1, 2, 4\}$ controls the receptive field size, enabling the network to capture long-range dependencies without substantially increasing parameter count. Higher dilation rates allow the convolution to skip input positions, effectively expanding the temporal receptive field.

Residual Multi-Layer Perceptron (ResidualMLP): ResidualMLP applies position-wise transformations with residual connections to model complex non-linear relationships within temporal features:

$$\text{ResidualMLP}(x) = \text{RMSNorm}(\text{MLP}(x) + \text{Residual}(x)) \tag{19}$$

$$\text{MLP}(x) = W_2 \cdot \text{GELU}(\text{Dropout}(W_1 x + b_1)) + b_2 \tag{20}$$

$$\text{Residual}(x) = \begin{cases} x & \text{if } d_{\text{in}} = d_{\text{out}} \\ W_{\text{res}} x & \text{otherwise} \end{cases} \tag{21}$$

where $W_1 \in \mathbb{R}^{d_{\text{hidden}}\times d_{\text{in}}}$, $W_2 \in \mathbb{R}^{d_{\text{out}}\times d_{\text{hidden}}}$, $b_1 \in \mathbb{R}^{d_{\text{hidden}}}$, $b_2 \in \mathbb{R}^{d_{\text{out}}}$ are learnable parameters, and $W_{\text{res}} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ is the residual projection matrix. The expansion factor typically sets the hidden dimension as $d_{\text{hidden}} = 2.67 \times d_{\text{out}}$, balancing expressivity with computational efficiency.

Frequency-Domain Fourier Operation: The Fourier operation processes temporal sequences in the frequency domain using learnable frequency component weighting:

$$\text{Fourier}(x) = \text{OutputProj}(\text{FreqProj}(\mathcal{F}_{\text{weighted}}(x))) \tag{22}$$

$$\mathcal{F}_{\text{weighted}}(x) = \sum_{k=1}^{K} w_k \cdot [\text{Real}(\text{rfft}(x)_k), \text{Imag}(\text{rfft}(x)_k)] \tag{23}$$

$$w_k = \text{softmax}(\beta_k), \quad k = 1, \ldots, K \tag{24}$$

$$\text{FreqProj}(f) = \text{GELU}(W_f \cdot f + b_f) \tag{25}$$

where $K = \min(\lfloor L/2 \rfloor + 1, 32)$ represents the number of frequency components retained, $\text{rfft}(\cdot)$ denotes the real-valued fast Fourier transform, $\beta_k \in \mathbb{R}^K$ are learnable frequency importance parameters, $W_f \in \mathbb{R}^{d_{\text{freq}}\times 2K}$ and $b_f \in \mathbb{R}^{d_{\text{freq}}}$ are frequency projection parameters, and $\text{OutputProj} : \mathbb{R}^{d_{\text{freq}}} \to \mathbb{R}^{d_{\text{out}}}$ is the final output projection. The learnable weights w_k allow the model to focus on relevant frequency bands adaptively.

Multi-Scale Wavelet Operation: The Wavelet operation captures temporal features across multiple scales using dilated convolutions that mimic wavelet decomposition:

$$\text{Wavelet}(x) = \text{RMSNorm}(\text{Fusion}([\text{WaveletScale}_d(x) \mid d \in \{1, 2, 4\}])) \tag{26}$$

$$\text{WaveletScale}_d(x) = \text{Dropout}(\text{GELU}(\text{BatchNorm}(\text{Pointwise}(\text{Depthwise}_d(x^T))))^T \tag{27}$$

$$\text{Fusion}(f_1, f_2, f_3) = \text{Conv1D}(\text{Concat}([f_1, f_2, f_3]), K = 1) \tag{28}$$

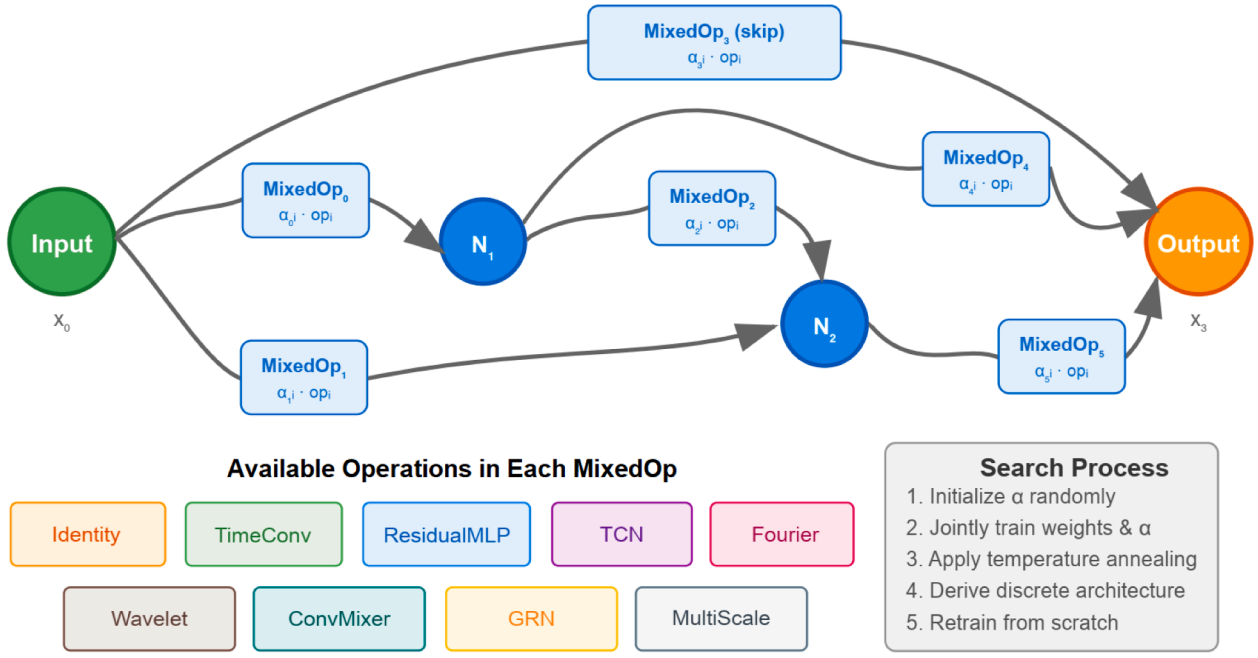


Fig. 1. DARTSCell diagram.

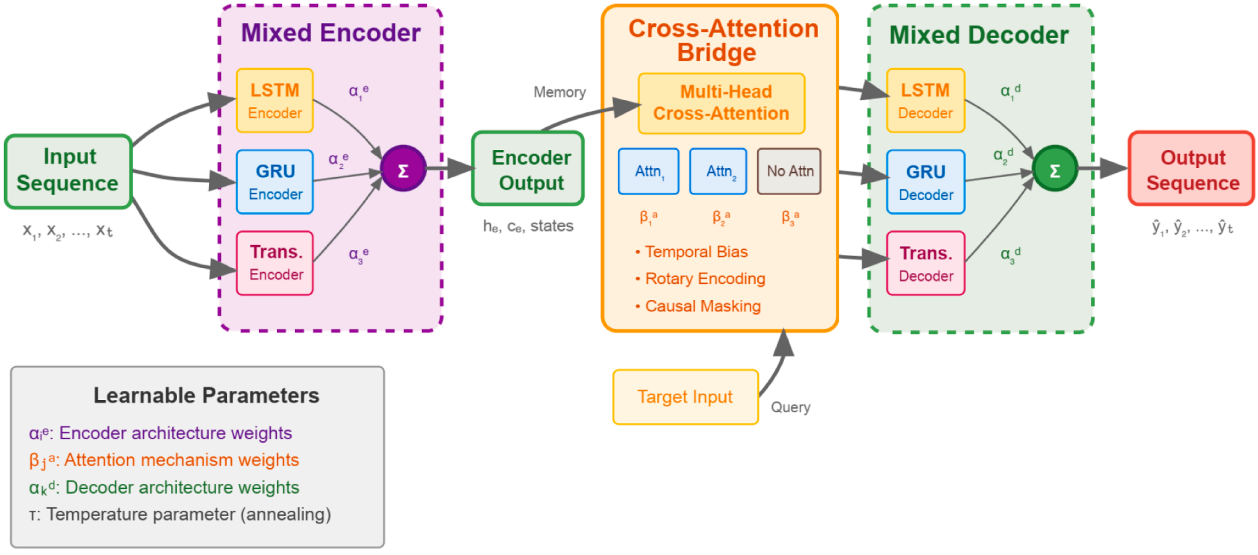


Fig. 2. Overview of the mixed encoder-decoder architecture showing parallel processing.

where $\text{WaveletScale}_d(x)$ processes the input with dilation rate d, $\text{Concat}(\cdot)$ denotes concatenation along the feature dimension, and $\text{Fusion}(\cdot)$ combines multi-scale features through a 1×1 convolution [59]. Each scale captures both fine-grained ($d = 1$) and coarse ($d = 4$) temporal patterns, with adaptive average pooling ensuring consistent output dimensions.

ConvMixer Operation: ConvMixer combines depthwise and pointwise convolutions with inner residual connections to enhance feature mixing:

$$\text{ConvMixer}(x) = \text{RMSNorm}(\text{PointwiseMix}(\text{DepthwiseMix}(x)) + \text{Residual}(x)) \quad (29)$$

$$\text{DepthwiseMix}(x) = \text{GELU}(\text{BatchNorm}(\text{Depthwise}(W_{\text{pre}} x^T, K = 9)))^T \quad (30)$$

$$\text{PointwiseMix}(x) = \text{Pointwise}(x^T + \text{Depthwise}(x^T, K = 9))^T \quad (31)$$

where $W_{\text{pre}} \in \mathbb{R}^{d_{\text{mix}} \times d_{\text{in}}}$ is a pre-processing projection matrix, and $K = 9$ denotes the large kernel size in depthwise convolution. The large kernel enables modeling of extended temporal contexts, while the inner residual connection ($x^T + \text{Depthwise}(x^T, K = 9)$) preserves information flow.

Gated Residual Network (GRN): GRN implements sophisticated gating mechanisms to control information flow and feature selection:

$$\text{GRN}(x) = \text{RMSNorm}(\text{Gate}(h) \odot \text{Transform}(h) + \text{Residual}(x)) \quad (32)$$

$$h = \text{GELU}(W_1 x + b_1) \quad (33)$$

$$\text{Gate}(h) = \sigma(W_g h + b_g) \quad (34)$$

$$\text{Transform}(h) = W_2\, h + b_2 \quad (35)$$

where $h \in \mathbb{R}^{B \times L \times d_{\text{hidden}}}$ is the hidden representation, $W_1 \in \mathbb{R}^{d_{\text{hidden}} \times d_{\text{in}}}$, $W_g \in \mathbb{R}^{d_{\text{out}} \times d_{\text{hidden}}}$, $W_2 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{hidden}}}$ are weight matrices, b_1, b_g, b_2 are bias vectors, $\sigma(\cdot)$ is the sigmoid activation function, and $\odot$ denotes element-wise multiplication. The gating mechanism enables adaptive

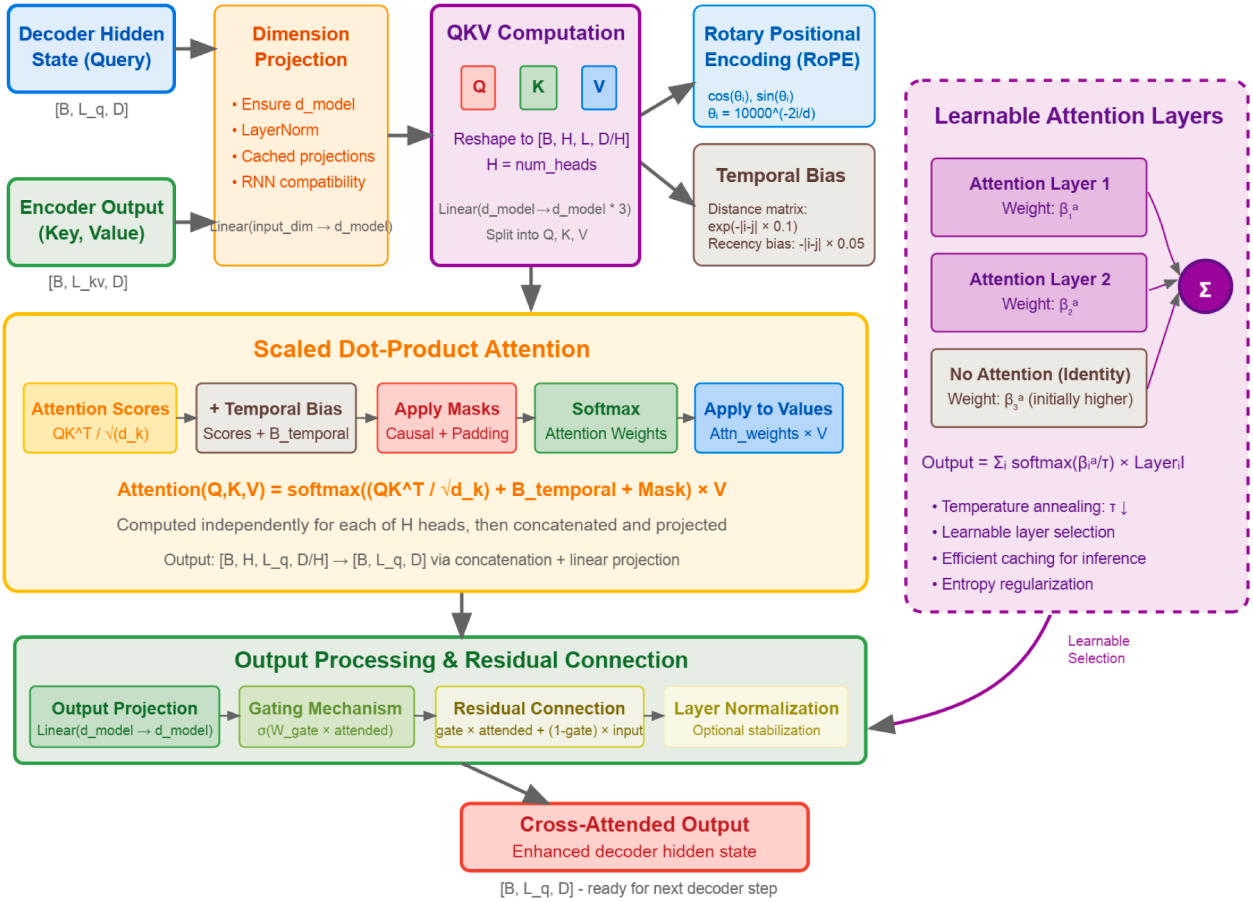


Fig. 3. Cross-attention bridge mechanism.

feature selection, allowing the network to emphasize relevant temporal patterns while suppressing noise.

Multi-Scale Convolution (MultiScaleConv): This operation processes input across multiple kernel sizes simultaneously and combines results through learned attention:

$$\text{MultiScaleConv}(x) = \text{RMSNorm}(\text{AF}([\text{ScaleConv}_k(x) \mid k \in \{1,3,5,7\}]) + \text{Residual}(x)) \tag{36}$$

$$\text{ScaleConv}_k(x) = \text{GELU}(\text{BatchNorm}(\text{Pointwise}(\text{Depthwise}(x^T, K=k)))^T \tag{37}$$

$$\text{AF}(f_1, f_2, f_3, f_4) = \sum_{i=1}^{4} \alpha_i \cdot f_i \tag{38}$$

$$\alpha_i = \text{softmax}(\text{AttentionNet}(\text{Concat}([f_1, f_2, f_3, f_4])))_i \tag{39}$$

where AF is the attention fusion, $\text{ScaleConv}_k(x)$ applies convolution with kernel size k, $\text{AttentionNet} : \mathbb{R}^{4d_{\text{out}}} \to \mathbb{R}^4$ is a learned attention network, and α_i represents the attention weight for scale i. The attention mechanism learns to weight different scales based on input characteristics, enabling adaptive multi-resolution processing.

Pyramid Convolution (PyramidConv): PyramidConv implements hierarchical feature extraction through progressive downsampling and upsampling with skip connections:

$$\text{PyramidConv}(x) = \text{RMSNorm}(\text{Upsample}(\text{Downsample}(x)) + \text{Residual}(x)) \tag{40}$$

$$\text{Downsample}(x) = \text{EncodeLevel}_L(\cdots(\text{EncodeLevel}_1(x))\cdots) \tag{41}$$

$$\text{Upsample}(x) = \text{DecodeLevel}_1(\cdots(\text{DecodeLevel}_L(x))\cdots) \tag{42}$$

$$\text{EncodeLevel}_i(x) = \text{Conv1D}(x, K=3, \text{stride}=2) \tag{43}$$

$$\text{DecodeLevel}_i(x) = \text{ConvTranspose1D}(x + \text{skip}_i, K=3, \text{stride}=2) \tag{44}$$

where L denotes the number of pyramid levels, EncodeLevel_i applies strided convolution for 2× downsampling, DecodeLevel_i uses transposed convolution for 2× upsampling, and skip_i represents skip connections from corresponding encoding levels. This hierarchical approach captures multi-resolution temporal patterns effectively.

Normalization: Each operation incorporates Root Mean Square Layer Normalization (RMSNorm) for stable training:

$$\text{RMSNorm}(x) = \frac{x}{\sqrt{\frac{1}{d_{\text{out}}}\sum_{i=1}^{d_{\text{out}}} x_i^2 + \epsilon}} \odot \gamma \tag{45}$$

where $\gamma \in \mathbb{R}^{d_{\text{out}}}$ represents learnable scaling parameters, $\epsilon = 10^{-8}$ ensures numerical stability, and the RMS is computed across the feature dimension. This normalization scheme provides computational efficiency compared to LayerNorm while maintaining comparable performance in time series modeling tasks.

3.3. Mixed encoder-decoder architecture components

The framework introduces a mixed encoder-decoder paradigm that enables dynamic architecture selection at the sequence modeling level. Fig. 2 presents this architecture, wherein the parallel processing uses LSTM [60], GRU [61], and Transformer components with learnable weight combination [62]. This approach addresses the fundamental limitation that different temporal patterns may benefit from different sequence modeling approaches. Rather than committing to a single architecture type, the framework simultaneously maintains and optimizes multiple encoder and decoder architectures, learning to weight their contributions based on input characteristics and forecasting requirements.

3.3.1. Mixed encoder

The mixed encoder component maintains three distinct encoder architectures, each specialized for different aspects of temporal modeling: LSTM networks for capturing long-term dependencies with explicit gating mechanisms [63], GRU providing computational efficiency with simplified gating [64], and Transformer encoders [65] leveraging self-attention mechanisms for parallel processing and global context modeling [66].

Let $\mathcal{E} = \{\text{LSTM}_{\text{enc}}, \text{GRU}_{\text{enc}}, \text{Transformer}_{\text{enc}}\}$ represent the set of encoder architectures. For an input sequence $x \in \mathbb{R}^{B\times L\times d_{\text{in}}}$ with batch size B, sequence length L, and input dimension d_{in}, the mixed encoder output is computed as:

$$h_{\text{enc}} = \sum_{i\in\{1,2,3\}} w_i^{\text{enc}} \cdot \mathcal{E}_i(x) \tag{46}$$

where $\mathcal{E}_1 = \text{LSTM}_{\text{enc}}$, $\mathcal{E}_2 = \text{GRU}_{\text{enc}}$, $\mathcal{E}_3 = \text{Transformer}_{\text{enc}}$, and each encoder produces hidden representations $h_{\text{enc}} \in \mathbb{R}^{B\times L\times d_{\text{latent}}}$ with latent dimension d_{latent}.

The learnable encoder weights are derived through temperature-controlled Gumbel softmax to enable differentiable architecture selection:

$$w_i^{\text{enc}} = \frac{\exp((\alpha_i^{\text{enc}} + g_i)/\tau)}{\sum_{j=1}^{3}\exp((\alpha_j^{\text{enc}} + g_j)/\tau)} \tag{47}$$

where $\alpha_i^{\text{enc}} \in \mathbb{R}$ are learnable architecture parameters, $g_i \sim \text{Gumbel}(0,1)$ are Gumbel noise samples, and $\tau > 0$ is the temperature parameter controlling the sharpness of the distribution.

Individual Encoder Specifications: Each encoder architecture implements distinct computational patterns:

The LSTM encoder processes sequences through gated memory cells:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{48}$$

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{49}$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{50}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{51}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{52}$$

$$h_t = o_t \odot \tanh(c_t) \tag{53}$$

where $f_t, i_t, o_t \in \mathbb{R}^{d_{\text{latent}}}$ are forget, input, and output gates, $c_t \in \mathbb{R}^{d_{\text{latent}}}$ is the cell state, $W_f, W_i, W_c, W_o \in \mathbb{R}^{d_{\text{latent}}\times(d_{\text{latent}}+d_{\text{in}})}$ are weight matrices, $b_f, b_i, b_c, b_o \in \mathbb{R}^{d_{\text{latent}}}$ are bias vectors, and $\sigma(\cdot)$ denotes the sigmoid function [67].

The GRU encoder simplifies the gating mechanism:

$$r_t = \sigma(W_r[h_{t-1}, x_t] + b_r) \tag{54}$$

$$z_t = \sigma(W_z[h_{t-1}, x_t] + b_z) \tag{55}$$

$$\tilde{h}_t = \tanh(W_h[r_t \odot h_{t-1}, x_t] + b_h) \tag{56}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{57}$$

where $r_t, z_t \in \mathbb{R}^{d_{\text{latent}}}$ are reset and update gates, and W_r, W_z, W_h and b_r, b_z, b_h follow similar dimensionality as LSTM parameters [68].

