STrans: Spontaneous Architecture Evolution for Adaptive Time Series Forecasting

Haoyi Jia

University of New South Wales
z5396170@zmail.unsw.edu.au

Abstract

While Transformers have revolutionized time series forecasting, they remain trapped by manual architecture design—every model uses the same attention mechanism, normalization, and activation choices. What if we could automatically discover the perfect architectural recipe for each dataset? This work introduces STrans (Spontaneous Transformer), a comprehensive neural architecture search framework for time series Transformers that simultaneously explores attention variants, normalization techniques, activation functions, and encoding operations. Using differentiable architecture search, STrans automatically discovers architectures that outperform manually designed baselines. However, the experiments reveal a surprising and counterintuitive finding: complex searched architectures often fail catastrophically, while simpler configurations generalize better. This "search overfitting" phenomenon challenges fundamental assumptions about automated architecture design in time series domains. The work not only advances automated model design but uncovers critical insights that will reshape how we think about neural architecture search for temporal data.

Introduction

Time series forecasting is fundamental to critical applications from financial trading and energy management to healthcare monitoring and climate prediction. Recent Transformer advances have significantly improved forecasting performance, with models like Informer (Zhou et al. 2021), Autoformer (Wu et al. 2021), and iTransformer (Liu et al. 2023) demonstrating remarkable capabilities in capturing long-range temporal dependencies. However, these models rely on manual architecture engineering, requiring extensive domain expertise and iterative experimentation to determine optimal combinations of attention mechanisms, normalization techniques, and activation functions for specific time series characteristics.

Neural Architecture Search (NAS) has emerged as a powerful paradigm for automated model design, achieving notable success across various domains (Liu, Simonyan, and Yang 2019). While recent surveys (Wen et al. 2022) have explored NAS applications in Transformers and time series, existing approaches often treat temporal data as generic sequential inputs, overlooking unique time series forecasting challenges: non-stationarity, seasonality, trend decomposition, and multi-horizon prediction with autoregressive decoding strategies.

Challenges. Applying NAS to time series forecasting presents domain-specific challenges beyond standard sequence modeling. Current Transformer designs rely on manual architecture selection, requiring extensive domain expertise to determine optimal combinations of attention mechanisms, normalization techniques, and activation functions for different temporal characteristics. Time series data exhibits complex patterns like seasonality and non-stationarity that demand specialized architectural components, while forecasting tasks require design considerations that differ fundamentally from classification. The intricate interactions between temporal modeling components make automated search prone to overfitting, particularly when architectures must generalize across diverse time series domains.

Method. To address these challenges, this work presents an automated architecture search framework tailored for multivariate time series forecasting. The framework automatically discovers optimal architectural combinations through differentiable architecture search over curated operation sets including attention variants, normalization strategies, encoding operations, and activation functions. Using mixed operations, the system efficiently explores the architectural space and consistently achieves superior performance across diverse temporal domains, demonstrating that automated search can uncover domain-specific design principles that outperform manually-designed architectures.

Contributions. This work makes several key contributions: (1) A comprehensive automated architecture search framework for time series Transformers systematically incorporating domain knowledge about temporal patterns and forecasting requirements. (2) Extensive experiments across diverse temporal domains show automatically discovered architectures consistently outperform manually-designed state-of-the-art methods. (3) Identification and analysis of critical "search overfitting" phenomenon in time series NAS, where complex automatically-discovered architectures underperform simpler baselines, with proposed mitigation strategies. (4) Detailed analysis of automated search provides practical guidelines for deploying architecture search in time series forecasting applications.

Related Work

Time Series Forecasting with Transformers

Transformer architectures have achieved remarkable progress in time series forecasting. Informer (Zhou et al. 2021) introduced ProbSparse attention for efficient long sequence processing, while Autoformer (Wu et al. 2021) incorporated trend-seasonal decomposition within attention mechanisms. FEDformer (Zhou et al. 2022) leveraged frequency domain transformations, and Stationary (Liu et al. 2022) addressed non-stationarity issues. Most notably, iTransformer (Liu et al. 2023) achieved breakthrough performance by inverting attention to operate across variates rather than time steps.

Despite these advances, existing approaches rely on manual architecture design, where researchers empirically select attention mechanisms, normalization techniques, and activation functions. This limits exploration of potentially superior architectural combinations across diverse temporal domains.

Neural Architecture Search

Neural Architecture Search has revolutionized automated model design. Early approaches like NASNet (Zoph et al. 2017) and ENAS (Pham et al. 2018) demonstrated that discovered architectures could outperform manual designs, though with high computational costs. DARTS (Liu, Simonyan, and Yang 2019) reduced search costs through continuous relaxation of discrete search spaces. Subsequent work like GDAS (Dong and Yang 2019) and PC-DARTS (Xu et al. 2020) improved efficiency and addressed memory limitations.

However, NAS application to time series remains limited. While surveys (Chitty-Venkata et al. 2022; Wen et al. 2022) explored NAS in Transformers and time series, existing approaches treat temporal data as generic sequences, overlooking domain-specific characteristics like seasonality and non-stationarity. Recent work like TransNAS-TSAD (Hag, Lee, and Rizzo 2025) began exploring NAS for time series anomaly detection, but comprehensive architecture search for forecasting transformers remains unexplored.

STrans Architecture

Structure Overview

STrans introduces a searchable transformer architecture that automatically discovers optimal combinations of attention mechanisms, normalization techniques, encoding operations, and activation functions for time series forecasting. The architecture employs mixed operations where each component is a weighted combination of multiple candidate operations.

Figure 1 shows the complete STrans architecture with its searchable components and dual residual pathways. Algorithm 1 in Appendix Algorithm presents the overall procedure of the framework.

Mixed Operations Framework

The core innovation of STrans lies in its mixed operations framework, which enables differentiable architecture search across multiple component types. Each mixed operation $\mathcal{M}$ represents a weighted combination of candidate operations:

$$\mathcal{M}(x) = \sum_{i=1}^{N} w_i \cdot \mathcal{O}_i(x), \tag{1}$$

where $\mathcal{O}_i$ denotes the i-th candidate operation, w_i are learnable architecture weights obtained through softmax normalization, and N is the number of candidate operations.

The STrans encoder layer integrates multiple mixed operations within a dual residual pathway design:

$$x = \text{MixedEnc1}(\text{residual}) + \text{MixedAttention}(x), \tag{2}$$

$$x = \text{MixedNorm1}(x), \tag{3}$$

$$x = \text{MixedEnc2}(\text{residual}) + \text{FFN}(x), \tag{4}$$

$$x = \text{MixedNorm2}(x). \tag{5}$$

Here, **MixedAttention** explores different attention mechanisms for capturing temporal dependencies, **MixedEnc1** and **MixedEnc2** represent searchable encoding operations that replace standard identity residual connections, and **MixedNorm1** and **MixedNorm2** enable exploration of various normalization strategies. The **FFN** component incorporates searchable activation functions within its feed-forward computation.

This framework automatically discovers optimal architectural combinations tailored to specific time series characteristics, moving beyond manual design limitations.

Attention Operations

This work explores five attention mechanisms that capture token interactions through different computational paradigms. Each mechanism operates on query $Q \in \mathbb{R}^{B\times L\times H\times D}$, key $K \in \mathbb{R}^{B\times S\times H\times D}$, and value $V \in \mathbb{R}^{B\times S\times H\times D}$ matrices, where B denotes batch size, L and S represent sequence lengths, H is the number of attention heads, and D is the head dimension.

FullAttention (Vaswani et al. 2017): The standard Transformer attention mechanism computes attention scores through scaled dot-product operations:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{D}}\right)V, \tag{6}$$

where the scaling factor $\sqrt{D}$ prevents the softmax function from saturating in high-dimensional spaces.

ConcatAttention (Bahdanau, Cho, and Bengio 2015): This mechanism applies separate linear transformations to queries and keys, enabling asymmetric relationship modeling:

$$s_j^t = v_c^\top \tanh(W_c^1 q_j + W_c^2 k_t), \tag{7}$$

where $W_c^1, W_c^2 \in \mathbb{R}^{D\times D}$ are learnable transformation matrices and $v_c \in \mathbb{R}^D$ is a learnable attention vector.

BilinearAttention (Luong, Pham, and Manning 2015): This approach models query-key interactions through a learned bilinear transformation:

$$s_j^t = q_j^\top W_b k_t, \tag{8}$$

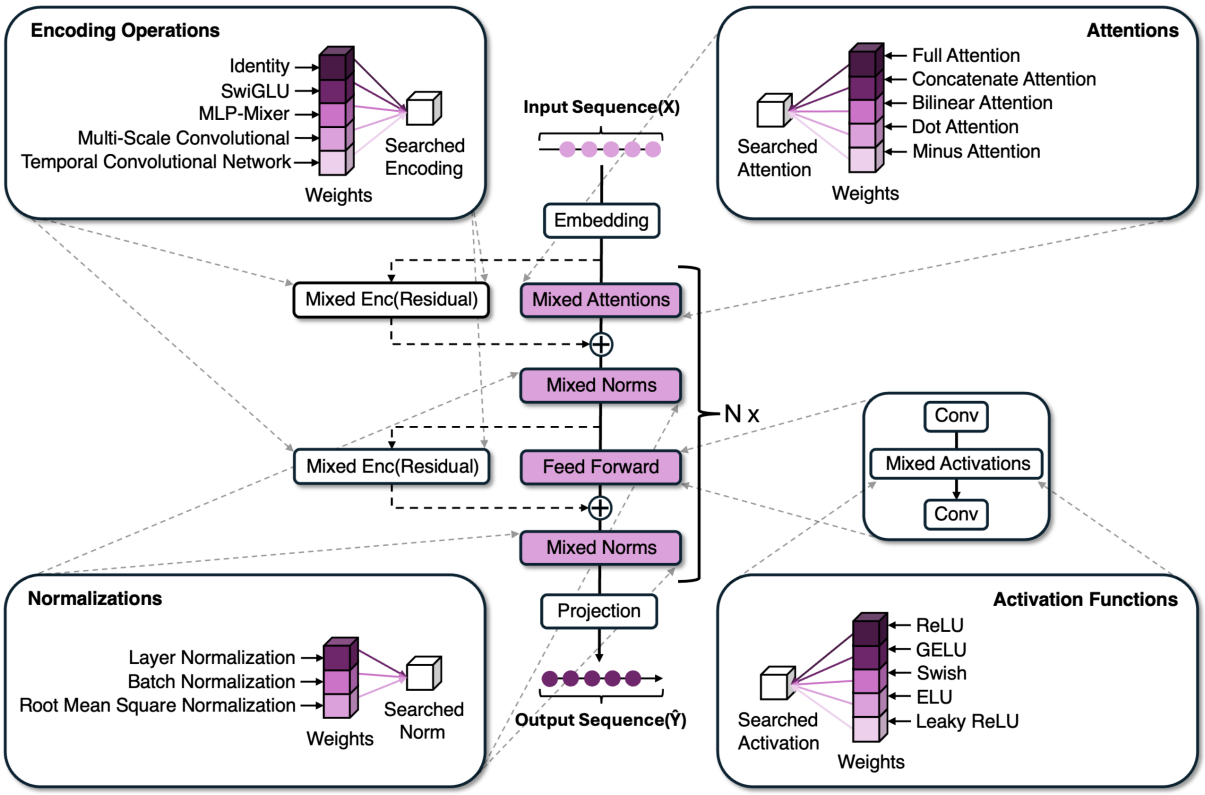


Figure 1: STrans Architecture: Complete overview showing mixed operations, dual residual pathways, and searchable components including attention mechanisms, encoding operations, normalization techniques, and activation functions.

where $W_b \in \mathbb{R}^{D\times D}$ enables more expressive interactions compared to standard dot-product attention.