The Transformer encoder employs multi-head self-attention:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)W^O \tag{58}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{59}$$

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{60}$$

where H is the number of attention heads, $d_k = d_{\text{latent}}/H$ is the dimension per head, $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d_{\text{latent}}\times d_k}$ are projection matrices for head i, and $W^O \in \mathbb{R}^{d_{\text{latent}}\times d_{\text{latent}}}$ is the output projection matrix [65].

3.3.2. Mixed decoder

The mixed decoder employs the same three architecture types for autoregressive sequence generation, with each decoder specialized for different aspects of temporal prediction. For target sequence $y \in \mathbb{R}^{B\times L_{\text{out}}\times d_{\text{out}}}$ with output length L_{out} and output dimension d_{out}, the mixed decoder computation is:

$$h_{\text{dec}} = \sum_{i\in\{1,2,3\}} w_i^{\text{dec}} \cdot \mathcal{D}_i(y, h_{\text{enc}}, s_{t-1}) \tag{61}$$

where $\mathcal{D} = \{\text{LSTM}_{\text{dec}}, \text{GRU}_{\text{dec}}, \text{Transformer}_{\text{dec}}\}$ represents the decoder set, h_{enc} is the encoder output, and s_{t-1} denotes the previous decoder state (hidden state for RNNs or previous output for Transformers).

The decoder weights follow an identical temperature-controlled Gumbel softmax:

$$w_i^{\text{dec}} = \frac{\exp((\alpha_i^{\text{dec}} + g_i)/\tau)}{\sum_{j=1}^{3}\exp((\alpha_j^{\text{dec}} + g_j)/\tau)} \tag{62}$$

where $\alpha_i^{\text{dec}} \in \mathbb{R}$ are learnable decoder architecture parameters that evolve independently from encoder selections, enabling asymmetric encoder-decoder combinations.

3.3.3. Normalization and compatibility

To ensure compatibility between different architectures, the framework implements an architecture normalization mechanism. Each encoder and decoder output is processed through architecture-specific projection layers:

$$\text{Normalize}(h, \text{arch\_type}) = \begin{cases} W_{\text{rnn}}h + b_{\text{rnn}} & \text{if arch\_type} \in \{\text{LSTM}, \text{GRU}\} \\ W_{\text{trans}}h + b_{\text{trans}} & \text{if arch\_type} = \text{Transformer} \end{cases} \tag{63}$$

where $W_{\text{rnn}}, W_{\text{trans}} \in \mathbb{R}^{d_{\text{latent}}\times d_{\text{latent}}}$ are learnable projection matrices and $b_{\text{rnn}}, b_{\text{trans}} \in \mathbb{R}^{d_{\text{latent}}}$ are bias vectors.

Hidden states from different architectures are normalized through Layer Normalization before blending:

$$\text{StateNorm}(s) = \frac{s - \mu}{\sqrt{\sigma^2 + \epsilon}} \odot \gamma + \beta \tag{64}$$

where μ and σ^2 are the mean and variance of s, $\epsilon = 10^{-6}$ ensures numerical stability, and $\gamma, \beta \in \mathbb{R}^{d_{\text{latent}}}$ are learnable parameters.

3.3.4. Temperature annealing and architecture selection

The temperature parameter τ controls the exploration-exploitation balance during training through an adaptive annealing schedule with multiple strategies. The scheduler implements a two-phase approach: an initial warmup period followed by progressive temperature decay.

During the warmup phase (first $e_w = 5$ epochs), the temperature is maintained at the initial value $\tau_0 = 2.0$ to ensure sufficient exploration of the architecture space before architecture weights begin to converge [69,70]. After warmup, the temperature evolves according to the normalized training progress $p = \frac{e-e_w}{E-e_w}$, where e is the current epoch, e_w is the warmup period, and E is the total number of training epochs. The scheduler supports multiple annealing strategies:

Cosine annealing [71]:

$$\tau_e = \tau_{\min} + \frac{\tau_0 - \tau_{\min}}{2}(1 + \cos(\pi p)) \tag{65}$$

Exponential decay:

$$\tau_e = \tau_0 \cdot \exp\left(\frac{\ln(\tau_{\min}/\tau_0)}{E - e_w} \cdot (e - e_w)\right) \tag{66}$$

Linear decay:

$$\tau_e = \tau_0 - (\tau_0 - \tau_{\min}) \cdot p \tag{67}$$

Step decay:

$$\tau_e = \begin{cases} \tau_0 & \text{if } p < 0.3 \\ 0.5\tau_0 & \text{if } 0.3 \le p < 0.7 \\ \tau_{\min} & \text{if } p \ge 0.7 \end{cases} \tag{68}$$

where $\tau_{\min} = 0.1$ is the minimum temperature threshold enforced across all strategies [72,73].

High temperatures ($\tau > 1$) encourage uniform weight distributions across architecture components, promoting exploration of diverse combinations [74]. Low temperatures ($\tau < 1$) induce sharper distributions, facilitating convergence toward specific architectures [75]. The cosine schedule provides smooth, gradual transitions and is commonly used as the default strategy due to its stable convergence properties and ability to escape local minima [71].

During final architecture derivation, the framework can either maintain the mixed approach with learned weights or derive discrete architectures by selecting dominant components:

$$\text{Encoder}^* = \arg\max_i w_i^{\text{enc}} \tag{69}$$

$$\text{Decoder}^* = \arg\max_i w_i^{\text{dec}} \tag{70}$$

3.4. Cross-attention bridge mechanism

Fig. 3 presents the cross-attention bridge mechanism showing multi-head attention with temporal bias and adaptive gating between encoder and decoder representations. The framework incorporates a unified cross-attention bridge that enables learnable information flow between encoder and decoder representations. This mechanism addresses limitations in traditional encoder-decoder architectures where information transfer is constrained to simple hidden state passing or basic attention mechanisms.

3.4.1. Multi-head cross-attention with temporal bias

The core attention mechanism extends standard multi-head attention with learnable temporal bias to capture sequence-dependent relationships specific to time series forecasting:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}} + B_{\text{temporal}}\right)V \tag{71}$$

where $Q \in \mathbb{R}^{B\times L_q\times d_k}$ represents decoder queries derived from current decoder states, $K, V \in \mathbb{R}^{B\times L_{\text{enc}}\times d_k}$ represent encoder keys and values, and $B_{\text{temporal}} \in \mathbb{R}^{L_q\times L_{\text{enc}}}$ constitutes a learnable temporal bias matrix.

The temporal bias matrix captures task-specific temporal relationships beyond standard positional encodings:

$$B_{\text{temporal}}[i, j] = \alpha_{\text{dist}} \cdot \exp\left(-\frac{|i-j|^2}{2\sigma_{\text{dist}}^2}\right) + \beta_{\text{recency}} \cdot \mathbb{I}[j \le i] \tag{72}$$

where $\alpha_{\text{dist}}, \beta_{\text{recency}} \in \mathbb{R}$ are learnable parameters, σ_{dist} controls the temporal distance sensitivity, and $\mathbb{I}[\cdot]$ is the indicator function providing causal bias for forecasting.

For multi-head attention with H heads, each head $h \in \{1, \ldots, H\}$ maintains its own temporal bias matrix $B_h \in \mathbb{R}^{L_q\times L_{\text{enc}}}$, enabling specialization for different temporal relationships:

$$\text{head}_h = \text{Attention}(QW_h^Q, KW_h^K, VW_h^V, B_h) \tag{73}$$

3.4.2. Learnable attention configuration selection

The framework implements an attention bridge selection mechanism that determines optimal attention layer configurations. Rather than fixing the attention architecture, the system learns to weight different configurations, including direct bypass:

$$w_i^{\text{attn}} = \frac{\exp((\alpha_i^{\text{attn}} + g_i)/\tau)}{\sum_{j=0}^{L_{\max}} \exp((\alpha_j^{\text{attn}} + g_j)/\tau)} \tag{74}$$

where $L_{\max}$ denotes the maximum number of attention layers, $\alpha_i^{\text{attn}} \in \mathbb{R}$ are learnable architecture parameters for attention layer i, and w_0^{attn} corresponds to the "no attention" option.

The attention bridge output combines multiple configurations:

$$\text{AttentionBridge}(Q, K, V) = w_0^{\text{attn}} \cdot Q + \sum_{i=1}^{L_{\max}} w_i^{\text{attn}} \cdot \text{AttentionLayer}_i(Q, K, V) \tag{75}$$

where the first term provides direct bypass when attention is not beneficial.

3.4.3. Adaptive architecture integration

The cross-attention bridge adapts its operation based on the active encoder-decoder combination through mode-specific query, key, and value generation:

For RNN-style architectures operating in sequential mode:

$$Q_t = W_Q^{\text{rnn}} h_{\text{dec},t} + b_Q^{\text{rnn}} \tag{76}$$

$$K = W_K^{\text{rnn}} H_{\text{enc}} + b_K^{\text{rnn}}, \quad V = W_V^{\text{rnn}} H_{\text{enc}} + b_V^{\text{rnn}} \tag{77}$$

where $h_{\text{dec},t} \in \mathbb{R}^{B\times 1\times d_{\text{latent}}}$ is the decoder hidden state at timestep t, $H_{\text{enc}} \in \mathbb{R}^{B\times L\times d_{\text{latent}}}$ contains encoder hidden sequences, and $W_Q^{\text{rnn}}, W_K^{\text{rnn}}, W_V^{\text{rnn}} \in \mathbb{R}^{d_k\times d_{\text{latent}}}$ are mode-specific projection matrices.

For Transformer-style architectures operating in parallel mode:

$$Q = W_Q^{\text{trans}} H_{\text{dec}} + b_Q^{\text{trans}} \tag{78}$$

$$K = W_K^{\text{trans}} H_{\text{enc}} + b_K^{\text{trans}}, \quad V = W_V^{\text{trans}} H_{\text{enc}} + b_V^{\text{trans}} \tag{79}$$

where $H_{\text{dec}} \in \mathbb{R}^{B\times L_q\times d_{\text{latent}}}$ represents the full decoder sequence.

3.4.4. Gated information fusion

The attention bridge incorporates a gating mechanism to control the balance between attention-based and direct information flow:

$$\text{Gate}(h_{\text{dec}}, h_{\text{attended}}) = \sigma(W_g[h_{\text{dec}}; h_{\text{attended}} + b_g) \tag{80}$$

$$h_{\text{output}} = g \odot h_{\text{attended}} + (1-g) \odot h_{\text{dec}} \tag{81}$$

where $g \in \mathbb{R}^{B\times L_q\times d_{\text{latent}}}$ is the gate activation, $W_g \in \mathbb{R}^{d_{\text{latent}}\times 2d_{\text{latent}}}$ is the gate projection matrix, $b_g \in \mathbb{R}^{d_{\text{latent}}}$ is the gate bias, and $[h_{\text{dec}}; h_{\text{attended}}]$ denotes concatenation.

3.4.5. Computational optimizations

The framework implements several efficiency optimizations for the attention mechanism:

Attention Dropout and Regularization:

$$\text{AttentionDropout}(\text{scores}) = \text{Dropout}\left(\text{softmax}\left(\frac{\text{scores}}{\sqrt{d_k}}\right), p_{\text{attn}}\right) \tag{82}$$

where p_{attn} is the attention dropout probability that promotes exploration during training.

Gradient Scaling: To prevent gradient explosion in attention weights, the framework applies gradient scaling:

$$\nabla_{\text{scaled}} = \min\left(1.0, \frac{C_{\text{clip}}}{\|\nabla\|_2}\right) \cdot \nabla \tag{83}$$

where $C_{\text{clip}} = 1.0$ is the clipping threshold and $\|\cdot\|_2$ denotes the L2 norm.

This cross-attention bridge mechanism enables the framework to learn optimal information flow patterns between encoder and decoder components while maintaining computational efficiency and architectural flexibility across different sequence modeling paradigms.

3.5. Zero-cost architecture evaluation metrics

The multi-fidelity framework leverages zero-cost metrics to estimate architecture performance potential without training, enabling efficient candidate screening in Phase 1. These metrics analyze properties observable at initialization and provide computational proxies for architecture quality assessment. The metrics fall into four categories based on the network characteristics they evaluate: information flow, model expressivity, optimization landscape properties, and architectural complexity.

Information Flow Metrics: These metrics assess how effectively signals propagate through network architectures, identifying potential issues such as vanishing or exploding gradients before training begins. SynFlow [76] quantifies signal preservation by examining the interaction between parameters and their gradients with respect to a synthetic

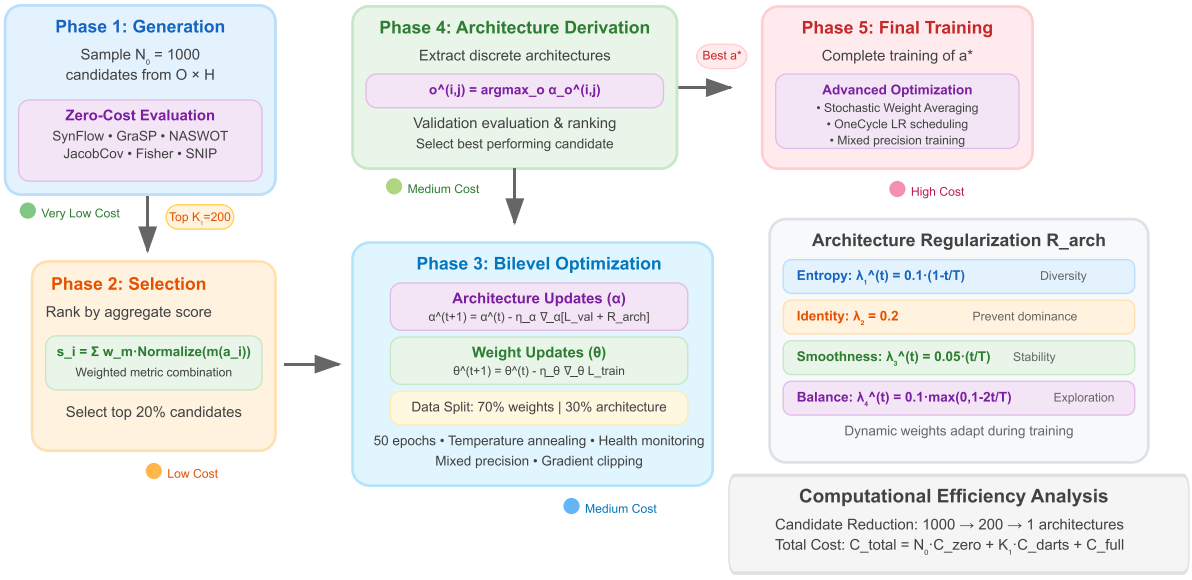


Fig. 4. Multi-fidelity search framework showing progressive candidate filtering through phases.

loss function computed on standardized dummy inputs. The metric evaluates network connectivity and gradient flow simultaneously:

$$\text{SynFlow}(\mathcal{N}_\theta) = \sum_{l=1}^{L} \sum_{i,j} \left| \theta_{i,j}^l \cdot \frac{\partial \mathcal{L}_{\text{syn}}}{\partial \theta_{i,j}^l} \right| \tag{84}$$

where $\mathcal{L}_{\text{syn}} = \sum_k z_k$ represents the sum of network outputs when processing inputs of all ones, with all parameters replaced by their absolute values to ensure positive signal flow. This synthetic setup isolates architectural properties from data-specific characteristics, enabling pure connectivity assessment. The absolute value operation prevents cancellation effects, and the product $\theta_{i,j}^l \cdot \frac{\partial \mathcal{L}_{\text{syn}}}{\partial \theta_{i,j}^l}$ captures both the parameter magnitude and its influence on the loss function.

Gradient Signal Preservation (GraSP) [77] evaluates gradient flow preservation through the relationship between parameters and their loss sensitivities, focusing on the preservation of gradient information during backpropagation:

$$\text{GraSP}(\mathcal{N}_\theta) = \sum_{i,j} \theta_{i,j} \cdot \mathcal{H}_{i,j} \tag{85}$$

where $\mathcal{H}_{i,j} = \frac{\partial^2 \mathcal{L}}{\partial \theta_{i,j} \partial \mathcal{L}} \approx \frac{\partial}{\partial \theta_{i,j}} \|\nabla_\theta \mathcal{L}\|$ approximates the Hessian-gradient product that characterizes how parameter changes affect gradient magnitude. This metric identifies architectures that maintain stable gradient flow, which is particularly important for deep networks and complex temporal dependencies in time series models.

Model Expressivity Metrics: These metrics assess the architectural capacity to represent complex functions and distinguish between different input patterns, providing insights into the model's representational power. Neural Architecture Search Without Training (NASWOT) [78] analyzes activation pattern diversity across different input samples to estimate the network's capacity for complex function representation:

$$\text{NASWOT}(\mathcal{N}_\theta) = \sum_{l=1}^{L} \text{rank}(\mathbf{K}^l) + \lambda(\mathbf{K}^l)_{\max} \tag{86}$$

where $\mathbf{K}^l = \text{sign}(\mathbf{A}^l) \cdot \text{sign}(\mathbf{A}^l)^T$ represents the kernel matrix constructed from binarized activations $\mathbf{A}^l \in \mathbb{R}^{B \times H^l}$ at layer l, with B representing batch size and H^l the number of hidden units. The rank of $\mathbf{K}^l$ quantifies the linear independence of activation patterns, while $\lambda(\mathbf{K}^l)_{\max}$ denotes the largest eigenvalue providing spectral information. Higher ranks indicate greater capacity to represent diverse input-output mappings. The sign operation creates binary activation patterns that capture the essential switching behavior of neural networks while remaining computationally tractable.

JacobCov [78] measures the covariance of Jacobian matrices to assess how diversely the network responds to input variations, providing insights into the model's sensitivity and discriminative capacity:

$$\text{JacobCov}(\mathcal{N}_\theta) = \frac{1}{n} \sum_{i=1}^{n} H\left(\lambda\left(\mathbf{J}_i \mathbf{J}_i^T\right)\right) \tag{87}$$

where $\mathbf{J}_{i,j,k} = \frac{\partial \mathcal{N}_\theta(\mathbf{X})_{i,j}}{\partial \mathbf{X}_{i,k}}$ represents the Jacobian matrix capturing output sensitivity to input changes for sample i, output dimension j, and input feature k. The function $H(\lambda(\cdot))$ computes the entropy of the eigenvalue spectrum λ of the covariance matrix $\mathbf{J}_i \mathbf{J}_i^T$, where higher entropy indicates more balanced sensitivity across different input dimensions. For computational efficiency, this is approximated using Hutchinson's method:

$$\text{JacobCov}(\mathcal{N}_\theta) \approx \log\left(\mathbb{E}_v\left[\left\|\nabla_{\mathbf{x}}\left(\mathbf{f}^T \mathbf{v}\right)\right\|^2\right] + \epsilon\right) \tag{88}$$

where $\mathbf{v}$ is a random vector and the expectation approximates the trace of the Jacobian covariance matrix [79].

Zen-NAS [80] evaluates network expressivity through signal-to-noise ratio analysis of ReLU activations, measuring the network's ability to maintain meaningful signal patterns:

$$\text{Zen-NAS}(\mathcal{N}_\theta) = \frac{1}{|\mathcal{R}|} \sum_{r \in \mathcal{R}} \frac{\mu_r^2}{\sigma_r^2 + \epsilon} \tag{89}$$

where $\mathcal{R}$ represents the set of ReLU-like activation layers, μ_r and σ_r are the mean and standard deviation of activations in layer r, and ϵ is a small constant for numerical stability. Higher signal-to-noise ratios indicate better capacity for feature discrimination and representation learning.

Optimization Landscape Metrics: These metrics analyze properties related to training dynamics and optimization behavior, providing insights into the ease of training and convergence characteristics. Fisher Information [81] assesses the curvature of the loss landscape around the initialization point, providing insights into parameter sensitivity and optimization difficulty:

$$\text{Fisher}(\mathcal{N}_\theta) = \sum_{i,j} \mathbf{F}_{i,j} \tag{90}$$

where the Fisher Information Matrix is defined as $\mathbf{F} = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\nabla_\theta \log p(\mathbf{y}|\mathbf{x},\theta) \nabla_\theta \log p(\mathbf{y}|\mathbf{x},\theta)^T\right]$, characterizing the expected

squared gradient of the log-likelihood. High Fisher information indicates steep loss landscapes that may lead to optimization difficulties, while moderate values suggest more trainable architectures. For time series forecasting, this metric helps identify architectures that balance model capacity with optimization stability.

Single-shot Network Pruning (SNIP) [82] evaluates parameter importance through loss sensitivity analysis, measuring how much each parameter contributes to the loss function at initialization:

$$\text{SNIP}(\mathcal{N}_\theta) = \sum_{i,j} \left| \theta_{i,j} \cdot \frac{\partial \mathcal{L}}{\partial \theta_{i,j}} \right| \tag{91}$$

This metric quantifies the connection strength between parameters and loss function, where larger values indicate parameters that significantly impact model performance. The absolute value ensures positive contributions, and the product $\theta_{i,j} \cdot \frac{\partial \mathcal{L}}{\partial \theta_{i,j}}$ captures both parameter magnitude and gradient information. Architectures with well-distributed SNIP scores typically exhibit better trainability and parameter utilization.

Weight Conditioning [83] measures numerical stability through the condition numbers of weight matrices, identifying potential optimization difficulties due to ill-conditioned transformations:

$$\text{Weight\_Cond}(\mathcal{N}_\theta) = \frac{1}{|\mathcal{L}|} \sum_{l=1}^{|\mathcal{L}|} \kappa(\mathbf{W}^l) \tag{92}$$

where $\kappa(\mathbf{W}^l) = \frac{\sigma_{\max}(\mathbf{W}^l)}{\sigma_{\min}(\mathbf{W}^l)}$ represents the condition number of weight matrix $\mathbf{W}^l$ at layer l, computed as the ratio of maximum to minimum singular values. Lower condition numbers indicate more stable numerical transformations that facilitate gradient-based optimization, while extremely high condition numbers can lead to training instabilities and convergence difficulties.