DotAttention (Luong, Pham, and Manning 2015): This mechanism emphasizes element-wise interactions between queries and keys:

$$s_j^t = v_d^\top \tanh(W_d(q_j \odot k_t)), \tag{9}$$

where $\odot$ denotes element-wise multiplication, $W_d \in \mathbb{R}^{D\times D}$, and $v_d \in \mathbb{R}^D$.

MinusAttention (Gehring et al. 2017): This variant focuses on the differences between query and key representations:

$$s_j^t = v_m^\top \tanh(W_m(q_j - k_t)), \tag{10}$$

where $W_m \in \mathbb{R}^{D\times D}$ and $v_m \in \mathbb{R}^D$ are learnable parameters that emphasize dissimilarity patterns in the feature space.

Alternative Encoding Operations

Standard Transformers employ identity residual connections to facilitate gradient flow (He et al. 2016). STrans expands this design space by exploring alternative encoding operations that can enhance feature extraction and model expressiveness for temporal data:

Identity Encoding: The standard residual connection: $\text{Enc}_{\text{identity}}(x) = x$.

SwiGLU Encoding (Shazeer 2020): Applies gated linear units for enhanced non-linearity:

$$\text{Enc}_{\text{swiglu}}(x) = \text{SiLU}(W_1 x) \odot (W_2 x), \tag{11}$$

where $\text{SiLU}(z) = z \cdot \sigma(z)$ and $W_1, W_2 \in \mathbb{R}^{D\times D}$.

MLP-Mixer Encoding (Tolstikhin et al. 2021): Utilizes simple multi-layer perceptrons:

$$\text{Enc}_{\text{mixer}}(x) = W_2 \cdot \text{GELU}(W_1 x + b_1) + b_2. \tag{12}$$

Multi-Scale Convolutional Encoding (Yu and Koltun 2016): Captures temporal patterns at different scales through parallel convolutions with varying kernel sizes k_i:

$$\text{Enc}_{\text{conv}}(x) = \text{Concat}\left[\text{Conv}_{k_1}(x), \text{Conv}_{k_2}(x), \ldots, \text{Conv}_{k_n}(x)\right]. \tag{13}$$

Temporal Convolutional Network Encoding (Bai, Kolter, and Koltun 2018): Employs dilated causal convolutions for long-range temporal dependency modeling:

$$\text{Enc}_{\text{tcn}}(x) = \text{Residual}(\text{Conv}_{\text{dilated}}(\text{ReLU}(\text{Conv}_{\text{dilated}}(x)))). \tag{14}$$

These encoding alternatives enable the framework to adaptively select optimal feature transformation strategies based on the specific characteristics of different time series datasets.

Alternative Normalization Operations

Normalization techniques play a crucial role in stabilizing training and improving model performance. The method explores three core normalization strategies that are particularly suited for time series forecasting tasks:

Layer Normalization: The standard normalization applied across the feature dimension:

$$\text{LN}(x) = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta \tag{15}$$

Batch Normalization: Normalizes across the batch dimension, adapted for sequential data:

$$\text{BN}(x) = \gamma \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{16}$$

where μ_B and σ_B^2 are computed across the batch and sequence dimensions.

Root Mean Square Normalization (RMSNorm): A simplified normalization that eliminates the mean centering:

$$\text{RMSNorm}(x) = \gamma \frac{x}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \tag{17}$$

Feed-Forward Network with Searchable Activations

The feed-forward network in STrans maintains the standard two-layer structure (Vaswani et al. 2017) while incorporating searchable activation functions to optimize non-linear transformations:

$$\text{FFN}_{intermediate} = \text{MixedActivation}(\text{Conv1D}(x)), \tag{18}$$

$$\text{FFN}_{output} = \text{Conv1D}(\text{Dropout}(\text{FFN}_{intermediate})), \tag{19}$$

$$\text{FFN}(x) = \text{Dropout}(\text{FFN}_{output}). \tag{20}$$

The searchable activation functions include:

ReLU (Nair and Hinton 2010): $f(x) = \max(0, x)$ - provides computational simplicity and effective gradient flow.

GELU (Hendrycks and Gimpel 2016): $f(x) = x \cdot \Phi(x)$ - offers smooth non-linearity with probabilistic interpretation.

Swish (Ramachandran, Zoph, and Le 2017): $f(x) = x \cdot \sigma(x)$ - self-gated activation combining linear and sigmoid components.

ELU (Clevert, Unterthiner, and Hochreiter 2016): $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$ - provides smooth negative values for faster convergence.

Leaky ReLU (Maas, Hannun, and Ng 2013): $f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$ - prevents dead neurons while maintaining computational efficiency.

The framework automatically discovers optimal activation functions for each layer through the architecture search process, enabling adaptive non-linear transformations tailored to specific time series characteristics.

Search method: Differentiable Architecture Search

The method builds upon the DARTS framework, extending it to handle the complex multi-component search space designed for time series Transformers. The search process involves continuous relaxation of the discrete architecture choice problem.