Sensitivity Analysis [84] measures the network's responsiveness to input perturbations and parameter changes, providing insights into optimization stability:

$$\text{Sensitivity}(\mathcal{N}_\theta) = \left\| \nabla_{\mathbf{x}} \mathcal{L} \right\|_2 + \frac{1}{|\Theta|} \sum_{\theta \in \Theta} \left| \frac{\partial \mathcal{L}}{\partial \theta} \right| \tag{93}$$

where the first term captures input sensitivity through gradient norms, and the second term measures parameter sensitivity averaged across all trainable parameters Θ. This metric identifies architectures that maintain appropriate sensitivity levels for effective learning while avoiding excessive instability.

Architectural Complexity Metrics: These metrics quantify the computational and structural complexity of network architectures, providing essential constraints for practical deployment.

Parameter Count measures the total number of trainable parameters in the network:

$$\text{Params}(\mathcal{N}_\theta) = \sum_{l=1}^{L} |\theta^l| \tag{94}$$

where $|\theta^l|$ represents the number of parameters in layer l. This metric provides a direct measure of model complexity and memory requirements, serving as a constraint in resource-limited environments.

Floating Point Operations Per Second (FLOPS) estimates the computational cost of a single forward pass: $\text{FLOPS}(\mathcal{N}_\theta) = \sum_{l=1}^{L} \text{ops}^l$, where ops^l represents the number of floating-point operations required for layer l. For convolutional layers, this includes kernel operations multiplied by output spatial dimensions, while for linear layers, it encompasses matrix multiplication operations. This metric provides insights into inference time and computational resource requirements.

Adaptive Normalization and Aggregation: The final aggregate score combines all metrics through weighted summation with adaptive normalization to handle different metric scales and dynamic ranges:

$$\text{Aggregate}(\mathcal{N}_\theta) = \sum_{m \in \mathcal{M}} w_m \cdot \text{Normalize}(m(\mathcal{N}_\theta)) \tag{95}$$

Table 1
Zero-cost architecture evaluation metrics for candidate screening.

Metric	Category	Property Evaluated	Norm.
SynFlow	Info Flow	Signal preservation through gradients	Log
GraSP	Info Flow	Gradient stability and weight sensitivity	Log
NASWOT	Expressivity	Activation diversity (kernel rank + spectrum)	[0,1]
JacobCov	Expressivity	Jacobian spectrum entropy (Hutchinson)	[0,1]
Zen-NAS	Expressivity	ReLU signal-to-noise ratio	[0,1]
Fisher	Opt. Landscape	Loss curvature and sharpness	Log
SNIP	Opt. Landscape	Parameter importance via gradients	Log
Weight_Cond	Opt. Landscape	Weight matrix condition number	Log
Sensitivity	Opt. Landscape	Input/parameter gradient sensitivity	[0,1]
Params	Complexity	Total trainable parameter count	Log
FLOPS	Complexity	Floating-point operations per forward pass	Log
Aggregate	Combined	Weighted sum of all metrics	Adaptive

where the normalization function adapts to metric characteristics:

$$\text{Normalize}(x) = \begin{cases} \frac{\log(x+\epsilon) - \mu_{\log}}{\sigma_{\log}} & \text{for unbounded metrics} \\ \frac{x - x_{\min}}{x_{\max} - x_{\min}} & \text{for bounded metrics} \end{cases} \tag{96}$$

Logarithmic normalization applies to metrics with large dynamic ranges (SynFlow, SNIP, Weight_Cond, Params, FLOPS), using population statistics $\mu_{\log}$ and $\sigma_{\log}$ computed across candidate architectures. Range normalization handles bounded metrics (NASWOT, JacobCov, Zen-NAS), ensuring all metrics contribute equally to the final assessment. The weights w_m can be either uniform or learned through meta-optimization on validation datasets, enabling adaptation to specific forecasting tasks and data characteristics (see Table 1).

3.6. Multi-fidelity architecture search framework

The framework employs a multi-fidelity search strategy that efficiently allocates computational resources across candidate architectures through progressive filtering. Fig. 4 illustrates the five phases of our method, which are (1) parallel candidate generation, (2) zero-cost screening, (3) bilevel optimization training, (4) architecture derivation, and (5) final training. The search operates through these phases with increasing computational cost and decreasing candidate pool size.

Algorithm 2 presents the multi-fidelity architecture search framework. This algorithm is designed to efficiently discover high-performing neural architectures by combining low-cost heuristics and high-fidelity optimization. In Phase 1, a diverse set of candidate architectures is sampled and evaluated using zero-cost proxies, which are aggregated into a score for rapid filtering. Phase 2 selects the top-K architectures based on these scores.

In Phase 3, each candidate undergoes bilevel optimization: architecture parameters are periodically updated using validation loss while model weights are trained on the training set, with additional mechanisms such as temperature scheduling and noise injection to maintain search diversity and robustness. The best-performing discrete architecture from these trials is selected in Phase 4 based on validation performance. Finally, Phase 5 performs full training of the chosen architecture using advanced optimization techniques to produce the final model. This framework balances exploration and exploitation across fidelity levels to optimize architecture search efficiency and performance. Each of these phases is explained in this subsection in detail.

3.6.1. Phase 1: Candidate generation and zero-cost evaluation

A diverse set of N_0 candidate architectures is sampled through randomized configuration selection from the joint space of operations and hyperparameters:

$$\mathcal{A}_0 = \{\text{Sample}(\mathcal{O}, \mathcal{H}) \mid i = 1, \ldots, N_0\} \tag{97}$$

where $\mathcal{O}$ represents the operation space and $\mathcal{H}$ denotes the hyperparameter space including latent dimensions $d_{\text{latent}} \in \{64, 128, 256\}$, cell numbers $N_{\text{cells}} \in \{2, 3, 4\}$, and node configurations $N_{\text{nodes}} \in \{2, 3, 4\}$.

Algorithm 2: Multi-fidelity architecture search framework.

Input : $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}}, \mathcal{D}_{\text{test}}$; Search parameters: $N_0, K_1, T_{\text{search}}, T_{\text{final}}$
Output: Optimized architecture a^* and trained model θ^*

```
// Phase 1: Parallel Candidate Generation & Zero-Cost Screening
A_0 ← ∅;
for i = 1 to N_0 do in parallel
    a_i ← Sample(O × H) ;            // Sample from operation and hyperparameter space
    s_i ← Σ_{m∈M} w_m · Normalize(ZeroCostEval(a_i, m)) ;  // Aggregate zero-cost score
    A_0 ← A_0 ∪ {(a_i, s_i)};
end
// Phase 2: Top-K Selection
A_1 ← TopK(A_0, K_1) based on scores s_i;
// Phase 3: Bilevel Optimization Training
A_2 ← ∅;
for (a, s) ∈ A_1 do
    α, θ ← Initialize(a);
    for t = 1 to T_search do
        // Architecture parameter updates on validation set
        if t ≥ T_warmup and t mod f_arch = 0 then
            L_arch ← L_val(α, θ) + R_arch(α);
            α ← α − η_α ∇_α L_arch;
        end
        // Model weight updates on training set
        L_train ← E_{(x,y)~D_train}[ℓ(f(x; α, θ), y)];
        θ ← θ − η_θ ∇_θ L_train;
        // Dynamic regulation
        τ^(t) ← TemperatureSchedule(t, T_search);
    end
    a_derived ← DeriveArchitecture(α) ;      // Extract discrete architecture
    v ← Evaluate(a_derived, D_val);
    A_2 ← A_2 ∪ {(a_derived, v)};
end
// Phase 4: Best Architecture Selection
a* ← arg min_a {v : (a, v) ∈ A_2};
// Phase 5: Final Training with Advanced Optimization
θ* ← FinalTrain(a*, D_train, D_val, T_final);
return (a*, θ*);
```

Each candidate undergoes zero-cost metric evaluation using analytical proxies that assess architecture quality without training. The aggregate score combines multiple metrics through weighted summation:

$$s_i = \sum_{m \in \mathcal{M}} w_m \cdot \text{Normalize}(m(a_i)) \tag{98}$$

where $\mathcal{M} = \{\text{SynFlow}, \text{GraSP}, \text{NASWOT}, \text{JacobCov}, \text{Fisher}, \text{SNIP}\}$ and w_m are empirically determined weights: $w_{\text{SynFlow}} = 0.25$, $w_{\text{GraSP}} = 0.15$, $w_{\text{NASWOT}} = 0.20$, $w_{\text{JacobCov}} = 0.15$, $w_{\text{Fisher}} = 0.15$, $w_{\text{SNIP}} = 0.10$.

3.6.2. Phase 2: candidate selection

The top $K_1 = \lfloor 0.2 \times N_0 \rfloor$ candidates are retained based on their aggregate scores:

$$\mathcal{A}_1 = \{a_i \in \mathcal{A}_0 \mid \text{rank}(s_i) \leq K_1\} \tag{99}$$

3.6.3. Phase 3: bilevel optimization training

Selected candidates undergo abbreviated DARTS training using a bilevel optimization framework that alternates between architecture parameter updates on validation data and network weight optimization on training data. The bilevel problem is formulated as:

$$\min_{\alpha} \quad \mathcal{L}_{\text{val}}(\alpha, \theta^*(\alpha)) + \mathcal{R}_{\text{arch}}(\alpha) \tag{100}$$

$$\text{subject to} \quad \theta^*(\alpha) = \arg\min_{\theta} \mathcal{L}_{\text{train}}(\alpha, \theta) \tag{101}$$

where α represents architecture parameters, θ denotes network weights, and $\mathcal{R}_{\text{arch}}(\alpha)$ is the architecture regularization term.

Alternating Optimization Strategy: The framework implements alternating updates within each training epoch:

$$\alpha^{(t+1)} = \alpha^{(t)} - \eta_\alpha \nabla_\alpha \left[\mathcal{L}_{\text{val}}(\alpha^{(t)}, \theta^{(t)}) + \mathcal{R}_{\text{arch}}(\alpha^{(t)})\right] \tag{102}$$

$$\theta^{(t+1)} = \theta^{(t)} - \eta_\theta \nabla_\theta \mathcal{L}_{\text{train}}(\alpha^{(t)}, \theta^{(t)}) \tag{103}$$

with learning rates $\eta_\alpha = 3 \times 10^{-4}$ and $\eta_\theta = 1 \times 10^{-3}$.

Data Partitioning: Training data is split into disjoint sets to prevent overfitting:

$$\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{model}} \cup \mathcal{D}_{\text{arch}}, \quad |\mathcal{D}_{\text{model}}| = 0.7|\mathcal{D}_{\text{train}}|, \quad |\mathcal{D}_{\text{arch}}| = 0.3|\mathcal{D}_{\text{train}}| \tag{104}$$

Architecture Regularization: The regularization term incorporates multiple components with dynamic weighting to address distinct aspects of architecture search stability [85–87]:

$$\mathcal{R}_{\text{arch}}(\alpha) = \lambda_1^{(t)} \mathcal{L}_{\text{entropy}}(\alpha) + \lambda_2 \mathcal{L}_{\text{identity}}(\alpha) + \lambda_3^{(t)} \mathcal{L}_{\text{smooth}}(\alpha) + \lambda_4^{(t)} \mathcal{L}_{\text{balance}}(\alpha) \tag{105}$$

where the regularization components are defined as:

$$\mathcal{L}_{\text{entropy}}(\alpha) = -\sum_{e,k} p_{e,k} \log p_{e,k}, \quad \lambda_1^{(t)} = 0.1 \cdot (1 - t/T) \tag{106}$$

$$\mathcal{L}_{\text{identity}}(\alpha) = \sum_{e} \max(0, p_{\text{identity}}^{(e)} - 0.5)^2, \quad \lambda_2 = 0.2 \tag{107}$$

$$\mathcal{L}_{\text{smooth}}(\alpha) = \frac{1}{|E|} \sum_{e} \|\alpha_t^{(e)} - \alpha_{t-1}^{(e)}\|_2^2, \quad \lambda_3^{(t)} = 0.05 \cdot (t/T) \tag{108}$$

$$\mathcal{L}_{\text{balance}}(\alpha) = \text{Var}(\{\sum_{e} p_{e,k} \mid k \in \mathcal{O}\}), \quad \lambda_4^{(t)} = 0.1 \cdot \max(0, 1 - 2t/T) \tag{109}$$

where $p_{e,k}$ represents the probability of operation k on edge e, T is the total number of training epochs, and $\text{Var}(\cdot)$ denotes variance. The entropy regularization with decreasing weight $\lambda_1^{(t)}$ encourages exploration early in training before converging to sparse solutions [85,88,89]. The identity regularization with $\lambda_2 = 0.2$ mitigates the unfair advantage of parameter-free skip connections [86,87]. The smoothness regularization with increasing weight $\lambda_3^{(t)}$ promotes gradual architecture parameter evolution [90,91]. The balance regularization with early emphasis via $\lambda_4^{(t)}$ harmonizes operation selection across different edges [87,92].

Optimization Techniques: The training incorporates several stability enhancements:

- **Mixed Precision:** Automatic mixed precision with GradScaler for acceleration
- **Gradient Clipping:** Maximum norms of 3.0 for architecture parameters and 5.0 for network weights
- **Separate Optimizers:** Adam for architecture parameters ($\beta_1 = 0.5, \beta_2 = 0.999$) and AdamW for network weights (weight decay 10^{-4})
- **Learning Rate Scheduling:** Cosine annealing for architecture parameters and OneCycle for network weights

3.6.4. Phase 4: architecture derivation and validation

Discrete architectures are extracted from trained models by selecting dominant operations:

$$o^{(i,j)} = \arg\max_{o \in \mathcal{O}} \alpha_o^{(i,j)} \tag{110}$$

where $\alpha_o^{(i,j)}$ represents the weight of operation o on edge (i, j). Each derived architecture undergoes validation evaluation to determine final performance ranking.

3.6.5. Phase 5: final training

The best-performing candidate from Phase 4 receives full training with advanced optimization techniques:

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{\text{train}}(a^*, \theta; \mathcal{D}_{\text{train}}) \tag{111}$$

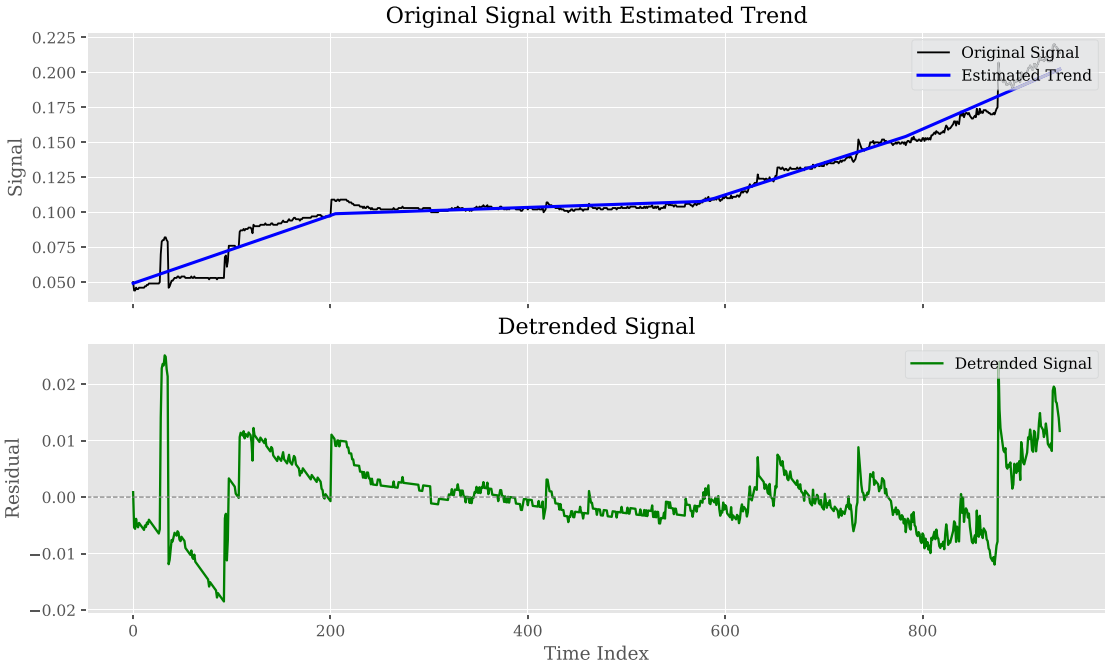


Fig. 5. Decomposition of time series into trend and residual components using ℓ_1 trend filtering.

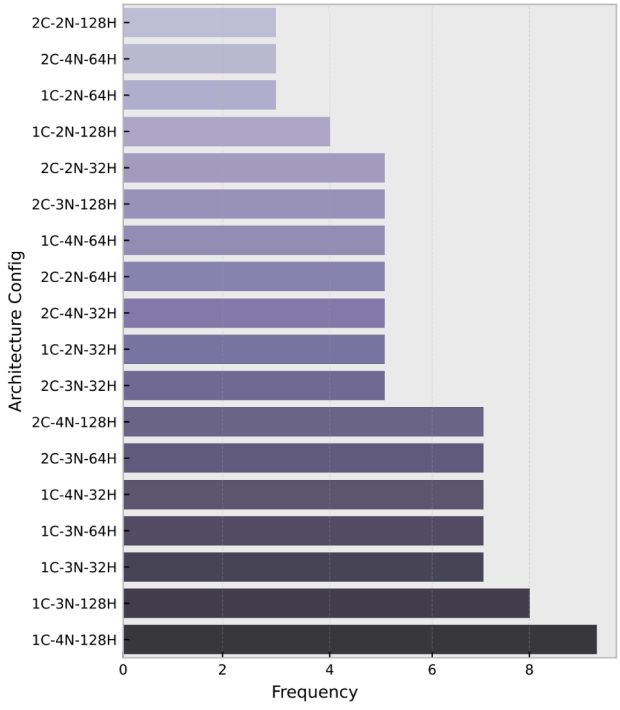


Fig. 6. Distribution of architecture configurations among top candidates (C = cells, N = nodes, H = hidden dimension).

where α^* represents the optimal architecture. The training incorporates:

Stochastic Weight Averaging (SWA): Beginning at epoch $\lfloor 0.33 T_{\text{final}} \rfloor$:

$$\theta_{\text{SWA}} = \frac{1}{n} \sum_{i=1}^{n} \theta_i \tag{112}$$

where n is the number of averaged checkpoints.

Advanced Scheduling: OneCycle learning rate scheduling with warmup period $0.3 T_{\text{final}}$ and cosine annealing.

Regularization: Dropout, weight decay, and data augmentation for improved generalization.

3.6.6. Regularization strategy

Algorithm 3 presents the architecture regularization strategy. This algorithm introduces a dynamic strategy to guide neural architecture search by enforcing regularization on architectural parameters. It computes a composite regularization loss, $\mathcal{R}_{\text{arch}}(\alpha)$, by weighting four components: entropy (to encourage exploration early on), identity penalty (to avoid overuse of identity operations), smoothness (to prevent abrupt changes in architecture parameters), and balance (to promote a diverse distribution of operations). These weights evolve across training epochs, gradually shifting emphasis from exploration to stability, enhancing the robustness and quality of the search process.

Algorithm 3: Architecture regularization.

Input : Architecture parameters α, current epoch t, total epochs T
Output: Regularization loss $\mathcal{R}_{\text{arch}}(\alpha)$

// Dynamic Regularization Weight Computation
$\lambda_1^{(t)} \leftarrow 0.1 \cdot (1 - t/T)$; // Entropy weight (decreasing)
$\lambda_2 \leftarrow 0.2$; // Identity penalty (constant)
$\lambda_3^{(t)} \leftarrow 0.05 \cdot (t/T)$; // Smoothness weight (increasing)
$\lambda_4^{(t)} \leftarrow 0.1 \cdot \max(0, 1 - 2t/T)$; // Balance weight (early training)
// Regularization Component Computation
$\mathcal{L}_{\text{entropy}} \leftarrow -\sum_{e,k} p_{e,k} \log p_{e,k}$ where $p_{e,k} = \text{softmax}(\alpha_k^{(e)})$;
$\mathcal{L}_{\text{identity}} \leftarrow \sum_e \max(0, p_{\text{identity}}^{(e)} - 0.5)^2$;
$\mathcal{L}_{\text{smooth}} \leftarrow \frac{1}{|E|} \sum_e \|\alpha_t^{(e)} - \alpha_{t-1}^{(e)}\|_2^2$; // If $t > 1$
$\mathcal{L}_{\text{balance}} \leftarrow \text{Var}(\{\sum_e p_{e,k} \mid k \in \mathcal{O}\})$;
// Aggregate Regularization
$\mathcal{R}_{\text{arch}} \leftarrow \lambda_1^{(t)} \mathcal{L}_{\text{entropy}} + \lambda_2 \mathcal{L}_{\text{identity}} + \lambda_3^{(t)} \mathcal{L}_{\text{smooth}} + \lambda_4^{(t)} \mathcal{L}_{\text{balance}}$;
return $\mathcal{R}_{\text{arch}}$;

3.6.7. Computational complexity analysis

The multi-fidelity approach significantly reduces computational cost compared to full training of all candidates. Let C_{zero}, C_{darts}, and C_{full} represent the costs of zero-cost evaluation, DARTS training, and full training

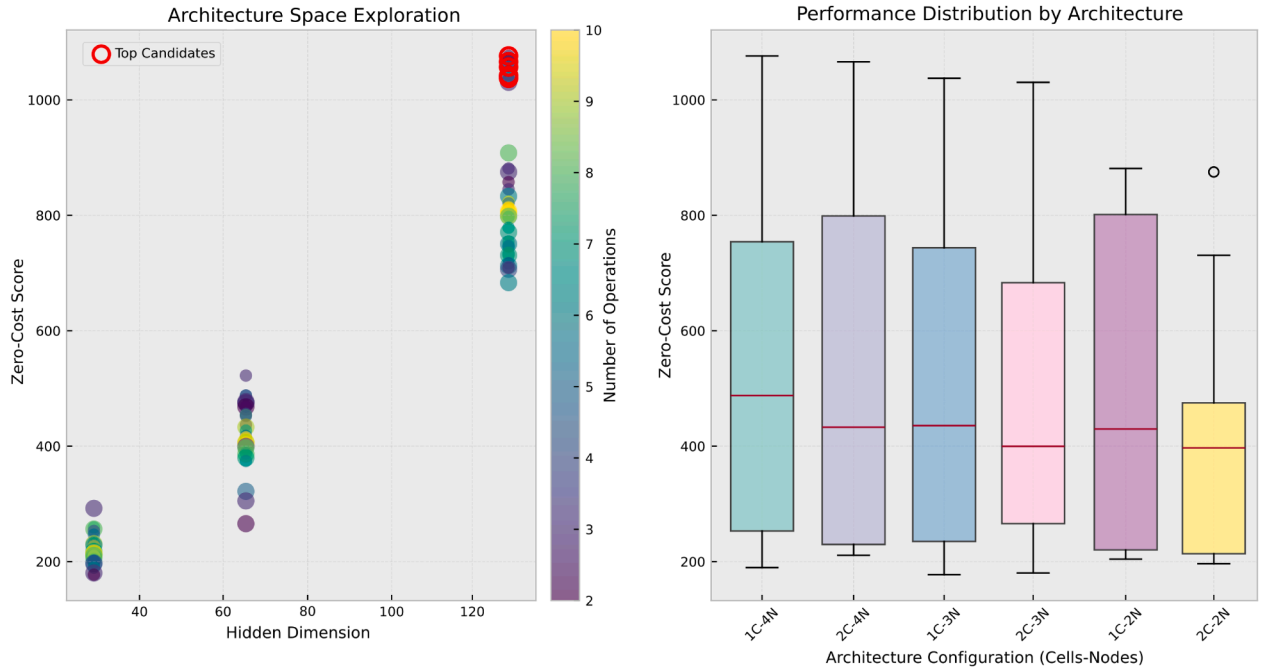


Fig. 7. Zero-cost metric score versus hidden dimension (top) and candidate distribution by operation count (bottom). Marker size corresponds to cell count; color intensity represents operation count.

respectively. The total computational cost is:

$$C_{\text{total}} = N_0 \cdot C_{\text{zero}} + K_1 \cdot C_{\text{darts}} + C_{\text{full}} \tag{113}$$

With typical values $N_0 = 1000$, $K_1 = 200$, $C_{\text{zero}} \ll C_{\text{darts}} \ll C_{\text{full}}$, this approach achieves significant speedup over exhaustive search while maintaining high-quality architecture discovery.

3.7. Model integration and evaluation

The DARTS model integrates all components within an encoder-decoder architecture optimized for forecasting. The model enables end-to-end learning of both neural architecture and forecasting parameters through joint optimization of architectural weights and network parameters.

Evaluation employs standard forecasting metrics:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{114}$$

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{115}$$

$$\text{RMSE} = \sqrt{\text{MSE}} \tag{116}$$

$$\text{R}^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} \tag{117}$$

where $SS_{\text{res}} = \sum_i (y_i - \hat{y}_i)^2$ and $SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2$.

The DARTS framework provides a foundation for automatically discovering optimal time series forecasting architectures while maintaining computational efficiency and architectural diversity through comprehensive regularization mechanisms.

3.8. Dataset description

To simulate salt contamination accumulating over time on insulator surfaces, six insulators were installed in a saline chamber. An 8.66 kV RMS, 60 Hz voltage was applied to all insulators in the same phase, while the salt concentration was gradually increased. This voltage level follows the NBR 10,621 (Brazilian National standard), which is analogous to IEC 60,507 [93], and is employed by the electrical power utility to determine insulator performance under artificial pollution for the 15 kV class in power grids. The insulator that withstood the application of high voltage for the longest time was considered in this study.

A LabVIEW-based interface was developed to monitor and record the applied voltage and the resulting leakage current. Each insulator was individually grounded, allowing measurement of the leakage current through a shunt resistor. Among the six insulators, two did not flash over, and their leakage current values were recorded until the end of the experiment. The remaining insulators were monitored only until surface breakdown occurred. The experiments were conducted at the High Voltage Laboratory of the Regional University of Blumenau (FURB), in Blumenau, SC, Brazil. To enable comparisons with future work, the dataset is publicly available at: https://github.com/SFStefenon/LeakageCurrent (accessed on February 09, 2026).

The experiment recorded 96,816 measurements of the leakage current of the evaluated insulators with a sampling rate of 1 second, corresponding to a total of 26 hours, 53 minutes, and 36 seconds of analysis. Considering that the focus of the analysis is to evaluate the signal variation trend, a downsample of 100 samples was used to reduce computational complexity, resulting in 968 records to be analyzed. For comparative analyses, 80% of the data was used to train the model and 20% for testing, with this division focused on predicting the increase in leakage current that precedes a failure (in the testing phase), which is the last 20% of the data in relation to the time window used.

4. Results and discussion

This section presents empirical results for the proposed multi-scale neural architecture, encompassing both the architecture search process and downstream forecasting performance. We begin by describing the preprocessing pipeline used to isolate trend and residual components in the time series data. Subsequently, we analyze the design space explored during neural architecture search, examining relationships between architectural parameters and zero-cost evaluation metrics. We provide a systematic assessment of the search process, including operation frequency distributions, computational cost-accuracy trade-offs, and

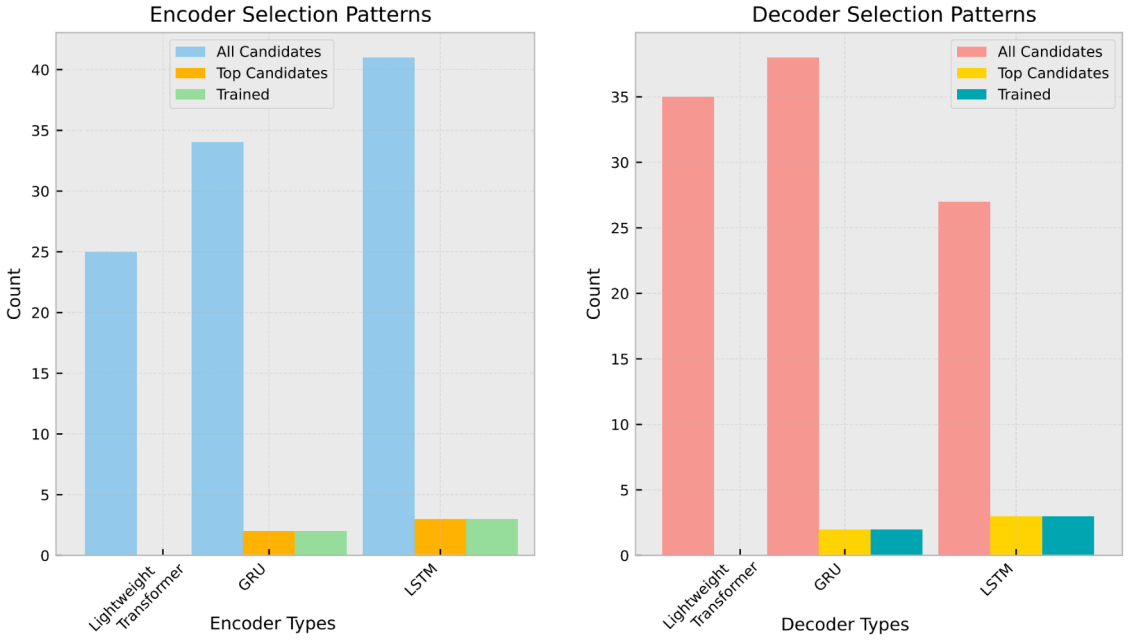


Fig. 8. Selection frequencies for encoder cells, decoder cells, and attention mechanisms among top-ranked candidates.

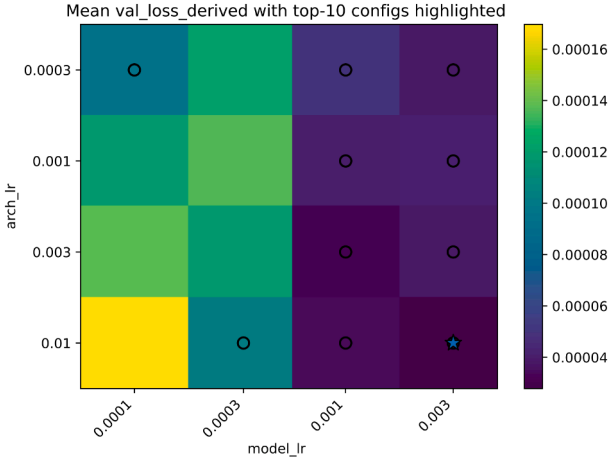


Fig. 9. Mean validation loss of derived architectures across the bilevel learning-rate grid. Circles denote top-10 configurations; the star indicates the best-performing setting.

comparative benchmarking against established forecasting models. Finally, we evaluate the sensitivity of the search procedure to initialization conditions and hyperparameter selection.

All experiments were conducted in Python using the `foreblocks` library, an open-source time-series forecasting framework[1]. The software stack used PyTorch 2.1.0, executed on a Linux workstation (Gentoo, `emerge`-based environment) with `zsh` as the default shell. The hardware platform comprises an Intel Core i9-14900K CPU (32 logical threads, up to 6.0 GHz), an NVIDIA GeForce RTX 4090 GPU, and 32 GiB of system memory (RAM). Unless stated otherwise, all reported timings (e.g., wall-clock runtime per search phase) were measured on this same machine to ensure reproducibility and consistent performance comparisons.

[1] https://github.com/lseman/foreblocks

4.1. Signal preprocessing: trend extraction and detrending

Given a univariate time series $y \in \mathbb{R}^n$, we extract a smooth trend component $x \in \mathbb{R}^n$ such that the residual $r = y - x$ captures local fluctuations or non-trend variations. We adopt a variational approach based on second-order total variation regularization, commonly referred to as ℓ_1 trend filtering. Fig. 5 illustrates this decomposition, comparing the original signal with the estimated trend and residual components.

The trend is estimated by solving the convex optimization problem:

$$\min_{x \in \mathbb{R}^n} \left\{ \frac{1}{2} \|y - x\|_2^2 + \lambda \|D^{(2)} x\|_1 \right\}, \tag{118}$$

where $\lambda > 0$ controls the trade-off between fidelity to the original signal and smoothness of the estimated trend. The operator $D^{(2)} \in \mathbb{R}^{(n-2)\times n}$ represents the discrete second-order difference matrix, defined such that $(D^{(2)}x)_t = x_t - 2x_{t+1} + x_{t+2}$ for $t = 1, \ldots, n-2$. The regularization term $\|D^{(2)}x\|_1$ penalizes abrupt changes in the second derivative of the trend, favoring piecewise linear solutions. This formulation approximates signals exhibiting multiple structural changes or non-smooth long-term trends.

To enhance robustness against outliers or impulsive noise, we replace the squared loss term with the Huber loss function $\rho_\delta(\cdot)$, which interpolates between ℓ_2 and ℓ_1 penalties depending on residual magnitude:

$$\min_{x \in \mathbb{R}^n} \left\{ \sum_{t=1}^{n} \rho_\delta(y_t - x_t) + \lambda \|D^{(2)} x\|_1 \right\}, \tag{119}$$

where $\rho_\delta(u) = \frac{1}{2}u^2$ for $|u| \le \delta$, and $\rho_\delta(u) = \delta(|u| - \frac{1}{2}\delta)$ otherwise. This formulation reduces sensitivity to extreme deviations while maintaining smooth behavior for small residuals.

Both formulations yield convex optimization problems solved numerically using the OSQP solver via the `cvxpy` framework. The detrended signal is obtained as $r = y - x$, isolating short-term variations after trend subtraction.

4.2. Architecture search space analysis

We investigate a neural architecture search strategy guided by zero-cost metrics (ZCMs) for sequence-to-sequence forecasting. The search

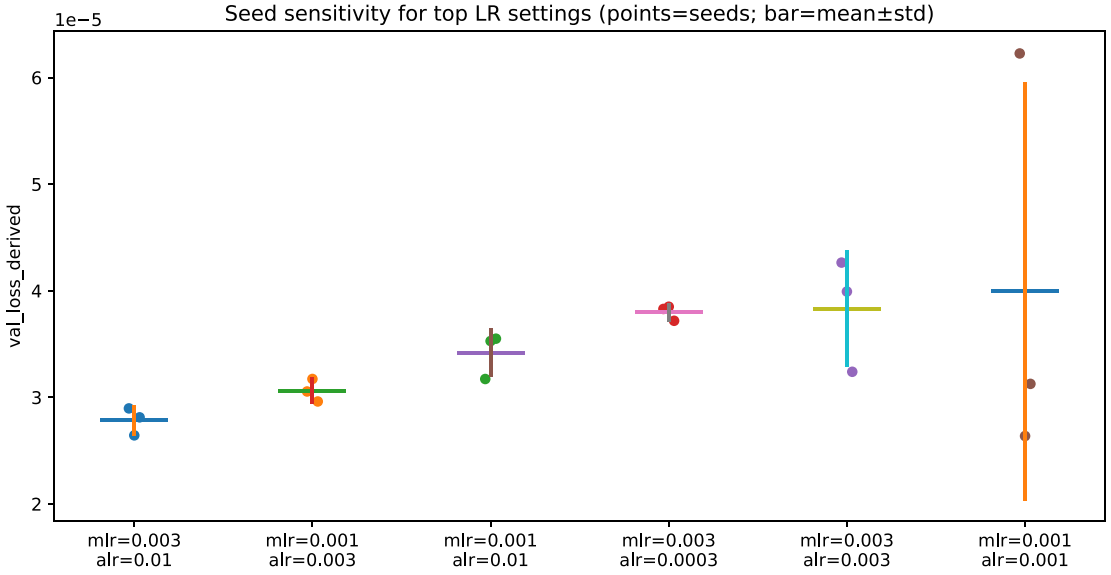


Fig. 10. Seed-level variability for top-performing learning-rate configurations. Points represent individual runs; bars show mean ± standard deviation.

space comprises 100 candidate architectures varying in depth (number of cells), width (hidden units), operation types, and recurrent cell choices. Based on aggregated ZCM scores, we select 5 top-ranked candidates for full training and evaluation.

4.2.1. Architectural configuration distribution

Fig. 6 presents the most frequently sampled architectural configurations. The majority of top-performing candidates belong to the 2-cell, 2-to-4-node, and 64-128 hidden unit families, which dominate the Pareto frontier between model complexity and zero-cost scores.

Notably, single-cell configurations are underrepresented among top-ranked candidates despite their lower parameter counts. This observation suggests that ZCM-guided selection implicitly favors architectures with sufficient representational capacity, potentially penalizing models unable to capture intermediate feature abstractions.

4.2.2. Search space landscape and zero-cost metric distribution

Fig. 7 visualizes the landscape of ZCM scores as a function of hidden dimension and architectural layout, where marker size represents cell count and color indicates operation count. While ZCM scores generally scale with hidden dimension, several compact architectures achieve competitive scores, indicating that the metrics capture architectural quality beyond mere parameter count.

The lower panel reveals that models with 6-9 operations concentrate in the high-scoring region, suggesting an optimal balance between architectural diversity and model parsimony. Beyond this range, increasing operation count yields diminishing returns in zero-cost performance indicators.

4.2.3. Encoder-decoder architecture preferences

Fig. 8 presents the selection frequency for encoder and decoder cell types among trained architectures. LSTM cells dominate both encoder and decoder positions, appearing in 60% of top candidates.

Lightweight Transformer variants, while present in the initial candidate pool, are absent from the final selection. This outcome likely reflects two factors: (i) zero-cost metrics may underestimate the long-term dependency modeling capabilities of attention-based encoders, particularly for evaluation signals computed from limited data samples, and (ii) LSTM architectures demonstrate higher compatibility with short-sequence forecasting tasks through their inherent gating mechanisms, which provide more favorable gradient properties during the zero-cost evaluation phase.

Furthermore, all trained candidates incorporate attention bridges at layer 0, indicating that explicit cross-timestep context mechanisms improve performance even within predominantly recurrent architectures. This pattern suggests that hybrid approaches combining recurrent processing with selective attention may offer advantages for sequence-to-sequence forecasting.

4.3. Bilevel optimization: Learning rate sensitivity

In bilevel optimization-based neural architecture search, two learning rates govern the optimization dynamics: the *model learning rate* η_w, which updates network weights in the inner loop, and the *architecture learning rate* η_α, which updates continuous architecture parameters in the outer loop. The interaction between η_w and η_α affects optimization stability, convergence speed, and final architecture quality. We conduct a systematic sensitivity analysis to characterize the robustness of our hyperparameter selection.

We evaluate a grid of learning-rate pairs:

$$\eta_w \in \{10^{-4}, 3\times10^{-4}, 10^{-3}, 3\times10^{-3}\}, \quad \eta_\alpha \in \{3\times10^{-4}, 10^{-3}, 3\times10^{-3}, 10^{-2}\}.$$

For each configuration, the DARTS search phase is executed for a fixed number of epochs across multiple random seeds. The final discrete architecture is derived from the continuous relaxation and evaluated on a validation set.

Fig. 9 presents mean validation loss of derived architectures across the learning-rate grid, with darker regions indicating lower (superior) loss values. The top-10 configurations are highlighted, and the optimal pair is marked with a star.

Performance varies systematically across the grid. Extremely small η_w values paired with large η_α degrade performance, likely due to insufficient weight adaptation relative to architecture parameter updates. Conversely, intermediate-to-high η_w values (e.g., 10^{-3} to 3×10^{-3}) combined with moderate η_α yield consistently favorable results.

The optimal region spans a contiguous neighborhood rather than an isolated point, indicating that the selected learning rates reside in a stable basin of the objective landscape. This observation supports the robustness of our hyperparameter choices and suggests that precise tuning is not strictly necessary within this regime.

To assess reproducibility, Fig. 10 visualizes per-seed validation losses for top-performing learning-rate configurations. Each point corresponds to an independent random seed; horizontal bars indicate mean ± standard deviation.

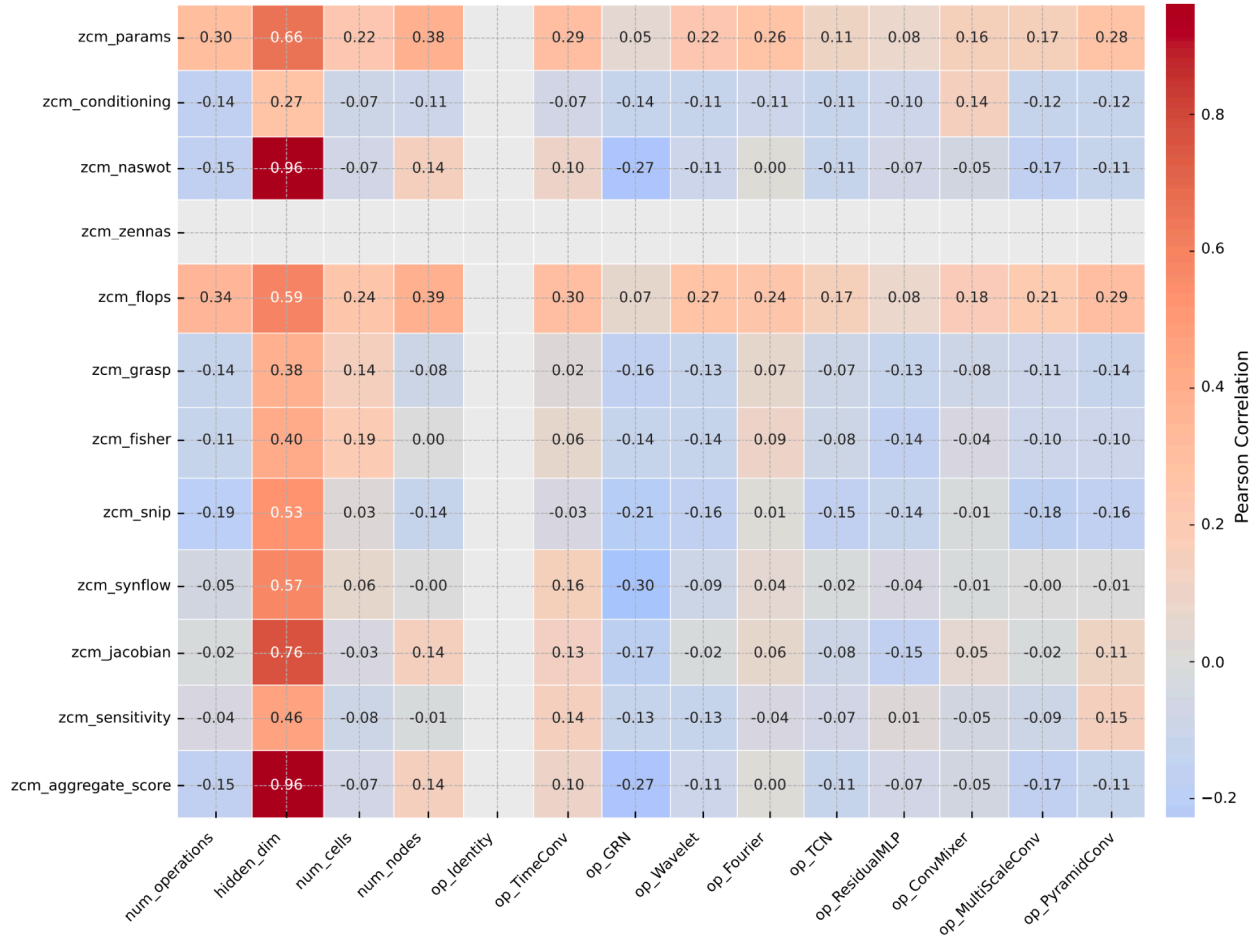


Fig. 11. Pearson correlation matrix between architectural parameters and zero-cost metrics.