Architecture Parameterization: For each layer $\ell \in \{1, 2, \ldots, L\}$, the framework maintains separate architecture parameters for each component type:

$$\boldsymbol{\alpha}^{(\ell)} = \{\alpha_{\text{attn}}^{(\ell)}, \alpha_{\text{norm1}}^{(\ell)}, \alpha_{\text{norm2}}^{(\ell)}, \alpha_{\text{enc1}}^{(\ell)}, \alpha_{\text{enc2}}^{(\ell)}, \alpha_{\text{act}}^{(\ell)}\} \tag{21}$$

where $\mathcal{O}_{\text{attn}}$, $\mathcal{O}_{\text{norm}}$, $\mathcal{O}_{\text{enc}}$, and $\mathcal{O}_{\text{act}}$ represent the operation sets for attention, normalization, encoding, and activation components, respectively. Note that this work maintains separate normalization and encoding parameters for the dual residual pathways in each layer.

Mixed Operation Computation: During the search phase, each component computes a weighted combination of all candidate operations. For the attention component in layer ℓ:

$$\text{MixedAttention}^{(\ell)}(\mathbf{x}) = \sum_{o \in \mathcal{O}_{\text{attn}}} \frac{\exp(\alpha_{\text{attn},o}^{(\ell)})}{\sum_{o' \in \mathcal{O}_{\text{attn}}} \exp(\alpha_{\text{attn},o'}^{(\ell)})} \cdot o(\mathbf{x}) \tag{22}$$

Similar formulations apply to the dual normalization operations (norm1, norm2), dual encoding operations (enc1, enc2), and activation components within the fixed feed-forward structure.

Bilevel Optimization Strategy: Following the DARTS paradigm, architecture search is formulated as a bilevel optimization problem:

$$\min_{\boldsymbol{\alpha}} \quad \mathcal{L}_{\text{val}}(\mathbf{w}^*(\boldsymbol{\alpha}), \boldsymbol{\alpha}) \tag{23}$$

$$\text{s.t.} \quad \mathbf{w}^*(\boldsymbol{\alpha}) = \arg\min_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}, \boldsymbol{\alpha}) \tag{24}$$

where $\mathbf{w}$ represents the network weights, $\boldsymbol{\alpha}$ denotes the architecture parameters, and $\mathcal{L}_{\text{train}}$, $\mathcal{L}_{\text{val}}$ are the training and validation losses, respectively.

Architecture Derivation: After convergence, STrans derives the final architecture by selecting the operation with the highest weight for each component:

$$o_{\text{attn}}^{(\ell)} = \arg\max_{o \in \mathcal{O}_{\text{attn}}} \alpha_{\text{attn},o}^{(\ell)} \tag{25}$$

To enhance robustness, avoid local optima, and prevent overfitting, the search stage generates multiple candidate architectures by considering the top-k operations for each component and enumerates their combinations, ultimately selecting architectures based on their combined validation performance scores.

Search Space Regularization: To further mitigate search overfitting in time series scenarios with limited data, STrans implements several regularization strategies:

- **Early Stopping**: Monitor validation performance during search and halt when improvement stagnates
- **Architecture Parameter Regularization (Krogh and Hertz 1991; Loshchilov and Hutter 2019)**: Apply L2 regularization to architecture parameters to prevent overconfident selections
- **Operation Diversity Encouragement (Chen et al. 2019)**: Include entropy-based terms to maintain diversity in operation selection during early search phases

Experiments

In this section, extensive experiments are conducted to validate the STrans method.

Datasets and Setup

This work evaluates STrans on benchmark datasets spanning multiple domains: ETT (ETTh1, ETTh2, ETTm1, ETTm2) for electricity consumption, Exchange for currency rates, Weather for meteorological data, ECL for electricity load, Solar-Energy for power production, and PEMS (PEMS03-08) for transportation networks. See the datasets section in the appendix.

Following standard protocols, the experiments use input sequence length $L_{\text{in}} = 96$ with prediction horizons $L_{\text{out}} \in \{96, 192, 336, 720\}$ for most datasets and $L_{\text{out}} \in \{12, 24, 48, 96\}$ for PEMS datasets. Chronological train-validation-test splits are adopted to prevent data leakage.

Evaluation Metrics

This work adopts two standard time series forecasting metrics:

Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{26}$$

Mean Absolute Error (MAE):

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{27}$$

where y_i and $\hat{y}_i$ represent ground truth and predicted values respectively, and N is the total number of predictions. MSE emphasizes larger errors while MAE provides robustness against outliers.

Compared Methods

STrans is evaluated against state-of-the-art time series forecasting models under identical experimental conditions.

State-of-the-Art Models: This work compares STrans against competitive recent models: **iTransformer** (Liu et al. 2023), achieving breakthrough performance through inverted attention; **PatchTST** (Nie et al. 2023), employing patch-based tokenization; **Crossformer** (Zhang and Yan 2023), utilizing hierarchical attention; **FEDformer** (Zhou et al. 2022), leveraging frequency domain transformations; **Autoformer** (Wu et al. 2021), incorporating decomposition mechanisms; **Stationary** (Liu et al. 2022), addressing non-stationarity; **DLinear** (Zeng et al. 2023), decomposing series with linear mappings; **RLinear** (Li et al. 2023), incorporating reversible normalization; **TimesNet** (Wu et al. 2022), converting 1D series into 2D space; **TiDE** (Das et al. 2023), employing encoder-decoder architecture; and **SCINet** (Liu et al. 2021), utilizing convolution and interaction networks.

Implementation Details: All baselines use official implementations with recommended hyperparameters. Experiments are conducted on NVIDIA A800 GPUs with identical preprocessing, data splits, and training procedures. Early stopping is applied based on validation performance.

Experimental Settings

The experimental framework consists of two distinct phases: architecture search and final model training, each with carefully tuned configurations to ensure robust and reproducible results.