The results demonstrate low variance across seeds for top configurations, confirming that performance improvements are not artifacts of favorable initialization. Standard deviations remain small relative to mean performance, indicating consistent optimization behavior across independent runs.

These analyses reveal: (*i*) bilevel optimization exhibits structured sensitivity patterns rather than chaotic instability; (*ii*) excessively large η_α can destabilize architecture search when not paired with sufficient η_w; (*iii*) moderate-to-high η_w values improve convergence of inner-loop weight updates, stabilizing architecture gradient estimates; and (iv) the selected configuration resides within a robust region rather than at unstable boundaries.

4.4. Zero-cost metric analysis

4.4.1. Correlational structure of architectural components and metrics

Fig. 11 presents Pearson correlations between architectural components and zero-cost metrics.

Several patterns emerge from the correlation analysis:

- Hidden dimension exhibits strong correlation with aggregate ZCM score ($r = 0.961$) and moderate correlation with individual metrics (e.g., NASWOT $r = 0.76$, Synflow $r = 0.57$).
- Operation count correlates with both parameter count and validation loss ($r = 0.800$), suggesting potential over-parameterization in high-operation-count architectures.
- Operation types show weak to moderate correlations with individual metrics (e.g., Fourier operations $r = 0.27$ with Synflow), indicating limited macro-level influence but potentially significant micro-architectural effects.

4.4.2. Distribution characteristics of zero-cost metrics

Fig. 12 displays the distributions of ZCM values across all candidate architectures.

Several metrics, including Synflow, NASWOT, and Jacobian trace, exhibit clear stratification between low- and high-performing candidates, providing discriminative power for architecture ranking. In contrast, metrics such as Grasp and Fisher demonstrate narrower value ranges with reduced discriminatory capacity in this search space. The aggregate ZCM score, constructed as a weighted combination of individual metrics, exhibits a wider dynamic range, enhancing candidate separation.

4.4.3. Metric importance and ranking stability

We analyze the sensitivity of architecture ranking to aggregation weight selection through complementary approaches: leave-one-out ablation and weight perturbation analysis.

Fig. 13 presents the leave-one-out importance analysis. Removing NASWOT from the metric ensemble substantially alters both global ranking (Spearman $\rho = 0.43$) and top-k selection (20% overlap with baseline). In contrast, removing any other individual metric yields rankings nearly identical to the baseline ($\rho \approx 1.0$, 100% top-k overlap).

The Jacobian-based metric contributes minimally but non-negligibly ($\Delta\rho \approx 0.01$), whereas gradient-based metrics (GRASP, SNIP, SynFlow, Fisher), complexity metrics (FLOPs, parameters), and conditioning-related metrics (sensitivity, zennas) exhibit complete redundancy within

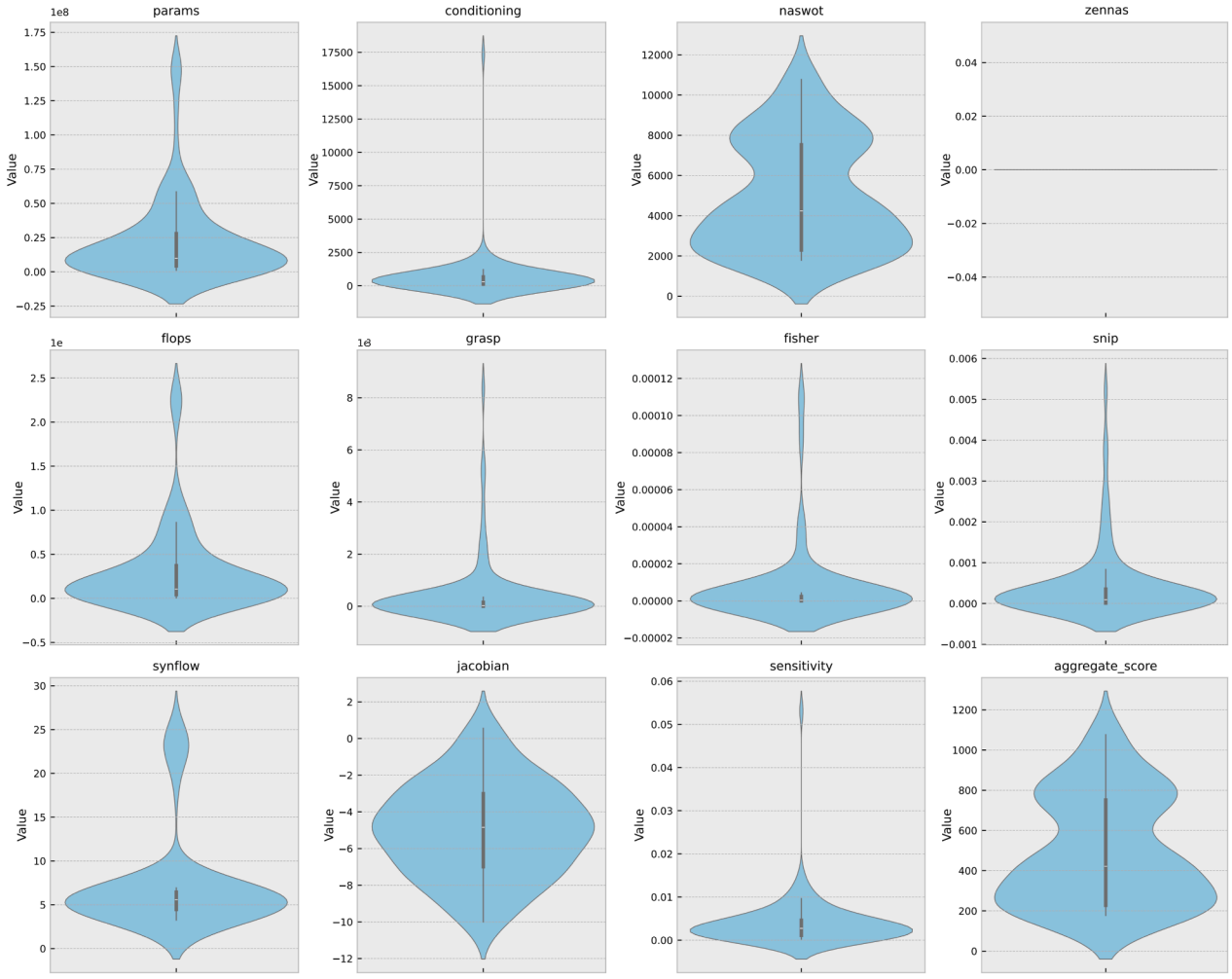


Fig. 12. Violin plots of zero-cost metric distributions across all candidates. Values are normalized for comparative visualization.

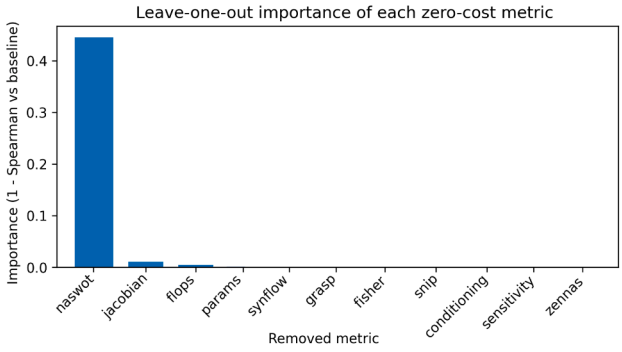


Fig. 13. Leave-one-out importance analysis quantifying ranking degradation (measured as $1 - \rho_{\text{Spearman}}$) when each metric is excluded. NASWOT demonstrates dominant importance ($\Delta\rho = 0.57$), while gradient-based and complexity metrics exhibit strong redundancy ($\Delta\rho < 0.02$).

the ensemble. This pattern indicates that NASWOT provides complementary structural information-specifically, trainability signals derived from input-output Jacobian properties-not captured by gradient norms or architectural complexity alone. Fig. 14 examines ranking stability across different weight assignment schemes.

Candidate rankings remain consistent across baseline weighting, uniform weights, and random perturbations (rand_00 through rand_05), with most candidates maintaining similar rank positions. Subset-based schemes that isolate specific metric families (gradient-only, activation-only, complexity-only) introduce moderate rank variations, particularly for mid-tier candidates (IDs 2, 9, 5), while top-performing candidates (IDs 4, 17, 8) remain stable. The subset_no_penalties and subset_pos_only schemes, which exclude regularization terms, produce the largest perturbations.

These findings demonstrate that the aggregation approach exhibits robustness to exact weighting coefficients-most metrics serve supporting roles rather than driving selection-yet NASWOT remains critical for accurate architecture ranking. The ensemble design leverages metric redundancy to achieve stable rankings while relying on NASWOT's unique contribution to differentiate high-quality architectures.

4.4.4. *Operation-level analysis*

Fig. 15 provides a detailed view of operation usage across sampled and trained architectures. Temporal convolutions, ConvMixer, and residual MLPs appear frequently in both sampled and trained models. However, performance impact analysis reveals that certain operations (e.g., GRN, PyramidConv) contribute negligibly or negatively to downstream validation loss. The architectural complexity histogram confirms that most candidates contain 6-10 operations, reinforcing the balance between expressive capacity and overfitting risk.

4.4.5. *Predictive power of zero-cost metrics*

Fig. 16 assesses the relationship between ZCM scores and training outcomes. Candidates in the top ZCM quartile exhibit a 20% training success rate, while all other quartiles fail to produce viable models.

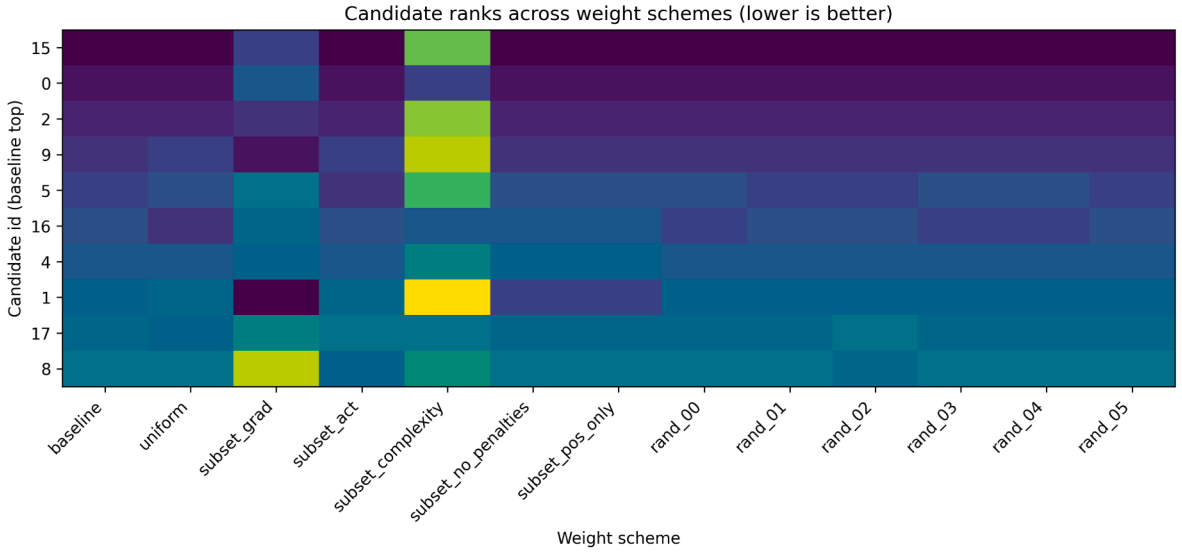


Fig. 14. Stability of candidate rankings across weight perturbation schemes. Each row represents a candidate (ordered by baseline ranking); each column represents a weighting configuration. Color intensity indicates rank position (darker indicates superior rank). Top-performing candidates maintain stability across schemes, while mid-tier candidates exhibit moderate sensitivity.

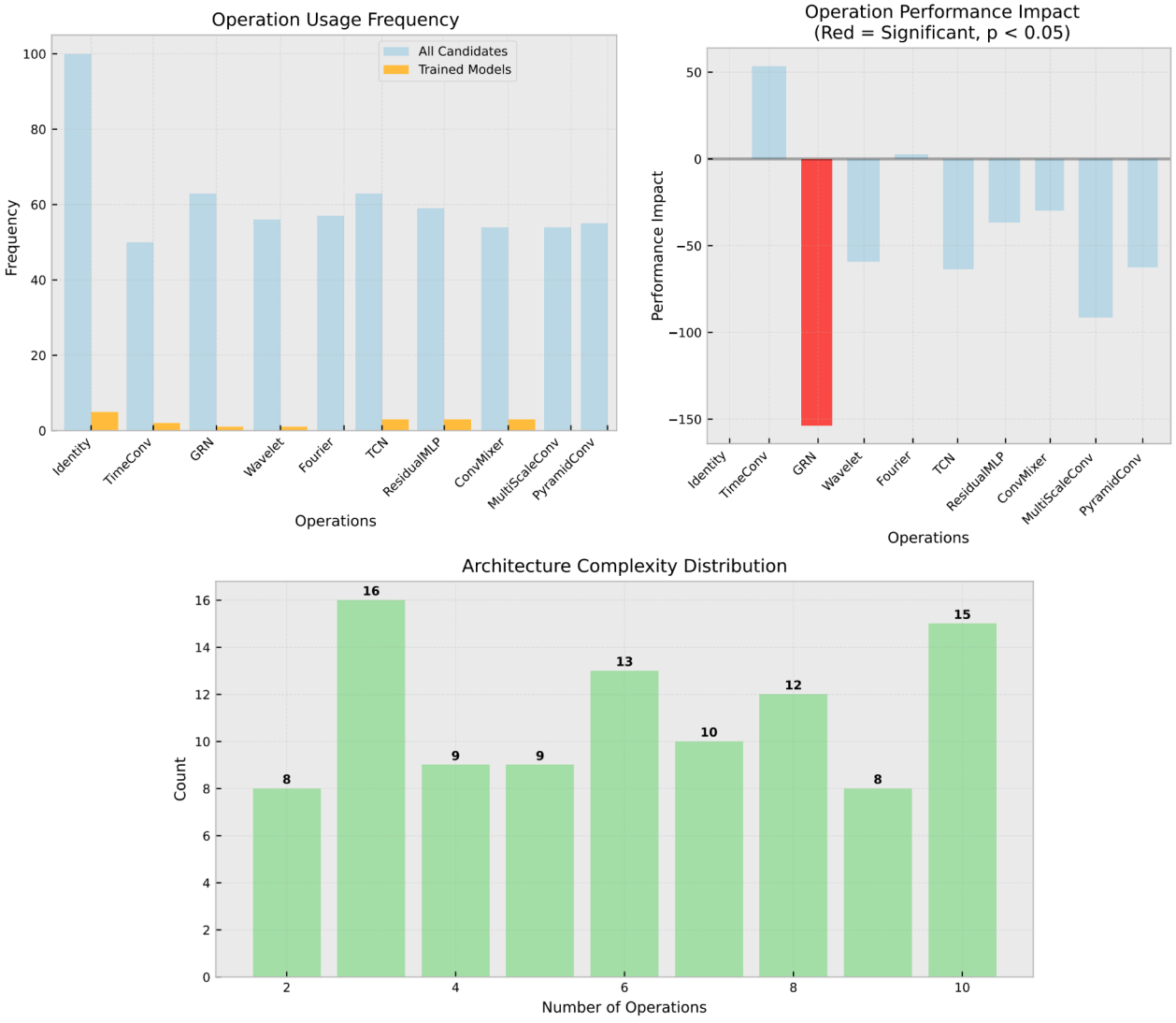


Fig. 15. Operation-level analysis: usage frequency (top), performance impact (middle), and architectural complexity distribution (bottom).

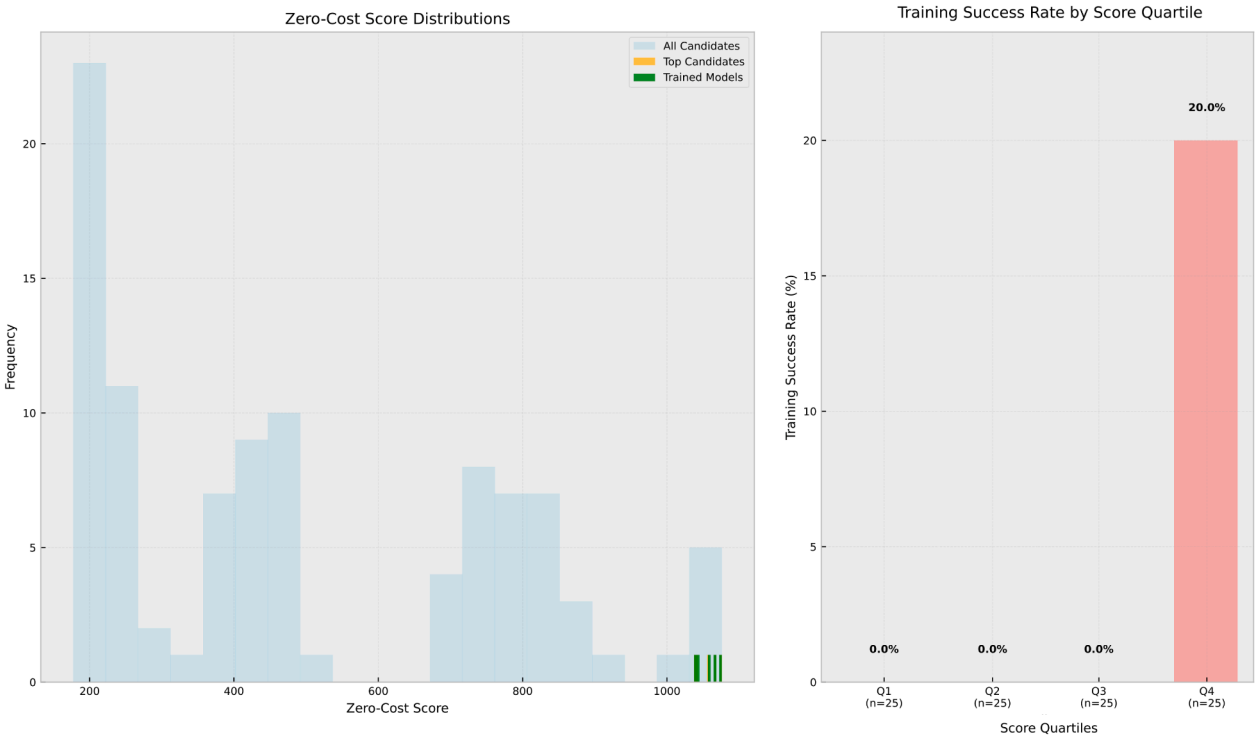


Fig. 16. Distribution of zero-cost scores (top) and training success rates by score quartile (bottom).

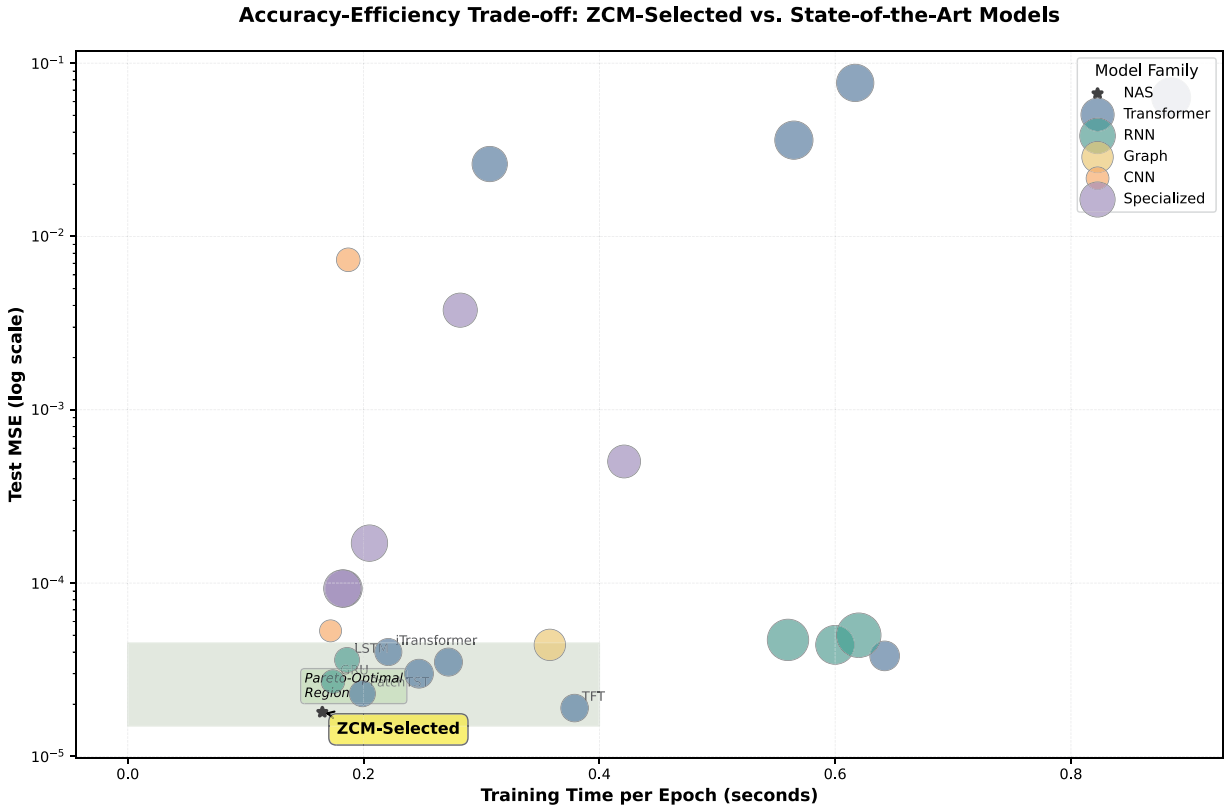


Fig. 17. Accuracy-efficiency trade-off across evaluated models. The ZCM-selected architecture (red star) achieves the best MSE with competitive training time, occupying the Pareto-optimal region. State-of-the-art transformer models (TFT, PatchTST) form a secondary frontier with comparable accuracy but increased computational cost. Legacy transformers (Informer, Autoformer, FEDformer) exhibit poor performance for this short-horizon forecasting task. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

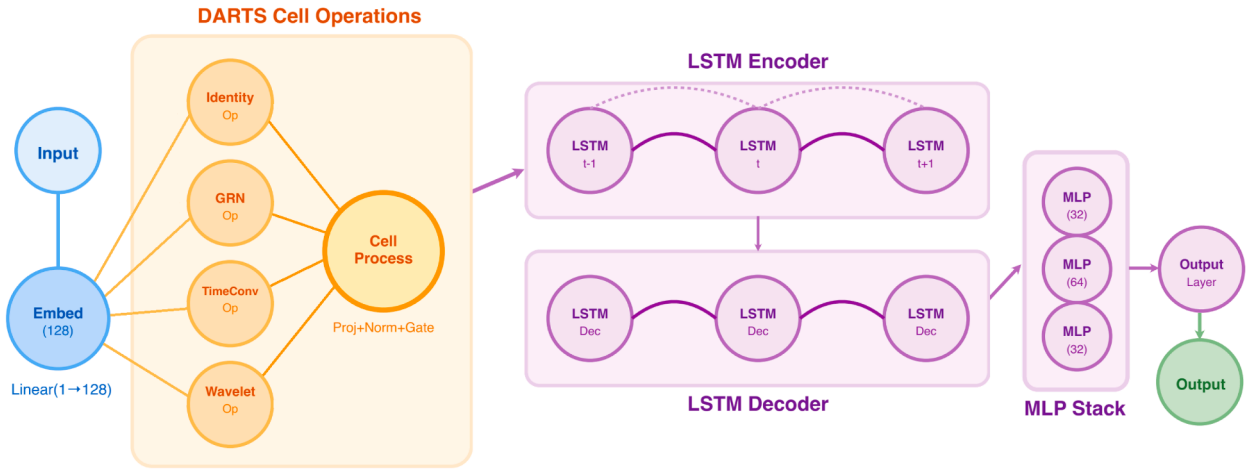


Fig. 18. Architecture topology of the selected model derived from ZCM-guided search.