Search Phase Configuration

Architecture Parameter Initialization: Architecture parameters $\boldsymbol{\alpha}$ are initialized using small random perturbations to ensure balanced exploration across all candidate operations:

$$\alpha_{c,o}^{(l)} \sim \mathcal{N}(0, 0.01^2) \tag{28}$$

where l denotes layer index, c represents component type (attention, normalization, encoding, activation), and o indicates operation index.

Network Weight Initialization: Model weights follow standard initialization schemes: embedding layers use Xavier uniform initialization, linear projections employ Kaiming normal initialization, and normalization parameters start with unit weights and zero biases.

Search Hyperparameters: Architecture search employs the following configuration: architecture learning rate $\alpha_{lr} = 1 \times 10^{-4}$ with Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999$); network learning rate $w_{lr} = 1 \times 10^{-4}$; architecture weight decay $\lambda_\alpha = 1 \times 10^{-3}$; network weight decay $\lambda_w = 3 \times 10^{-4}$; batch size adapted to dataset size and memory constraints; gradient clipping at 5.0; gradient accumulation when necessary; and cosine annealing scheduler with minimum learning rate $\eta_{min} = 1 \times 10^{-5}$. The search phase runs for 50 epochs with early stopping based on validation loss stagnation (patience = 10).

Final Selection: All the model genotypes are ranked by their total score received, with the top 8 genotypes selected for final training to ensure diversity while maintaining high performance potential.

Training Phase Configuration

Training Hyperparameters: Final model training uses refined hyperparameters: learning rate $\eta = 1 \times 10^{-4}$ with AdamW optimizer; batch size adapted per dataset complexity; training for 50 epochs with early stopping (patience = 10); weight decay of 3×10^{-4}; dropout rate of 0.1; gradient clipping at 5.0; and cosine annealing learning rate schedule.

Model Configuration: STrans employs dataset-adaptive architectural configurations with model dimensions and encoder layers selected based on input complexity and computational requirements; number of attention heads $n_{\text{heads}} = 8$; sequence length $L_{\text{in}} = 96$; and TimeF temporal feature embedding.

Main Results

Consistent Superior Performance: Tables 4 and 5 (Appendix Experimental Results) show STrans achieves consistent improvements across datasets, demonstrating automated architecture search effectiveness. Notable gains include 3.0% on high-dimensional Solar-Energy (137 features), 5.6% on financial Exchange data, and 8.5% on

Models		STrans (Ours)		iTransformer (2024)		RLinear (2023)		PatchTST (2023)		Crossformer (2023)		TiDE (2023)		TimesNet (2023)		DLinear (2022a)		SCINet (2022)		FEDformer (2022)		Stationary (2022b)		Autoformer (2021)	
Metric	Length	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	**0.322**	**0.364**	0.334	0.368	0.355	0.376	0.329	0.367	0.404	0.426	0.364	0.387	0.338	0.375	0.345	0.372	0.418	0.438	0.379	0.419	0.386	0.398	0.505	0.475
	192	0.373	0.394	0.377	0.391	0.391	0.392	**0.367**	**0.385**	0.450	0.451	0.398	0.404	0.374	0.387	0.380	0.389	0.439	0.450	0.426	0.441	0.459	0.444	0.553	0.496
	336	0.409	0.417	0.426	0.420	0.424	0.415	**0.399**	**0.410**	0.532	0.515	0.428	0.425	0.410	0.411	0.413	0.413	0.490	0.485	0.445	0.459	0.495	0.464	0.621	0.537
	720	0.480	0.455	0.491	0.459	0.487	0.450	**0.454**	**0.439**	0.666	0.589	0.487	0.461	0.478	0.450	0.474	0.453	0.595	0.550	0.543	0.490	0.585	0.516	0.671	0.561
ETTm2	96	0.179	0.264	0.180	0.264	0.182	0.265	**0.175**	**0.259**	0.287	0.366	0.207	0.305	0.187	0.267	0.193	0.292	0.286	0.377	0.203	0.287	0.192	0.274	0.255	0.339
	192	0.247	0.308	0.250	0.309	0.246	0.304	**0.241**	**0.302**	0.414	0.492	0.290	0.364	0.249	0.309	0.284	0.362	0.399	0.445	0.269	0.328	0.280	0.339	0.281	0.340
	336	0.312	0.349	0.311	0.348	0.307	**0.342**	**0.305**	0.343	0.597	0.542	0.377	0.422	0.321	0.351	0.369	0.427	0.637	0.591	0.325	0.366	0.334	0.361	0.339	0.372
	720	0.408	0.403	0.412	0.407	0.407	**0.398**	**0.402**	0.400	1.730	1.042	0.558	0.524	0.408	0.403	0.554	0.522	0.960	0.735	0.421	0.415	0.417	0.413	0.433	0.432
ETTh1	96	0.381	0.400	0.386	0.405	0.386	**0.395**	0.414	0.419	0.423	0.448	0.479	0.464	0.384	0.402	0.386	0.400	0.654	0.599	**0.376**	0.419	0.513	0.491	0.449	0.459
	192	0.431	0.431	0.441	0.436	0.437	**0.424**	0.460	0.445	0.471	0.474	0.525	0.492	0.436	0.429	0.437	0.432	0.719	0.631	**0.420**	0.448	0.534	0.504	0.500	0.482
	336	0.471	0.451	0.487	0.458	0.479	**0.446**	0.501	0.466	0.570	0.546	0.565	0.515	0.491	0.469	0.481	0.459	0.778	0.659	**0.459**	0.465	0.588	0.535	0.521	0.496
	720	0.484	0.483	0.503	0.491	**0.481**	**0.470**	0.500	0.488	0.653	0.621	0.594	0.558	0.521	0.500	0.519	0.516	0.836	0.699	0.506	0.507	0.643	0.616	0.514	0.512
ETTh2	96	**0.275**	**0.330**	0.297	0.349	0.288	0.338	0.302	0.348	0.745	0.584	0.400	0.440	0.340	0.374	0.333	0.387	0.707	0.621	0.358	0.397	0.476	0.458	0.346	0.388
	192	0.382	0.397	0.380	0.400	**0.374**	**0.390**	0.388	0.400	0.877	0.656	0.528	0.509	0.402	0.414	0.477	0.476	0.860	0.689	0.429	0.439	0.512	0.493	0.456	0.452
	336	0.427	0.434	0.428	0.432	**0.415**	**0.426**	0.426	0.433	1.043	0.731	0.643	0.571	0.452	0.452	0.594	0.541	1.000	0.744	0.496	0.487	0.552	0.551	0.482	0.486
	720	0.427	0.444	0.427	0.445	**0.420**	**0.440**	0.431	0.446	1.104	0.763	0.874	0.679	0.462	0.468	0.831	0.657	1.249	0.838	0.463	0.474	0.562	0.560	0.515	0.511
ECL	96	**0.145**	**0.239**	0.148	0.240	0.181	0.270	0.219	0.314	0.237	0.329	0.168	0.272	0.197	0.282	0.247	0.345	0.193	0.308	0.169	0.273	0.201	0.317	0.201	0.317
	192	**0.161**	**0.250**	0.162	0.253	0.188	0.274	0.231	0.322	0.236	0.330	0.184	0.289	0.196	0.285	0.257	0.355	0.201	0.315	0.182	0.286	0.222	0.334	0.222	0.334
	336	**0.177**	**0.267**	0.178	0.269	0.204	0.293	0.246	0.337	0.249	0.344	0.198	0.300	0.209	0.301	0.269	0.369	0.214	0.329	0.200	0.304	0.231	0.338	0.231	0.338
	720	**0.218**	**0.316**	0.225	0.317	0.246	0.324	0.280	0.363	0.284	0.373	0.220	0.320	0.245	0.333	0.299	0.390	0.246	0.355	0.222	0.321	0.254	0.361	0.254	0.361
Exchange	96	**0.082**	**0.203**	0.086	0.206	0.088	0.205	0.256	0.367	0.094	0.218	0.107	0.234	0.088	0.218	0.267	0.396	0.148	0.278	0.111	0.237	0.197	0.323	0.197	0.323
	192	**0.175**	**0.299**	0.177	**0.299**	0.176	**0.299**	0.470	0.509	0.184	0.307	0.226	0.344	0.176	0.315	0.351	0.459	0.271	0.315	0.219	0.335	0.300	0.369	0.300	0.369
	336	0.315	0.409	0.331	0.417	**0.301**	**0.397**	1.268	0.883	0.349	0.431	0.367	0.448	0.313	0.427	1.324	0.853	0.460	0.427	0.421	0.476	0.509	0.524	0.509	0.524
	720	**0.827**	**0.690**	0.847	0.691	0.901	0.714	1.767	1.068	0.852	0.698	0.964	0.746	0.839	0.695	1.058	0.797	1.195	0.695	1.092	0.769	1.447	0.941	1.447	0.941
Weather	96	0.162	**0.209**	0.174	0.214	0.177	0.218	**0.158**	0.230	0.202	0.261	0.172	0.220	0.196	0.255	0.221	0.306	0.217	0.296	0.173	0.223	0.266	0.336	0.266	0.336
	192	0.213	0.256	0.221	**0.254**	0.225	0.259	**0.206**	0.277	0.242	0.298	0.219	0.261	0.237	0.296	0.261	0.340	0.276	0.336	0.245	0.285	0.307	0.367	0.307	0.367
	336	**0.270**	0.297	0.278	**0.296**	0.278	0.297	0.272	0.335	0.287	0.335	0.280	0.306	0.283	0.335	0.309	0.378	0.339	0.380	0.321	0.338	0.359	0.395	0.359	0.395
	720	0.354	0.349	0.358	**0.347**	0.354	0.348	0.398	0.418	0.351	0.386	0.365	0.359	**0.345**	0.381	0.377	0.427	0.403	0.428	0.414	0.410	0.419	0.428	0.419	0.428
Solar-Energy	96	**0.198**	**0.235**	0.203	0.237	0.234	0.286	0.310	0.331	0.312	0.399	0.250	0.292	0.290	0.378	0.237	0.344	0.242	0.342	0.215	0.249	0.884	0.711	0.884	0.711
	192	**0.229**	**0.260**	0.233	0.261	0.267	0.310	0.734	0.725	0.339	0.416	0.296	0.318	0.320	0.398	0.280	0.380	0.285	0.380	0.254	0.272	0.834	0.692	0.834	0.692
	336	**0.244**	**0.271**	0.248	0.273	0.290	0.315	0.750	0.735	0.368	0.430	0.319	0.330	0.353	0.415	0.304	0.389	0.282	0.376	0.290	0.296	0.941	0.723	0.941	0.723
	720	**0.247**	**0.274**	0.249	0.275	0.289	0.317	0.769	0.765	0.370	0.425	0.338	0.337	0.356	0.413	0.308	0.388	0.357	0.427	0.285	0.295	0.882	0.717	0.882	0.717

Table 1: Multivariate forecasting results across prediction lengths (96, 192, 336, 720) with input length 96. Best in **purple**, second in **pink**.

Weather forecasting versus iTransformer, validating that automated search discovers superior component combinations over manual designs.