Table 2
Performance comparison during ZCM-guided architecture selection. Phase 3 reports validation metrics for top-5 candidates identified by zero-cost screening; phase 5 reports final test performance after full training.

Candidate	Parameters	Val. Loss	MSE	MAE
Phase 3: Candidate Evaluation				
1	378	0.000056	0.000701	0.022205
2	858	0.000047	0.001967	0.037774
3	202	0.000059	0.000171	0.010728
4	146	0.000014	0.000194	0.011905
5	1410	0.000047	0.000200	0.010721
Phase 5: Final Model Performance				
Selected	**378**	**0.000009**	**0.000018**	**0.002439**

This result supports the utility of ZCMs for early-stage architecture pruning, substantially reducing computational costs by avoiding full training of unpromising candidates. The bimodal distribution of scores among trained models indicates that false positives (high ZCM scores with poor validation performance) remain a challenge, suggesting opportunities for refinement of the metric aggregation strategy.

4.5. Comparative performance evaluation

Table 2 summarizes the ZCM-guided selection process across two phases. Phase 3 employs zero-cost proxy metrics to screen the architectural search space and identify promising candidates. While Candidate 4 achieves the lowest validation loss at this stage (Val. Loss = 0.000014), the final selection considers architectural complexity, parameter efficiency, and expected generalization capacity after full optimization. This process selects Candidate 1, which balances competitive early-stage performance with architectural parsimony (378 parameters).

After full training (Phase 5), the selected architecture achieves test MSE of 1.8×10^{-5} and MAE of 2.439×10^{-3}, representing substantial improvement over its Phase 3 validation metrics. This behavior underscores the importance of considering learning dynamics and generalization potential beyond early-stage validation performance.

To assess the effectiveness of the ZCM-guided architecture search, we compare the selected model against a comprehensive set of baselines spanning traditional sequence-to-sequence models, Transformer-based forecasting architectures, and specialized time series methods. Table 3 presents results for 20 baseline models evaluated under identical experimental conditions (input sequence length, forecast horizon, data preprocessing, and evaluation protocol).

The ZCM-selected architecture achieves the lowest MSE (0.000018) and MAE (0.002439) among all evaluated models, demonstrating competitive performance against both traditional architectures and recent state-of-the-art forecasting methods. Several patterns emerge from the comparative analysis:

4.5.1. State-of-the-art transformer models

Among Transformer-based architectures, Temporal Fusion Transformer (TFT) [94] and PatchTST [95] demonstrate strong performance, achieving MSE values of 0.000019 and 0.000023, respectively. TFT incorporates variable selection networks and multi-head attention mechanisms specifically designed for multi-horizon forecasting, while PatchTST leverages patch-based tokenization and channel independence to capture local temporal patterns efficiently. The selected architecture outperforms both models, indicating that the ZCM-guided search successfully identified an effective architectural configuration that matches or exceeds task-specific designs.

Recent variants such as TimeMixer [96], TimeXer [97], and iTransformer [98] introduce specialized mechanisms for temporal mixing, cross-variate modeling, and inverted attention patterns. While these architectures demonstrate reasonable performance (MSE: 0.000030-0.000040), they underperform relative to simpler approaches on this dataset, suggesting that their design choices may be better suited for multivariate forecasting scenarios with complex cross-series dependencies.

Earlier Transformer variants (Informer [20], Autoformer [19], FEDformer [99]) exhibit substantially degraded performance (MSE: 0.026-0.077), likely due to suboptimal attention mechanisms and insufficient adaptation to univariate industrial time series characteristics. These models were primarily designed for long-sequence forecasting with large receptive fields, which may introduce unnecessary complexity for the 24-hour forecast horizon considered here.

4.5.2. Recurrent neural networks

Standard GRU and LSTM architectures achieve competitive results (MSE: 0.000027 and 0.000036), demonstrating the continued effectiveness of recurrent models for short-horizon forecasting. The performance gap between vanilla RNNs and the baseline seq2seq variants (LSTM + Attention: MSE = 0.000044) indicates that encoder-decoder structures with attention mechanisms provide marginal benefits. The ZCM-selected architecture bridges this gap by automatically incorporating LSTM cells with optimized attention mechanisms (as shown in Fig. 8), achieving superior accuracy without manual architectural design.

Table 3
Comprehensive performance comparison on the Insulator 1 dataset. All models use identical input/output configurations (168-hour lookback, 24-hour forecast horizon). Metrics computed on the test set. Models are grouped by architectural family and sorted by MSE within each group.

Family	Model	RMSE	MSE	MAE	MAPE (%)	Time (s)
ZCM-Guided Architecture (Proposed)						
NAS	Selected Architecture	**0.004243**	**0.000018**	**0.002439**	**196.45**	**0.165**
Transformer-Based Forecasting Models						
Transformer	TFT	0.004353	0.000019	0.002794	225.48	0.379
Transformer	PatchTST	0.004753	0.000023	0.002702	265.59	0.199
Transformer	TimeMixer	0.005448	0.000030	0.003512	360.43	0.247
Transformer	TimeXer	0.005949	0.000035	0.004657	587.82	0.272
Transformer	TimesNet	0.006169	0.000038	0.004291	292.03	0.642
Transformer	iTransformer	0.006327	0.000040	0.004663	439.79	0.221
Transformer	VanillaTransformer	0.161807	0.026181	0.146512	19766.44	0.307
Transformer	Autoformer	0.189624	0.035957	0.138764	16639.02	0.565
Transformer	FEDformer	0.252021	0.063515	0.221554	27554.95	0.885
Transformer	Informer	0.277360	0.076929	0.205444	25626.87	0.617
Recurrent Neural Networks						
RNN	GRU	0.005241	0.000027	0.003158	240.71	0.174
RNN	LSTM	0.006036	0.000036	0.004825	555.49	0.186
RNN	Seq2Seq LSTM (baseline)	0.007071	0.000050	0.005508	—	0.620
RNN	Seq2Seq GRU (baseline)	0.006856	0.000047	0.005462	—	0.560
RNN	LSTM + Attention (baseline)	0.006633	0.000044	0.005201	—	0.600
Convolutional and Graph-Based Models						
CNN/Graph	TCN	0.007284	0.000053	0.005445	746.36	0.172
CNN/Graph	BiTCN	0.085642	0.007334	0.080104	9308.12	0.187
CNN/Graph	StemGNN	0.006598	0.000044	0.005425	673.23	0.358
Specialized Forecasting Architectures						
Specialized	NBEATS	0.009647	0.000093	0.008466	995.16	0.183
Specialized	NBEATSx	0.009647	0.000093	0.008466	995.16	0.182
Specialized	NHITS	0.013030	0.000170	0.010980	1326.43	0.205
Specialized	DeepAR	0.022435	0.000503	0.021366	2650.75	0.421
Specialized	TiDE	0.061285	0.003756	0.053444	6351.09	0.282

4.5.3. Convolutional and specialized architectures

Temporal Convolutional Networks (TCN) achieve moderate performance (MSE: 0.000053), while N-BEATS [100], a specialized architecture based on backward and forward residual links, produces higher errors (MSE: 0.000093). These results suggest that pure convolutional or residual-based approaches may struggle to capture the temporal dependencies present in insulator current signals. DeepAR [101], a probabilistic forecasting model, exhibits substantial degradation (MSE: 0.000503), indicating that the overhead of probabilistic modeling may not be justified for deterministic point forecasting in this context.

4.5.4. Computational efficiency

Training time varies substantially across models (0.165-0.885 seconds per epoch). The selected architecture achieves competitive inference speed (0.165s), faster than most Transformer-based models (0.199-0.885s) and comparable to lightweight RNN variants (0.174-0.186s). This efficiency advantage makes the approach suitable for deployment in resource-constrained industrial monitoring systems where rapid model updates or retraining may be required.

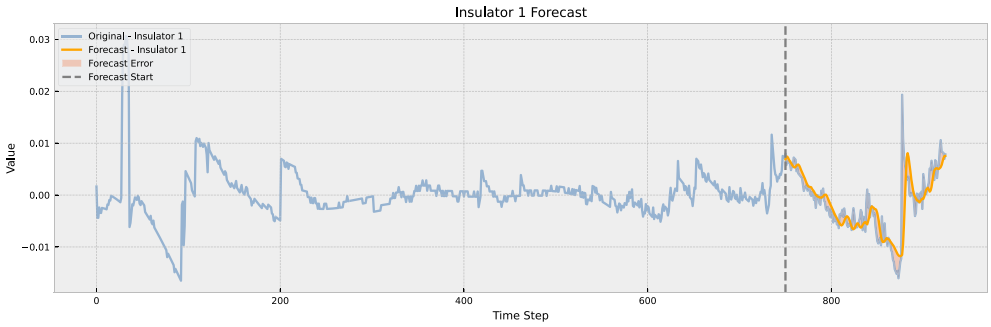
The comparison highlights several advantages of the ZCM-guided approach:

1. **Performance parity with task-specific architectures:** The selected model achieves accuracy comparable to or exceeding specialized forecasting architectures (TFT, PatchTST) that incorporate domain-specific inductive biases. This demonstrates that data-driven architecture search can discover effective configurations without relying on hand-crafted design patterns.
2. **Competitive accuracy across metrics:** The ZCM-selected architecture achieves the best performance across multiple evaluation metrics (MSE, RMSE, MAE, MAPE), indicating robust predictive capability rather than optimization for a single objective.
3. **Robustness across architectural families:** The selected model outperforms representatives from multiple architectural families (Transformers, RNNs, CNNs, specialized forecasting models), suggesting that the search process successfully navigated a diverse design space to identify a locally optimal configuration.
4. **Reduced hyperparameter sensitivity:** Unlike manually designed architectures that require extensive tuning of depth, width, attention heads, and other structural hyperparameters, the ZCM-guided approach automatically determines these choices based on zero-cost proxy metrics. This reduces the risk of suboptimal architectural decisions that may arise from limited hyperparameter search budgets.
5. **Efficient computational profile:** The selected architecture achieves state-of-the-art accuracy while maintaining training times competitive with lightweight models, demonstrating that the search process balanced multiple competing objectives effectively.

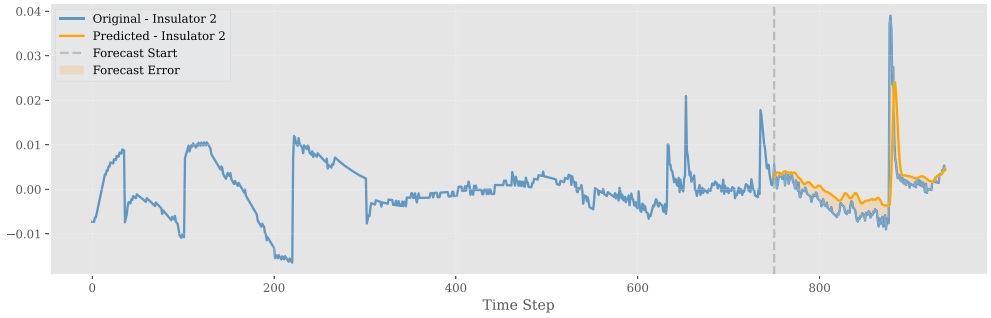
Fig. 17 visualizes the accuracy-efficiency trade-off across all evaluated models. The ZCM-selected architecture occupies the Pareto-optimal region, simultaneously minimizing prediction error and training time. This positioning demonstrates that automated architecture search can discover configurations that would be difficult to identify through manual design, particularly when balancing multiple competing objectives.

4.6. Forecasting and error analysis

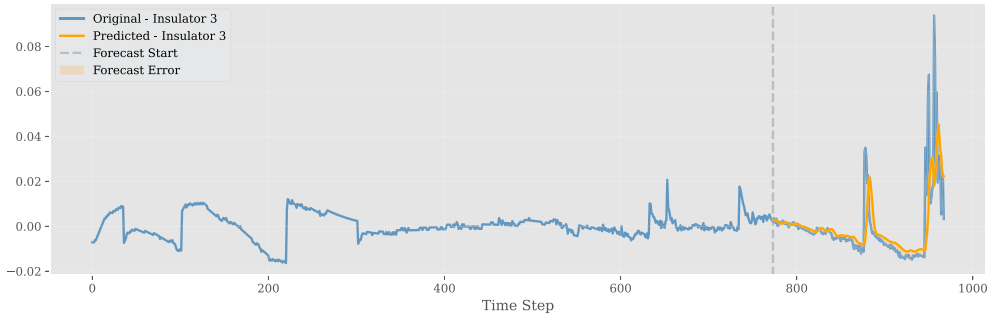
Fig. 18 presents the final selected architecture topology. Fig. 19 visualizes forecasting performance on the insulators 1 to 4. The predicted sequence tracks the true signal with low residual error and stable transitions across the forecast horizon.



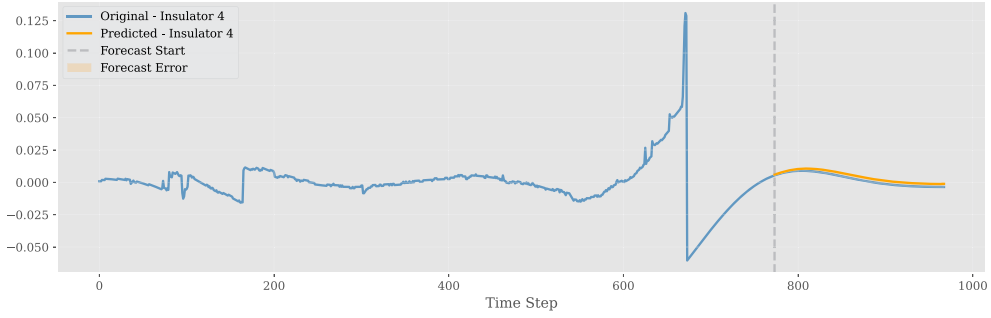
(a) Insulator 1



(b) Insulator 2



(c) Insulator 3



(d) Insulator 4

Fig. 19. Forecasting performance of the selected architecture across four insulators. In all cases, predicted values closely follow the ground truth dynamics, with deviations highlighted by the shaded error region between curves. The model preserves amplitude, trend behavior, and short-term fluctuations consistently across units.

Prediction errors concentrate at signal peaks, where greater variability and non-linearities occur. This pattern is consistent with observations in related work, where filtering techniques mitigate peak-region errors. For example, Buratto et al. [102] employ the Christiano-Fitzgerald filter, while Branco et al. [103] apply the Christiano-Fitzgerald random walk filter. Wavelet transforms, implemented as network layers in Rodríguez et al. [104], represent another approach; notably, wavelet operations are included in our DARTS search space (Fig. 1), allowing data-driven selection of appropriate filtering mechanisms.

4.7. Performance under cross-validation

To evaluate the generalization capability and robustness of the selected architecture, we adopt a repeated cross-validation strategy

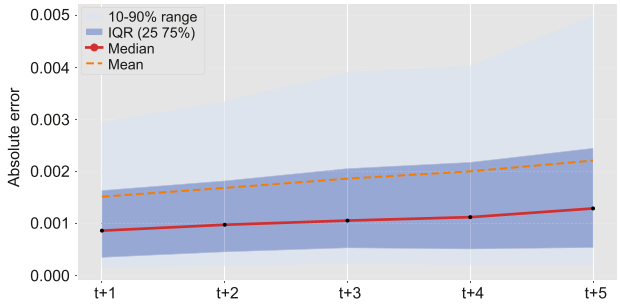


Fig. 20. Forecast error growth expressed as quantile bands aggregated across all cross-validation splits and random seeds. The light band shows the 10-90% range, the darker band highlights the interquartile range (25-75%), while solid and dashed lines indicate the median and mean absolute error, respectively. The gradual widening of the bands demonstrates controlled error accumulation over the 5-step forecast horizon.

using multiple random train/validation splits (80/20 ratio) combined with different random seeds for reproducibility. This approach allows us to quantify both average performance and variability across different data partitions, providing a more reliable assessment than a single-split evaluation.

Fig. 20 reveals a smooth and predictable increase in absolute error with forecast horizon. Median error grows approximately linearly from $t+1$ to $t+5$, while the interquartile range remains relatively stable until the last step, indicating consistent model behavior across most samples. The modest widening of the 10-90% envelope at longer horizons suggests that extreme errors remain rare, which is desirable for industrial monitoring applications where large outliers can be particularly costly.

Fig. 21 provides a more detailed view of the error distributions. The violin shapes remain symmetric and compact for the first three steps, becoming slightly right-skewed at $t+4$ and $t+5$ due to a small number of larger errors. Importantly, the median (central line inside each box) increases only gradually, and the interquartile range expands modestly, confirming that the majority of predictions remain accurate even as uncertainty naturally accumulates with longer horizons.

Finally, Fig. 22 quantifies overall performance robustness by showing the distribution of MSE values computed on the held-out validation sets across all repeats. The tight interquartile range (approximately 1.0×10^{-5} to 1.6×10^{-5}) and absence of extreme outliers demonstrate that the ZCM-guided architecture consistently achieves strong predictive accuracy regardless of the specific train/validation partition or initialization seed used.

4.8. Sensitivity to operator pool initialization

A potential source of variability in neural architecture search stems from the initial operator pool $\mathcal{O}$ from which candidate architectures are sampled. To assess sensitivity to this initialization, we conduct a robustness analysis over multiple randomly generated operator pools.

4.8.1. Experimental protocol

Let $\mathcal{O}$ denote the full set of available operators. We sample P distinct operator subsets $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_P \subseteq \mathcal{O}$, each serving as a constrained search space. For each pool $\mathcal{O}_p$, we generate N candidate architectures by sampling structural hyperparameters and selecting operators exclusively from $\mathcal{O}_p$.

Each candidate c is evaluated using the zero-cost metric aggregation:

$$S(c) = \sum_{k=1}^{K} w_k \, f_k(c),$$

where $f_k(c)$ denotes the k-th raw metric and w_k represents the baseline aggregation weights. Candidates are ranked within each pool, and the top-K architectures are selected for comparison.

4.8.2. Stability metrics

To quantify cross-pool robustness, we define for each architecture signature σ:

- **Top-K Frequency:** $\mathrm{Freq}(\sigma) = \sum_{p=1}^{P} \mathbb{I}\{\sigma \in \mathrm{TopK}(\mathcal{O}_p)\}$, indicating how often σ appears among top-K candidates.
- **Average Rank:** $\mathrm{AvgRank}(\sigma) = \frac{1}{P_\sigma} \sum_{p \in \mathcal{P}_\sigma} r_p(\sigma)$, where $r_p(\sigma)$ is the rank of σ in pool p, and $\mathcal{P}_\sigma$ denotes pools containing σ.
- **Worst-Case Rank:** $\mathrm{WorstRank}(\sigma) = \max_{p \in \mathcal{P}_\sigma} r_p(\sigma)$.

4.8.3. Results and interpretation

Fig. 23 displays rank distributions across operator pools for the most stable architecture signatures.

Two distinct behaviors emerge:

1. Several signatures exhibit near-constant rank distributions (median rank = 1, minimal variance), demonstrating strong robustness to operator pool perturbations.
2. Other signatures show higher interquartile ranges and occasional rank degradation, suggesting partial dependence on specific operator configurations.

The persistence of architectures maintaining top ranks across diverse operator pools indicates that the selection strategy does not overfit to particular initializations of $\mathcal{O}$. Instead, it consistently identifies high-performing structural motifs that generalize across operator subsets.

Nevertheless, architectures with wider rank dispersion highlight that certain operator combinations may interact favorably only within specific pool contexts. This observation motivates future investigation of adaptive operator pruning strategies that reduce initialization variance while preserving search expressiveness. Additionally, incorporating domain knowledge to constrain the initial operator pool based on problem characteristics may improve consistency without sacrificing generality.

4.9. Discussion

The experimental results demonstrate that zero-cost metric-guided neural architecture search can identify compact, high-performing models for time series forecasting while simplifying the model selection process relative to manually designed architectures. Several key insights emerge from the analysis:

Architectural preferences: The search process consistently favors moderate-depth architectures (2 cells) with intermediate hidden dimensions (64-128 units) and 6-9 operations. Single-cell architectures, despite lower parameter counts, are underrepresented among top candidates, suggesting that representational capacity requirements dominate efficiency considerations in the ZCM evaluation phase.

Metric complementarity: While multiple zero-cost metrics provide redundant information, NASWOT contributes uniquely to architecture ranking through trainability signals derived from Jacobian analysis. This finding suggests that ensemble-based metric aggregation should prioritize metrics capturing orthogonal aspects of architectural quality rather than merely averaging correlated indicators.

Bilevel optimization stability: The systematic sensitivity analysis reveals that bilevel optimization exhibits structured rather than chaotic hyperparameter dependence. The selected learning rates reside in a stable basin of the optimization landscape, reducing the risk of performance degradation due to minor hyperparameter perturbations.

Operator pool robustness: Architecture rankings demonstrate partial but not complete invariance to operator pool initialization. Top-performing signatures maintain consistency across diverse pools, indicating that the method discovers intrinsically strong patterns. However, mid-tier architectures show greater sensitivity, suggesting opportunities for adaptive pool refinement strategies.

Practical implications: The ZCM-guided approach achieves competitive forecasting accuracy with minimal manual intervention, automatically determining architectural choices (LSTM vs. GRU, attention inclusion, operation types) that would otherwise require extensive

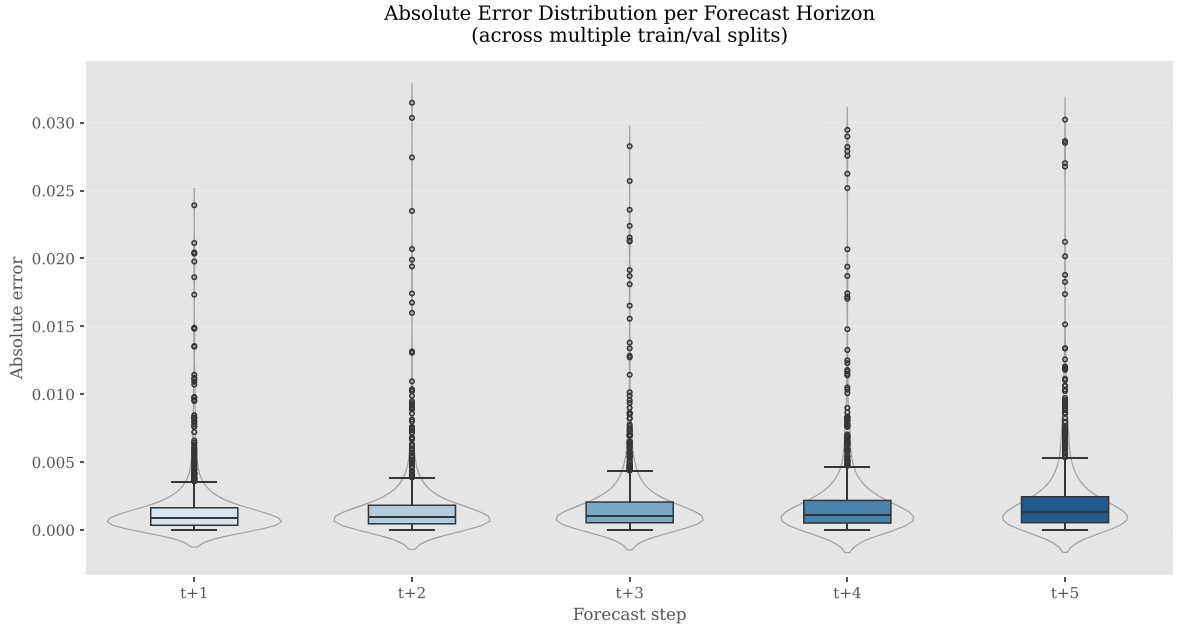


Fig. 21. Distribution of absolute forecast errors per horizon step, shown as violin plots overlaid with box plots. Data are aggregated across multiple train/validation splits and random seeds. The narrowing of the central box and decreasing density of outliers from t+1 to t+5 illustrate that the model maintains good calibration even at longer horizons, with only a moderate increase in dispersion.

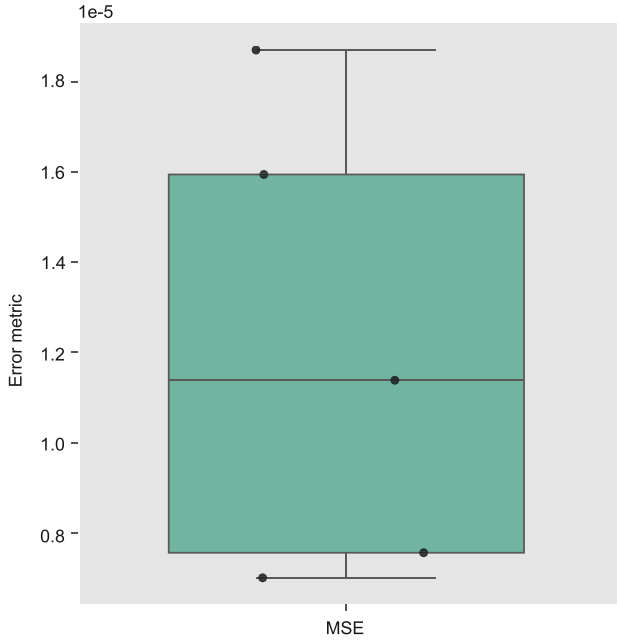


Fig. 22. Box plot showing the spread of final mean squared error (MSE) values obtained across different data splits and random seeds. The narrow interquartile range and limited outlier presence indicate high stability of the selected architecture under varying training conditions.

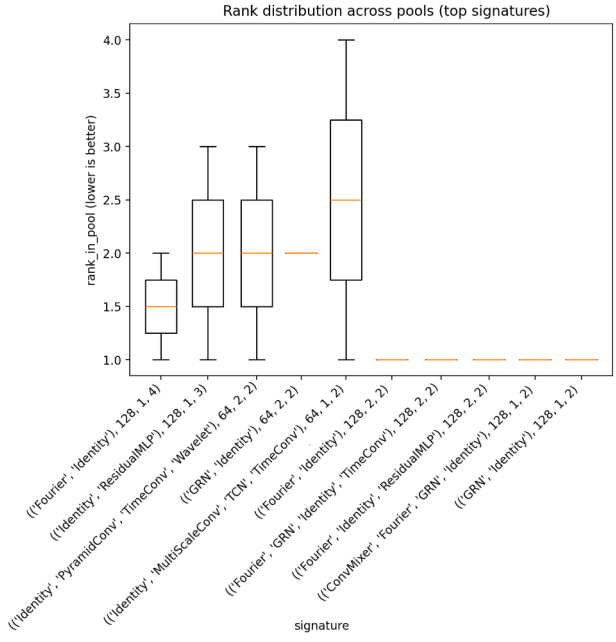


Fig. 23. Rank distribution of top architecture signatures across randomly sampled operator pools. Lower ranks indicate superior performance within a given pool.

empirical validation. This automation reduces development time and domain-specific tuning, facilitating deployment across diverse forecasting applications.

Practical Deployment Considerations: The proposed DARTS framework offers strong engineering feasibility for insulator monitoring in high-voltage power grids, enabling predictive maintenance to avert flashovers. Inference speed supports real-time forecasting: The discovered architectures, leveraging optimal mixed blocks (e.g., efficient GRU/LSTM and streamlined Transformer components via bilevel optimization and zero-cost metrics), remain lightweight and process typical sequences with modest resources (50–200 MB memory). This compatibility suits embedded edge devices like Raspberry Pi in resource-constrained environments. The framework integrates seamlessly with existing IoT sensors and leakage current systems, facilitating hybrid proactive alerts without major retrofits. Given the lightweight optimal blocks, the network enables fast inference for real-time applications.

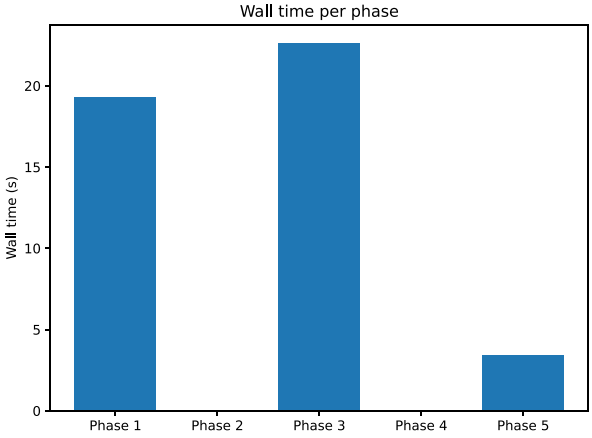


Fig. 24. Wall-clock time breakdown of the multi-fidelity search procedure by phase. Phases that perform repeated candidate evaluation and training dominate the total runtime.

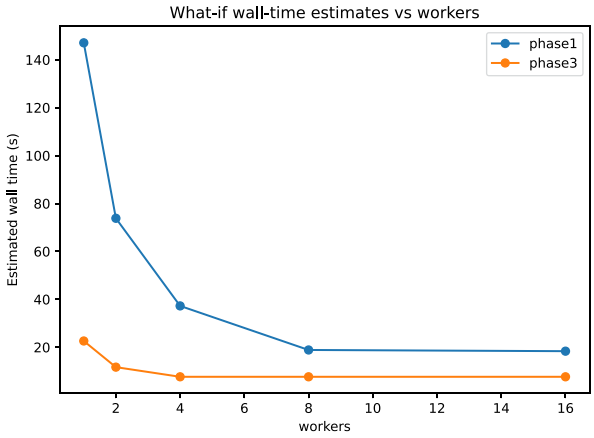


Fig. 25. What-if wall-time estimates versus number of workers for phase 1 (parallelizable screening) and phase 3 (top-k training/derivation). The curves indicate strong initial speedups and diminishing returns at larger worker counts.

Future work can focus on quantization (e.g., INT8 or mixed-precision) to further minimize size, latency, and power use in remote deployments.

Limitations: The primary challenge remains false positive rates in zero-cost evaluation-20% success rate among top-ranked candidates indicates room for improvement in proxy metric fidelity. Additionally, the method's reliance on short evaluation sequences may underestimate long-term dependency modeling benefits of certain architectures (e.g., Transformers). Future work should explore multi-fidelity strategies that combine zero-cost proxies with limited full training to refine candidate selection while maintaining computational efficiency.

A practical limitation of the proposed pipeline is the wall-clock computational cost of the multi-fidelity search procedure. To quantify this overhead, we instrumented the full search loop and recorded the elapsed (wall) time required by each phase (Fig. 24), considering 30 candidates and 8 top-k ones. In our implementation, Phase 1 (candidate generation and zero-cost screening) and Phase 3 (short DARTS training and derivation/evaluation for top-ranked candidates) dominate the runtime budget, whereas the selection steps (Phases 2 and 4) contribute negligibly. This profile is expected because Phases 1 and 3 include repeated model instantiation, forward/backward passes, and validation evaluation, which scale approximately linearly with the number of candidates and the number of training epochs, respectively.

Despite this limitation, the search procedure is amenable to parallel execution, particularly in Phase 1, where candidate evaluations are independent and can be scheduled concurrently. To illustrate the potential gains, Fig. 25 reports *what-if* wall-time estimates as a function of the number of workers, computed by replaying the empirically observed per-candidate runtimes under an idealized parallel scheduler.

The estimates show that increasing the worker count can substantially reduce end-to-end screening time, with diminishing returns at higher levels of parallelism due to load imbalance (heterogeneous candidate runtimes) and practical overheads (e.g., Python scheduling, memory contention, and device synchronization).

Importantly, the achievable speedup depends on the computational backend. When zero-cost evaluation and short training phases are predominantly CPU-bound, thread-level parallelism can yield near-linear speedups until saturating core and memory bandwidth resources. Conversely, when the pipeline is GPU-bound, parallel candidate execution may be limited by device serialization and kernel scheduling, leading to weaker scaling. Therefore, while parallelism mitigates the computational cost, search-time remains a relevant constraint for large candidate pools, long horizons, or repeated runs across datasets. Future work will explore (i) early-stopping and adaptive budgeting strategies within Phase 3, (ii) stronger pruning of the candidate set using calibrated proxies, and (iii) hybrid CPU/GPU scheduling to improve hardware utilization and reduce wall time.

5. Conclusion

This work presented a differentiable neural architecture search framework, leveraging zero-cost metrics, designed for time series forecasting in the context of high-voltage insulator monitoring. The proposed approach integrates a mixed encoder-decoder architecture with LSTM, GRU, and Transformer modules, coupled via a novel cross-attention bridge with learnable temporal bias. This design enables the model to adaptively select the most suitable components based on the input data, improving modeling flexibility.

To address the computational challenges commonly associated with neural architecture search, we introduced a multi-fidelity search strategy that incorporates zero-cost evaluation metrics for rapid candidate screening. This combination allows for efficient exploration of the architecture space without requiring full model training, significantly reducing resource requirements while maintaining high performance.

Our experimental results on real-world leakage current datasets confirm that the discovered architectures outperform traditional manually designed baselines in fault forecasting. The models are able to capture complex temporal patterns and degradation behaviors that are critical for predictive maintenance in power systems.

Looking ahead, future research will explore the extension of this framework to multi-modal sensor data, cross-domain transferability of discovered architectures, and the incorporation of domain-specific inductive biases. These directions aim to further enhance the scalability and generalization of differentiable neural architecture search methods for broader industrial forecasting applications.

CRediT authorship contribution statement

Laio Oriel Seman: Writing – original draft, Software, Methodology, Investigation, Conceptualization; **William Gouvêa Buratto:** Writing – original draft; **Gabriel Villarrubia Gonzalez:** Writing – review & editing, Supervision; **Valderi Reis Quietinho Leithardt:** Writing – review & editing; **Ademir Nied:** Writing – review & editing, Supervision; **Stefano Frizzo Stefenon:** Writing – review & editing.

Data availability

The dataset is publicly available at: https://github.com/SFStefenon/LeakageCurrent

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The APC was supported by the project Self-adaptive platform based on intelligent agents for the optimization and management of operational processes in logistic warehouses (PLAUTON), PID2023-151701OB-C21, funded by MCIN/AEI/10.13039/501100011033/FEDER, EU. Buratto would like to thank UDESC for the financial support for his interuniversity exchange doctorate at the University of Salamanca. Also, the authors would like to thank the Coordination for the Improvement of Higher Education Personnel (CAPES - Brazil) for the scholarship to Buratto. This study was financed (i) in part by CAPES under the doctoral scholarship number 88887.808258/2023-00, and (ii) by Council for Scientific and Technological Development (CNPq) under grant numbers 305910/2024-8 and 307858/2025-1. A realização desta investigação foi parcialmente financiada por fundos nacionais através da FCT - Fundação para a Ciência e Tecnologia, I.P. no âmbito dos projetos UIDB/04466/2025 e UIDP/04466/2025. A realização desta investigação foi parcialmente financiada por fundos nacionais através da FCT - Fundação para a Ciência e Tecnologia, I.P. no âmbito do projeto 16881, LISBOA2030-FEDER-00816400. https://doi.org/10.54499/2023.16583.ICDT.

References

[1] Y. Liu, H. Zong, S. Gao, B.X. Du, Contamination deposition and discharge characteristics of outdoor insulators in fog-haze conditions, Int. J. Electr. Power Energy Syst. 121 (2020) 106176. https://doi.org/10.1016/j.ijepes.2020.106176

[2] M.P. Corso, S.F. Stefenon, G. Singh, M.V. Matsuo, F.L. Perez, V.R.Q. Leithardt, Evaluation of visible contamination on power grid insulators using convolutional neural networks, Electr. Eng. 105 (2023) 3881-3894. https://doi.org/10.1007/s00202-023-01915-2

[3] L. Maraaba, K. Al-Soufi, T. Ssennoga, A.M. Memon, M.Y. Worku, L.M. Alhems, Contamination level monitoring techniques for high-voltage insulators: a review, Energ. 15 (20) (2022) 7656. https://doi.org/10.3390/en15207656

[4] Y. Liu, D. Liu, X. Huang, C. Li, Insulator defect detection with deep learning: a survey, IET Gener. Transm. Distrib. 17 (16) (2023) 3541–3558. https://doi.org/10.1049/gtd2.12916

[5] Z. Zhang, H. Zhang, C. Zhou, X. Ma, Y. Li, R. Liu, A probabilistic neural network assessment method for insulator pollution levels based on infrared images, IEEE Trans. Dielectr. Electr. Insul. 31 (5) (2024) 2711–2720. https://doi.org/10.1109/TDEI.2024.3388378

[6] S. Deng, L. Chen, Y. He, Insulator defect detection from aerial images in adverse weather conditions, Appl. Intell. 55 (6) (2025) 365. https://doi.org/10.1007/s10489-025-06280-0

[7] S.F. Stefenon, G. Singh, B.J. Souza, R.Z. Freire, K.-C. Yow, Optimized hybrid YOLOu-Quasi-ProtoPNet for insulators classification, IET Gener. Transm. Distrib. 17 (15) (2023) 3501–3511. https://doi.org/10.1049/gtd2.12886

[8] S.F. Stefenon, L.O. Seman, B.A. Pavan, R.G. Ovejero, V.R.Q. Leithardt, Optimal design of electrical power distribution grid spacers using finite element method, IET Gener. Transm. Distrib. 16 (9) (2022) 1865–1876. https://doi.org/10.1049/gtd2.12425

[9] M.-R. Halloum, B. Subba Reddy, Superhydrophobic coating for performance enhancement of polymeric outdoor insulators used in HVDC systems, IEEE Trans. Dielectr. Electr. Insul. 32 (5) (2025) 2551–2558. https://doi.org/10.1109/TDEI.2025.3596950

[10] B. Mehmood, R. Atef Ghunem, A. El-Hag, L.-L. Tay, UHF detection of the eroding DC dry-band arcing on silicone rubber insulation, IEEE Trans. Dielectr. Electr. Insul. 32 (3) (2025) 1380–1386. https://doi.org/10.1109/TDEI.2024.3510222

[11] S.F. Stefenon, J.R. Oliveira, A.S. Coelho, L.H. Meyer, Diagnostic of insulators of conventional grid through LabVIEW analysis of FFT signal generated from ultrasound detector, IEEE Lat. Am. Trans. 15 (5) (2017) 884–889. https://doi.org/10.1109/TLA.2017.7910202

[12] L.O. Seman, S.F. Stefenon, V.C. Mariani, L.S. Coelho, Ensemble learning methods using the Hodrick-Prescott filter for fault forecasting in insulators of the electrical power grids, Int. J. Electr. Power Energy Syst. 152 (2023) 109269. https://doi.org/10.1016/j.ijepes.2023.109269

[13] O.E. Gouda, M.M.F. Darwish, K. Mahmoud, M. Lehtonen, T.M. Elkhodragy, Pollution severity monitoring of high voltage transmission line insulators using wireless device based on leakage current bursts, IEEE Access. 10 (2022) 53713–53723. https://doi.org/10.1109/ACCESS.2022.3175515

[14] A. Medeiros, A. Sartori, S.F. Stefenon, L.H. Meyer, A. Nied, Comparison of artificial intelligence techniques to failure prediction in contaminated insulators based on leakage current, J. Intell. Fuzzy Syst. 42 (4) (2022) 3285–3298. https://doi.org/10.3233/JIFS-211126

[15] S. Ma, W. Ding, Y. Zheng, L. Zhou, Z. Yan, J. Xu, Edge-cloud collaboration-driven predictive planning based on LSTM-attention for wastewater treatment, Comput. Ind. Eng. 195 (2024) 110425. https://doi.org/10.1016/j.cie.2024.110425

[16] H. Zang, R. Xu, L. Cheng, T. Ding, L. Liu, Z. Wei, G. Sun, Residential load forecasting based on LSTM fusing self-attention mechanism with pooling, Energy. 229 (2021) 120682. https://doi.org/10.1016/j.energy.2021.120682

[17] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, in: International Conference on Learning Representations, 7, ICLR, New Orleans, USA, 2019, pp. 1–13.

[18] X. Chen, R. Xie, J. Wu, C.-J. Hsieh, DrNAS: dirichlet neural architecture search, in: International Conference on Learning Representations, 9, ICLR, 2021, pp. 1–17.

[19] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: decomposition transformers with auto-correlation for long-term series forecasting, in: Advances in Neural Information Processing Systems (NeurIPS), 34, 2021, pp. 22419–22430.

[20] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, 35, 2021, pp. 11106–11115.