Ablation studies and analysis

Searched models on the benchmark datasets: I present the optimal searched models in Figure 2. We can find that: (a) The top-performing models on different tasks are different. For example, ETTh1 task utilizes Full, Bilinear, and Minus Attention with hybrid convolutional and MLP components. In comparison, The Exchange task uses the Cocat and Bilinear Attention mechanisms.

Efficiency analysis: Table 2 shows time and GPU costs for the Exchange task. Computational costs are tracked for each pipeline stage with sequence length 96, prediction length 96, and 8 input features. The search uses 50 epochs with batch size 16 for architecture search and batch size 8 for validation, achieving MSE of 0.084.

Model performance under different prediction lengths This work systematically evaluates STrans performance across varying prediction horizons to understand how forecasting accuracy degrades with extended prediction lengths. Using ETTh1 and Exchange Rate datasets with fixed input sequence length of 96, this study tests prediction lengths of 96, 192, 336, and 720 time steps to identify optimal forecasting windows for different temporal domains. The experimental results are presented in Figure 3 and 4.

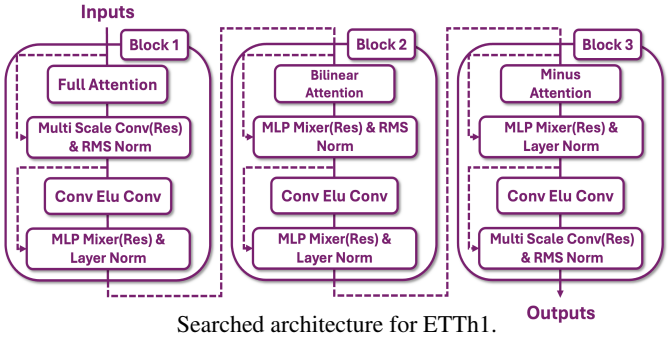
Stage	Duration	GPU Days
Architecture Search	59 min 10 sec	0.041
Model Validation	100 min 57 sec	0.070
Total Pipeline	**160 min 7 sec**	**0.111**

Table 2: Efficiency Analysis.

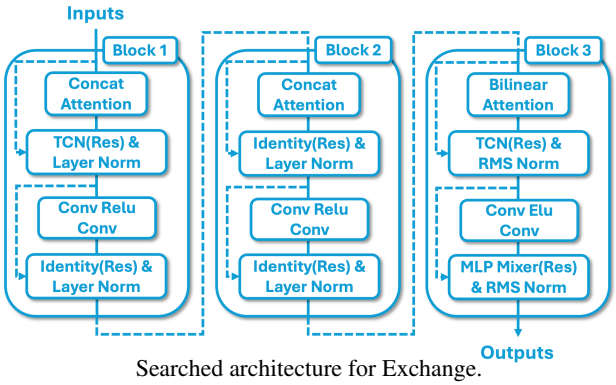
The key findings for this group of experiments are as follows:

Contrasting Degradation Patterns: ETTh1 exhibits stable linear degradation with only 27.0% MSE increase from 96 to 720 steps, reflecting predictable electrical systems with seasonal patterns. Exchange Rate shows catastrophic exponential deterioration with 908.5% MSE increase, revealing inherent financial market volatility and non-stationarity that makes long-term forecasting extremely challenging.

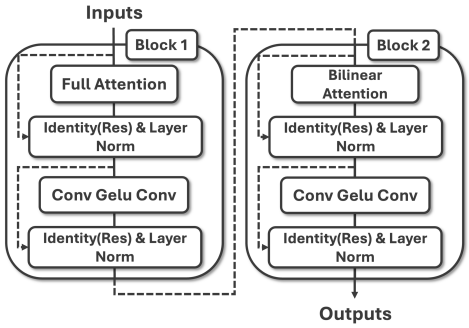
Architecture Robustness: With the same discovered ar-



Searched architecture for ETTh1.



Searched architecture for Exchange.



Searched architecture for Solar.

Figure 2: Searched STrans architectures on ETTh1, Exchange, and Solar datasets. Each design reflects dataset-specific temporal modeling strategies.

chitecture, STrans adapts well across domains. The shared attention mechanisms and encoders capture both smooth electrical trends and volatile financial patterns, confirming the generalizability of its components.

Performance under different model dimensions: This work examines how model dimension (128, 256, 512) affects STrans performance on ETTh1 and Exchange Rate, providing insight into suitable model capacity for time-series forecasting.

Dimension 256 offers the best performance–efficiency trade-off, improving MSE over dimension 128 by 1.16 % on ETTh1 and 3.18 % on Exchange Rate. Dimension 512 provides no additional benefit—ETTh1 slightly degrades and Exchange Rate remains unchanged—while incurring 4× higher computational cost, making 256 the preferred choice in practice.

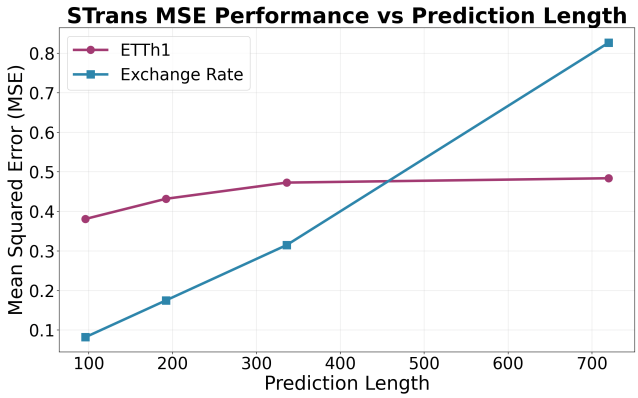


Figure 3: STrans MSE performance across prediction lengths for ETTh1 and Exchange Rate.