[21] J.-E. Chiu, S.-Z. Fu, On-line recognition of mixture control chart patterns using hybrid CNN and LSTM for auto-correlated processes, Comput. Ind. Eng. 198 (2024) 110674. https://doi.org/10.1016/j.cie.2024.110674

[22] S. Hong, M. Kang, J. Kim, J. Baek, Sequential application of denoising autoencoder and long-short recurrent convolutional network for noise-robust remaining-useful-life prediction framework of lithium-ion batteries, Comput. Ind. Eng. 179 (2023) 109231. https://doi.org/10.1016/j.cie.2023.109231

[23] Y. Wang, R.W. Liu, J. Liu, L. Yang, Y. Liu, P. He, Co-prediction of multiple transportation demands in Ro-Ro terminal based on dates fine-grained temporal fusion transformer, Comput. Ind. Eng. 208 (2025) 111373. https://doi.org/10.1016/j.cie.2025.111373

[24] S.M. Gbashi, O.O. Olatunji, P.A. Adedeji, N. Madushele, Pso-optimised autoencoder for fault prediction in wind turbine planet carrier bearing, Results Eng. 26 (2025) 104844. https://doi.org/10.1016/j.rineng.2025.104844

[25] C.-Y. Lin, Y.-C. Tseng, W.-S. Yao, Integrating spatiotemperporal features into fault prediction using a multi-dimensional method, Results Eng. 27 (2025) 105856. https://doi.org/10.1016/j.rineng.2025.105856

[26] M. Parvin, H. Yousefi, B. Mohammadi-Ivatloo, Photovoltaic fault detection algorithm using ensemble learning enhanced with deep neural network feature engineering, Results Eng. 27 (2025) 106491. https://doi.org/10.1016/j.rineng.2025.106491

[27] P. Rengasamy, R. Rajesh, Explainable artificial intelligence framework for wind turbine fault detection using random forest - extreme gradient boosting hybrid model, Results Eng. 28 (2025) 107446. https://doi.org/10.1016/j.rineng.2025.107446

[28] J.P.M. Carvalho, S.F. Stefenon, K.-C. Yow, R.G. Ovejero, V.R.Q. Leithardt, N-BEATS neural network applied for insulator fault prediction considering EMD methods, in: International Conference on Electrical, Computer, Communications and Mechatronics Engineering, 5, IEEE, Zanzibar, Tanzania, 2025, pp. 1–6. https://doi.org/10.1109/ICECCME64568.2025.11277686

[29] W. Zhao, M. Xu, X. Cheng, Z. Zhao, An insulator in transmission lines recognition and fault detection model based on improved Faster RCNN, IEEE Trans. Instrum. Meas. 70 (2021) 1–8. https://doi.org/10.1109/TIM.2021.3112227

[30] S.F. Stefenon, L.O. Seman, G. Singh, K.-C. Yow, Enhanced insulator fault detection using optimized ensemble of deep learning models based on weighted boxes fusion, Int. J. Electr. Power Energy Syst. 168 (2025) 110682. https://doi.org/10.1016/j.ijepes.2025.110682

[31] A.A. Salem, K.Y. Lau, Z. Abdul-Malek, N. Mohammed, A.M. Al-Shaalan, A.A. Al-Shamma'a, H.M.H. Farh, Polymeric insulator conditions estimation by using leakage current characteristics based on simulation and experimental investigation, Polymers 14 (4) (2022) 737. https://doi.org/10.3390/polym14040737

[32] L. Singh, A. Alam, K.V. Kumar, D. Kumar, P. Kumar, Z.A. Jaffery, Design of thermal imaging-based health condition monitoring and early fault detection technique for porcelain insulators using machine learning, Environ. Technol. Innov. 24 (2021) 102000. https://doi.org/10.1016/j.eti.2021.102000

[33] H. Ji, X. Zhang, Y. Guo, S. Xiao, G. Wu, Evaluation method for the UV aging state of composite insulators based on hyperspectral characteristic, IEEE Trans. Instrum. Meas. 72 (2023) 1–9. https://doi.org/10.1109/TIM.2023.3248093

[34] B. Zhang, S. Shu, C. Chen, X. Wang, J. Xu, C. Fang, Composite insulator defect identification method based on acoustic–electric feature fusion and MMSAE network, Energ. 16 (13) (2023) 4906. https://doi.org/10.3390/en16134906

[35] S.F. Stefenon, L.O. Seman, A.C.R. Klaar, R.G. Ovejero, V.R.Q. Leithardt, Hypertuned-YOLO for interpretable distribution power grid fault location based on EigenCAM, Ain Shams Eng. J. 15 (6) (2024) 102722. https://doi.org/10.1016/j.asej.2024.102722

[36] N.F. Sopelsa Neto, S.F. Stefenon, L.H. Meyer, R.G. Ovejero, V.R.Q. Leithardt, Fault prediction based on leakage current in contaminated insulators using enhanced time series forecasting models, Sensors 22 (16) (2022) 6121. https://doi.org/10.3390/s22166121

[37] S.F. Stefenon, L.O. Seman, N.F. Sopelsa Neto, L.H. Meyer, V.C. Mariani, C.L. dos Santos, Group method of data handling using Christiano-Fitzgerald random walk filter for insulator fault prediction, Sensors 23 (13) (2023) 6118. https://doi.org/10.3390/s23136118

[38] A.C.R. Klaar, S.F. Stefenon, L.O. Seman, V.C. Mariani, C.L. dos Santos, Optimized EWT-Seq2Seq-LSTM with attention mechanism to insulators fault prediction, Sensors 23 (6) (2023) 3202. https://doi.org/10.3390/s23063202

[39] H. Jin, Y. Zhang, Y. Jia, Z. Yuan, Y. Tu, Condition diagnosis of composite insulator based on knowledge graph, IEEE Trans. Dielectr. Electr. Insul. 32 (1) (2025) 28–35. https://doi.org/10.1109/TDEI.2024.3510219

[40] S. Fang, S. Shu, B. Zhang, J. Xu, C. Fang, X. Wang, An ultrasonic guided waves-based assessing method for decay-like degrees of composite insulator core rods, IEEE Trans. Instrum. Meas. 74 (2025) 1–13. https://doi.org/10.1109/TIM.2025.3565104

[41] A. Lutfi, A. El-Hag, K. Shaban, Nonintrusive ultrasonic sensing and deep learning for outdoor ceramic insulator assessment, IEEE Trans. Dielectr. Electr. Insul. 31 (6) (2024) 2993–3000. https://doi.org/10.1109/TDEI.2024.3403072

[42] K. Sit, A.K. Das, D. Mukherjee, N. Haque, S. Deb, A.K. Pradhan, S. Dalai, B. Chatterjee, Condition monitoring of overhead polymeric insulators employing hyperbolic window Stockwell transform of surface leakage current signals, IEEE Sens. J. 21 (9) (2021) 10957–10964. https://doi.org/10.1109/JSEN.2021.3061797

[43] D. Maadjoudj, O. Kherif, A. Mekhaldi, M. Teguar, Features characterizing the surface state of HV insulator glass model under desert pollution, IEEE Trans. Dielectr. Electr. Insul. 28 (6) (2021) 1964–1972. https://doi.org/10.1109/TDEI.2021.009739

[44] Y. Cheng, D. Liu, Adin-DETR: adapting detection transformer for end-to-end real-time power line insulator defect detection, IEEE Trans. Instrum. Meas. 73 (2024) 1–11. https://doi.org/10.1109/TIM.2024.3420265

[45] H. Wu, J. Wang, D. Nan, Q. Cui, W. Li, Fault location and fault cause identification method for transmission lines based on pose normalized multioutput convolutional nets, IEEE Trans. Instrum. Meas. 74 (2025) 1–12. https://doi.org/10.1109/TIM.2024.3488157

[46] S. Hao, B. An, X. Ma, X. Sun, T. He, S. Sun, PKAMNet: a transmission line insulator parallel- Gap fault detection network based on prior knowledge transfer and attention mechanism, IEEE Trans. Power Deliv. 38 (5) (2023) 3387–3397. https://doi.org/10.1109/TPWRD.2023.3274823

[47] U. Shafique, S. Muhammad Alam, U. Rashid, W. Javed, H. Anwaar, M. Shah Zeb, T. Ahmad, U. Imtiaz, F. Nzanywayingoma, Infrared thermography-based insulator fault classification via unsupervised clustering and semi-supervised learning, IEEE Access. 12 (2024) 180781–180791. https://doi.org/10.1109/ACCESS.2024.3404930

[48] S.F. Stefenon, J.P. Americo, L.H. Meyer, R.B. Grebogi, A. Nied, Analysis of the electric field in porcelain pin-type insulators via finite elements software., IEEE Lat. Am. Trans. 16 (10) (2018) 2505–2512. https://doi.org/10.1109/TLA.2018.8795129

[49] L.O. Seman, L.S. Aquino, S.F. Stefenon, K.-C. Yow, V.C. Mariani, L. dos Santos Coelho, Simultaneously anomaly detection and forecasting for predictive maintenance using a zero-cost differentiable architecture search-based network, Comput. Ind. Eng. 208 (2025) 111412. https://doi.org/10.1016/j.cie.2025.\protect\penalty-\@M111412

[50] Y. Xue, X. Han, Z. Wang, Self-adaptive weight based on dual-attention for differentiable neural architecture search, IEEE Trans. Ind. Inf. 20 (4) (2024) 6394–6403. https://doi.org/10.1109/TII.2023.3348843

[51] W. Wang, X. Zhang, H. Cui, H. Yin, Y. Zhang, FP-DARTS: fast parallel differentiable neural architecture search for image classification, Pattern Recognit. 136 (2023) 109193. https://doi.org/10.1016/j.patcog.2022.109193

[52] H. Liu, K. Simonyan, Y. Yang, DARTS: differentiable architecture search, 2 (2018) 1–13. https://doi.org/10.48550/arXiv.1806.09055

[53] R. Jie, J. Gao, Differentiable neural architecture search for high-dimensional time series forecasting, IEEE Access. 9 (2021) 20922–20932. https://doi.org/10.1109/ACCESS.2021.3055555

[54] D. Deng, M. Lindauer, Optimizing time series forecasting architectures: a hierarchical neural architecture search approach, 2 (2024) 1–31. https://doi.org/10.48550/arXiv.2406.05088

[55] D. Levchenko, E. Rappos, S. Ataee, B. Nigro, S. Robert-Nicoud, Chain-structured neural architecture search for financial time series forecasting, Int. J. Data Sci. Analyt. 20 (4) (2025) 3727–3736. https://doi.org/10.1007/s41060-024-00690-y

[56] Y. Liu, X. Li, Y. Hu, Differentiable neural architecture search for domain adaptation in fault diagnosis, Mech. Syst. Signal Process. 202 (2023) 110639. https://doi.org/10.1016/j.ymssp.2023.110639

[57] J.B. Thomas, K.V. Shihabudheen, Neural architecture search algorithm to optimize deep transformer model for fault detection in electrical power distribution systems, Eng. Appl. Artif. Intell. 120 (2023) 105890. https://doi.org/10.1016/j.engappai.2023.105890

[58] Q. Jing, J. Yan, Y. Wang, R. He, L. Lu, A novel differentiable neural network architecture automatic search method for GIS partial discharge pattern recognition, Meas. 195 (2022) 111154. https://doi.org/10.1016/j.measurement.2022.111154

[59] H.R. Sezavar, S. Hasanzadeh, N. Fahimi, A novel hybrid mathematical deep learning technique for early warning of flashover in composite insulators, Sci. Rep. 15 (2025) 33448. https://doi.org/10.1038/s41598-025-19151-y

[60] Y. Zhang, H. Su, R. Wang, J. Deng, Y. Wang, W. Guo, R. Li, Short-term forecast method of wind power output based on multi-scale CNN-LSTM in extreme weather, Int. J. Electr. Power Energy Syst. 172 (2025) 111191. https://doi.org/10.1016/j.ijepes.2025.111191

[61] D. Li, G. Sun, S. Miao, Y. Gu, Y. Zhang, S. He, A short-term electric load forecast method based on improved sequence-to-sequence GRU with adaptive temporal dependence, Int. J. Electr. Power Energy Syst. 137 (2022) 107627. https://doi.org/10.1016/j.ijepes.2021.107627

[62] A. Ahmad, X. Xiao, H. Mo, D. Dong, TFTformer: a novel transformer - based model for short-term load forecasting, Int. J. Electr. Power Energy Syst. 166 (2025) 110549. https://doi.org/10.1016/j.ijepes.2025.110549

[63] M.I. Hossain, H.Z. Anonto, T. Riyad, A. Shufian, M.S. Hossain, B.B. Pathik, Adaptive fault diagnosis in power transmission lines using deep learning and LSTM autoencoders for enhancing grid reliability, Int. J. Electr. Power Energy Syst. 174 (2026) 111458. https://doi.org/10.1016/j.ijepes.2025.111458

[64] Y. Ma, X. Zhang, W. Dai, C. Zhang, Intelligent fault diagnosis for offshore wind turbine gearbox: a hybrid framework integrating enhanced VMD and GRU, Int. J. Electr. Power Energy Syst. 173 (2025) 111402. https://doi.org/10.1016/j.ijepes.2025.111402

[65] E.C. da Silva, E.C. Finardi, S.F. Stefenon, Enhancing hydroelectric inflow prediction in the Brazilian power system: a comparative analysis of machine learning models and hyperparameter optimization for decision support, Electr. Power Syst. Res. 230 (2024) 110275. https://doi.org/10.1016/j.epsr.2024.110275

[66] P. Zhao, W. Hu, D. Cao, Z. Zhang, W. Liao, Z. Chen, Q. Huang, Enhancing multivariate, multi-step residential load forecasting with spatiotemporal graph attention-enabled transformer, Int. J. Electr. Power Energy Syst. 160 (2024) 110074. https://doi.org/10.1016/j.ijepes.2024.110074

[67] W. Chen, Z. Zhang, D. Tang, C. Liu, Y. Gui, Q. Nie, Z. Zhao, Probing an LSTM-PPO-based reinforcement learning algorithm to solve dynamic job shop scheduling problem, Comput. Ind. Eng. 197 (2024) 110633. https://doi.org/10.1016/j.cie.2024.110633

[68] W. Fang, Y. Guo, W. Liao, S. Huang, N. Yang, J. Liu, A parallel gated recurrent units (P-GRUs) network for the shifting lateness bottleneck prediction in make-to-order production system, Comput. Ind. Eng. 140 (2020) 106246. https://doi.org/10.1016/j.cie.2019.106246

[69] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, Z. Li, DARTS+: improved differentiable architecture search with early stopping, (2019) 1–15. https://doi.org/10.48550/arXiv.1909.06035

[70] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, K. He, Accurate, large minibatch SGD: training ImageNet in 1 h, 2 (2017) 1–12. https://doi.org/10.48550/arXiv.1706.02677

[71] I. Loshchilov, F. Hutter, SGDR: stochastic gradient descent with warm restarts, in: International Conference on Learning Representations (ICLR), 5, 2017, pp. 1–16.

[72] J. Shin, K. Park, D.-K. Kang, TA-DARTS: temperature annealing of discrete operator distribution for effective differential architecture search, Appl. Sci. 13 (18) (2023) 10138. https://doi.org/10.3390/app131810138

[73] J. Zhang, Z. Ding, Small temperature is all you need for differentiable architecture search, 1 (2023) 1–12. https://doi.org/10.48550/arXiv.2306.06855

[74] E. Jang, S. Gu, B. Poole, Categorical reparameterization with Gumbel-softmax, in: International Conference on Learning Representations (ICLR), 5, 2017, pp. 1–13.

[75] C.J. Maddison, A. Mnih, Y.W. Teh, The concrete distribution: a continuous relaxation of discrete random variables, in: International Conference on Learning Representations (ICLR), 5, 2017, pp. 1–20.

[76] H. Tanaka, D. Kunin, D.L. Yamins, S. Ganguli, Pruning neural networks without any data by iteratively conserving synaptic flow, in: Advances in Neural Information Processing Systems, 33, 2020, pp. 6377–6389.

[77] L. Wang, Y. Zhao, Y. Jinnai, Y. Tian, R. Fonseca, Picking winning tickets before training by preserving gradient flow, in: International Conference on Learning Representations, 8, ICLR, Addis Ababa, Ethiopia, 2020, pp. 1–11.

[78] J. Mellor, J. Turner, A. Storkey, E.J. Crowley, Neural architecture search without training, in: International Conference on Machine Learning, 139, PMLR, Vienna, Austria, 2021, pp. 7588–7598.

[79] J. Meng, M. Yue, D. Diallo, Nonlinear extension of battery constrained predictive charging control with transmission of Jacobian matrix, Int. J. Electr. Power Energy Syst. 146 (2023) 108762. https://doi.org/10.1016/j.ijepes.2022\protect\penalty-\@M108762

[80] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, R. Jin, Zen-NAS: a zero-Shot NAS for high-performance image recognition, in: International Conference on Computer Vision, IEEE, Montreal, Canada, 2021, pp. 337–346. https://doi.org/10.1109/ICCV48922.2021.00040

[81] L. Theis, I. Korshunova, A. Tejani, F. Huszár, Faster gaze prediction with dense networks and Fisher pruning, 2 (2018) 1–18. https://doi.org/10.48550/arXiv.1801.05787

[82] N. Lee, T. Ajanthan, P.H.S. Torr, SNIP: single-shot network pruning based on connection sensitivity, in: International Conference on Learning Representations, 7, ICLR, New Orleans, USA, 2019, pp. 1–15.

[83] A. Krishnan, M.A. Baba, R. Mehta, R. Duggal, Predicting neural network accuracy from weights, 4 (2020) 1–18. https://doi.org/10.48550/arXiv.2002.11448

[84] M.S. Abdelfattah, A. Mehrotra, L. Dudziak, N.D. Lane, Zero-cost proxies for lightweight NAS, in: International Conference on Learning Representations, 9, ICLR, Vienna, Austria, 2021, pp. 1–17.

[85] K. Jing, L. Chen, J. Xu, An architecture entropy regularizer for differentiable neural architecture search, Neural Netw. 158 (2023) 111–120. https://doi.org/10.1016/j.neunet.2022.11.015

[86] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, W. Ouyang, β-DARTS: beta-decay regularization for differentiable architecture search, in: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1, 2022, pp. 10864–10873. https://doi.org/10.1109/CVPR52688.2022.01060

[87] X. Chu, T. Zhou, B. Zhang, J. Li, Fair DARTS: eliminating unfair advantages in differentiable architecture search, in: European Conference on Computer Vision (ECCV), Springer, 2020, pp. 465–480. https://doi.org/10.1007/978-3-030-58555-6_28

[88] Z. Lu, Q. Wang, J. You, H. Jiang, Q. Liu, Y. Wang, M. Tan, DU-DARTS: Decreasing the uncertainty of differentiable architecture search, in: British Machine Vision Conference (BMVC), 2021, pp. 1–13. https://doi.org/10.5244/c.35.54
[89] Y. Zhang, Q. Yao, K. Yue, X.-S. Chen, Y. Zhang, C. Ma, Discretization-aware architecture search, Pattern Recognit. 120 (2021) 108161. https://doi.org/10.1016/j.patcog.2021.108186
[90] Y. Chen, Z. Liu, S.-H. Kim, N.I. Cho, Regularizing differentiable architecture search with smooth activation, 1 (2025) 1–22. https://doi.org/10.48550/arXiv.2504.16306
[91] X. Chen, C.-J. Hsieh, Stabilizing differentiable architecture search via perturbation-based regularization, in: International Conference on Machine Learning (ICML), 37, PMLR, 2020, pp. 1554–1565.
[92] S. Movahedi, M. Adabinejad, A. Imani, A. Keshavarz, M. Dehghani, A. Shakery, B.N. Araabi, Λ-DARTS: mitigating performance collapse by harmonizing operation selection among cells, in: International Conference on Learning Representations (ICLR), 11, 2023, pp. 1–27.
[93] IEC-60507, Artificial pollution tests on high-voltage ceramic and glass insulators to be used on A.C. systems, International Standard, International Electrotechnical Commission (IEC), Geneva, Switzerland, 2013.
[94] B. Lim, S.Ö. Arık, N. Loeff, T. Pfister, Temporal fusion transformers for interpretable multi-horizon time series forecasting, Int. J. Forecast. 37 (4) (2021) 1748–1764. https://doi.org/10.1016/j.ijforecast.2021.03.012
[95] Y. Nie, A time series is worth 64Words: long-term forecasting with transformers, 1 (2022) 1–24. https://doi.org/10.48550/arXiv.2211.14730
[96] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J.Y. Zhang, J. Zhou, TimeMixer: decomposable multiscale mixing for time series forecasting, 1 (2024) 1–28. https://doi.org/10.48550/arXiv.2405.14616
[97] Y. Wang, H. Wu, J. Zhou, S. Wang, T. Hu, H. Luo, L. Ma, J. Zhou, TimeXer: empowering transformers for time series forecasting with exogenous variables, 4 (2024) 1–30. https://doi.org/10.48550/arXiv.2402.19072
[98] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, Transformer: inverted transformers are effective for time series forecasting, in: International Conference on Learning Representations (ICLR), 12, 2024, pp. 1–25.
[99] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, FEDformer: frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning (ICML), 39, PMLR, Baltimore, Maryland, USA, 2022, pp. 27268–27286.
[100] B.N. Oreshkin, D. Carpov, N. Chapados, Y. Bengio, N-BEATS: neural basis expansion analysis for interpretable time series forecasting, Int. Conf. Learn. Represent. (ICLR) 8 (2020) 1–31.
[101] D. Salinas, V. Flunkert, J. Gasthaus, T. Januschowski, DeepAR: probabilistic forecasting with autoregressive recurrent networks, Int. J. Forecast. 36 (3) (2020) 1181–1191. https://doi.org/10.1016/j.ijforecast.2019.07.001
[102] W.G. Buratto, R.N. Muniz, R. Cardoso, A. Nied, C.T. da Costa, G.V. Gonzalez, Hybrid group method of data handling for time-series forecasting of thermal generation dispatch in electrical power systems, Electr. Eng. 107 (2025) 13929-13945. https://doi.org/10.1007/s00202-025-03242-0
[103] N.W. Branco, M.S.M. Cavalca, R.G. Ovejero, Bootstrap aggregation with Christiano-Fitzgerald random walk filter for fault prediction in power systems, Electr. Eng. 106 (3) (2024) 3657–3670. https://doi.org/10.1007/s00202-023-02146-1
[104] F. Rodríguez, I. Azcárate, J. Vadillo, A. Galarza, Forecasting intra-hour solar photovoltaic energy by assembling wavelet based time-frequency analysis with deep learning neural networks, Int. J. Electr. Power Energy Syst. 137 (2022) 107777. https://doi.org/10.1016/j.ijepes.2021.107777