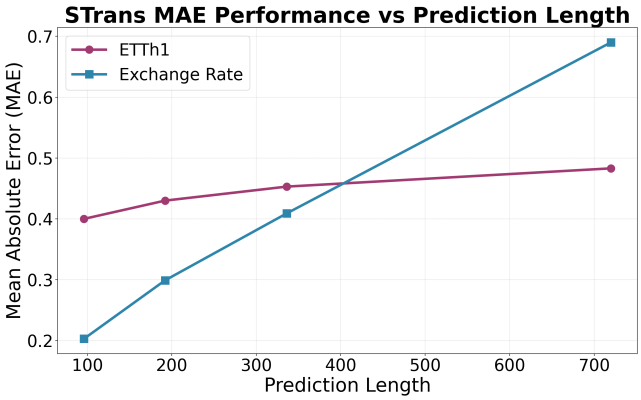


Figure 4: STrans MAE performance across prediction lengths for ETTh1 and Exchange Rate.

Dataset	Dim	MSE	MAE	Improv.	Params
ETTh1	128	0.385	0.404	-	1.2M
	256	**0.380**	**0.400**	1.16%	4.5M
	512	0.381	0.402	1.10%	17.8M
Exchange	128	0.085	0.205	-	1.1M
	256	**0.082**	**0.202**	3.18%	4.2M
	512	**0.082**	0.203	3.18%	16.5M

Table 3: Model dimension analysis on ETTh1 and Exchange Rate datasets. Evaluation of STrans with model dimensions $d_{model} \in \{128, 256, 512\}$ using sequence length 96, prediction length 96.

Conclusion

STrans introduces a comprehensive neural architecture search framework for time series Transformers, automatically discovering optimal combinations of attention mechanisms, normalization techniques, encoding operations, and activation functions. Extensive experiments demonstrate competitive performance across diverse temporal domains, revealing that automated search can uncover domain-specific design principles and establish new directions for adaptive forecasting systems.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Chen, X.; Xie, L.; Wu, J.; and Tian, Q. 2019. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1294–1303.

Chitty-Venkata, K. T.; Emani, M.; Vishwanath, V.; and Somani, A. K. 2022. Neural architecture search for transformers: A survey. *IEEE Access*, 10: 108374–108412.

Clevert, D.-A.; Unterthiner, T.; and Hochreiter, S. 2016. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations*.

Das, A.; Kong, W.; Leach, A.; Mathur, S.; Sen, R.; and Yu, R. 2023. Long-term forecasting with tide: Time-series dense encoder. *arXiv preprint arXiv:2304.08424*.

Dong, X.; and Yang, Y. 2019. Searching for a robust neural architecture in four GPU hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1761–1770.

Gehring, J.; Auli, M.; Grangier, D.; Yarats, D.; and Dauphin, Y. N. 2017. Convolutional sequence to sequence learning. In *International conference on machine learning*, 1243–1252. PMLR.

Hag, I. U.; Lee, B. S.; and Rizzo, D. M. 2025. TransNAS-TSAD: harnessing transformers for multi-objective neural architecture search in time series anomaly detection. *Neural Computing and Applications*, 37(4): 2455–2477.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Krogh, A.; and Hertz, J. A. 1991. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4.

Li, Z.; Rao, Z.; Pan, L.; Wang, P.; and Xu, Z. 2023. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*.

Liu, H.; Simonyan, K.; and Yang, Y. 2019. DARTS: Differentiable Architecture Search. In *International Conference on Learning Representations*.

Liu, M.; Zeng, A.; Chen, M.; Xu, Z.; Lai, Q.; Ma, L.; and Xu, Q. 2021. SCINet: Time series modeling and forecasting with sample convolution and interaction. *Advances in Neural Information Processing Systems*, 34: 5816–5828.

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in Neural Information Processing Systems*, 35: 9881–9893.

Liu, Y.; Wu, H.; Wang, J.; and Long, M. 2023. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. *arXiv preprint arXiv:2310.06625*.

Loshchilov, I.; and Hutter, F. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421.

Maas, A. L.; Hannun, A. Y.; and Ng, A. Y. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, 3.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814.

Nie, Y.; Nguyen, N. H.; Sinthong, P.; and Kalagnanam, J. 2023. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. In *International Conference on Machine Learning*, 4095–4104.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Searching for activation functions. In *arXiv preprint arXiv:1710.05941*.

Shazeer, N. 2020. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202*.

Tolstikhin, I. O.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; Zhai, X.; Unterthiner, T.; Yung, J.; Steiner, A.; Keysers, D.; Uszkoreit, J.; et al. 2021. MLP-mixer: An all-MLP architecture for vision. In *Advances in neural information processing systems*, volume 34, 24261–24272.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wen, Q.; Zhou, T.; Zhang, C.; Chen, W.; Ma, Z.; Yan, J.; and Sun, L. 2022. Transformers in time series: A survey. *arXiv preprint arXiv:2202.07125*.

Wu, H.; Hu, T.; Liu, Y.; Zhou, H.; Wang, J.; and Long, M. 2022. TimesNet: Temporal 2D-variation modeling for general time series analysis. *arXiv preprint arXiv:2210.02186*.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34: 22419–22430.

Xu, Y.; Xie, L.; Zhang, X.; Chen, X.; Qi, G.-J.; Tian, Q.; and Xiong, H. 2020. PC-DARTS: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*.

Yu, F.; and Koltun, V. 2016. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 11121–11128.

Zhang, Y.; and Yan, J. 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11106–11115.

Zhou, T.; Ma, Z.; Wen, Q.; Wang, X.; Sun, L.; and Jin, R. 2022. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, 27268–27286. PMLR.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2017. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8697–8710.