XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING

Grammatical Neuroevolution for Efficient Multivariate Time Series Forecasting

Jefferson O. Andrade[1], Heitor M. de Oliveira[2], Caio C. O. Coronel[2], Luiz F. E. dos Santos[2], Karin S. Komati[1]

[1]*Graduate Program in Applied Computing, Serra Campus, Ifes*
jefferson.andrade@ifes.edu.br, kkomati@ifes.edu.br

[2]*Dept. of Information Systems, Serra Campus, Ifes*
Av. dos Sabiás, 330, Serra - ES, 29166-630, Brazil
oliheitor189@gmail.com, caiocesarcoronel@gmail.com, luiz.elizeta.123@gmail.com

Abstract. This work introduces Grammatical Neuroevolution (GNE), a novel approach that applies Grammatical Evolution (GE) to automatically design neural network architectures for multivariate time series forecasting. Traditional methods often depend on manual design or fixed templates, which may not capture the complex temporal and cross-variable dependencies in energy demand data. GNE uses a context-free grammar to define the architecture search space, covering layer types, neuron counts, activation functions, hyperparameters, and connectivity patterns. The system evolves sequences of production rules to generate diverse and potentially optimal architectures without human intervention. We evaluate GNE on a synthetic multivariate time series dataset using standard metrics such as Root Mean Squared Error, Mean Absolute Error, and R^2. Results show that GNE achieves superior forecasting accuracy, discovering a specialized architecture that significantly outperforms a traditional statistical model. This demonstrated ability to generate high-performance, tailored models is particularly advantageous for complex forecasting tasks and for deployment in performance-critical applications. These findings underscore the potential of grammar-based approaches for enhancing predictive modeling in complex domains.

Keywords: Grammatical Neuroevolution, Neural Architecture Search, Energy Demand Prediction, Grammatical Evolution.

1 Introduction

Multivariate time series forecasting plays a vital role in engineering and scientific domains, from predicting power grid loads to modeling industrial processes [1]. Accurate forecasting—capturing the complex interplay among multiple variables—is essential for effective planning, resource allocation, and control. Artificial Neural Networks (ANNs) offer the ability to model nonlinear patterns and interactions [2], but their performance heavily depends on architecture design, which remains a challenging task [3].

One of the main difficulties in time series forecasting is the vast hyperparameter search space, influenced by varying data scales, sampling frequencies, and sequence lengths. These factors demand model configurations adapted to specific tasks, making manual tuning increasingly impractical. Neuroevolution addresses this issue by optimizing both hyperparameters and network topology through evolutionary principles such as population-based search, crossover, and mutation [4]. This enables parallel exploration of the architecture space and reduces the risk of convergence to local optima. The effectiveness of neuroevolution relies on how architectures are represented and evolved. To this end, we propose a Grammatical Neuroevolution (GNE) framework that automatically designs neural networks for multivariate time series forecasting. It employs Grammatical Evolution (GE), which uses formal grammars to guide the search and ensure that generated architectures are syntactically valid and structurally coherent [5]. This automated strategy enables scalable, optimized model generation for complex forecasting tasks.

Our contributions are twofold. First, we introduce the Structured Artificial Neural Network Definition Language (SANDL), a declarative and hierarchical language for representing ANN architectures. SANDL offers both expressiveness and constraint, making it suitable for programmatic manipulation by GE. Second, we present a GNE system that leverages SANDL to automate the generation and optimization of architectures, replacing manual design with an evolutionary discovery process capable of producing efficient models tailored to the dynamics of time series data.

The structure of the article is as follows: Section 2 presents related works. Section 3 describes the SANDL and GNE process. Section 4 presents the experimental setup, Section 5 reports and discusses the results obtained. Finally, Section 5 concludes the article and outlines directions for future research.

2 Related Works

The paper of Mahdinejad and Murphy [6] explores the generalizability of U-Net models evolved through grammatical neuroevolution, which optimizes both architecture and hyperparameters using evolutionary algorithms. Building on prior work that applied this approach to iris blood vessel segmentation on the DRIVE dataset, the study evaluates whether the best evolved models can transfer to other tasks. These models were tested on another segmentation benchmark (CHASEDB1) and a different domain—flood-affected area detection. Results show that the evolved models perform well across datasets, often being significantly smaller than baseline U-Nets while maintaining competitive accuracy. These findings highlight the potential of grammatical neuroevolution for transfer learning across domains.

The paper of Winter and Teahan [7] introduces Neuvo GEAF, a novel approach that uses grammatical evolution (GE) to automatically evolve activation functions tailored to specific neural network architectures and datasets. While standard functions like ReLU are widely adopted, they may not yield optimal results for all tasks. Experiments on multiple binary classification datasets show statistically significant F1-score improvements (2.4% to 9.4%) over ReLU, without increasing model size. These results support the development of efficient networks for resource-constrained edge devices and suggest that evolving task-specific activation functions can substantially enhance classification performance.

Our work is distinguished from Winter and Teahan's primarily by the scope of the evolutionary process. Whereas their approach employs Grammatical Evolution to optimize a specific network component — the activation function — for a given architecture, our methodology using SANDL and NEG is focused on evolving the macro-structure of the network itself. Consequently, our contribution lies in the automated design of the entire neural topology for time series, as opposed to the optimization of a functional element for classification tasks.

3 Methodology

GE operates through a mapping process that translates a simple genotype, a vector of integers called codons, into a complete neural network architecture (the phenotype). This mapping relies on a formal grammar, typically defined in Backus-Naur Form (BNF), which specifies all valid model structures. The grammar contains non-terminal symbols (abstract placeholders like ⟨*hidden_layer*⟩) and terminal symbols (concrete elements like '`LSTM`', 128, or '`Adam`'). The derivation begins at a start symbol (e.g., ⟨*ann*⟩) and proceeds left to right. For each non-terminal encountered, a codon is consumed to select a production rule using the modulo operation. This continues until a fully defined architecture is produced. If the genotype is exhausted before completion, a wrapping mechanism restarts the codon sequence.

To represent and manipulate neural network architectures, we developed SANDL (Structured Artificial Neural Network Definition Language) — a declarative and human-readable format that acts as the phenotypic representation of these architectures. SANDL is optimized for the programmatic generation and modification of neural networks aimed at time series forecasting. The structure of any valid SANDL program is formally defined by a BNF grammar, which serves as the foundation for the GE process. This ensures that any architecture generated is syntactically valid. An excerpt from this grammar illustrates its component-based design:

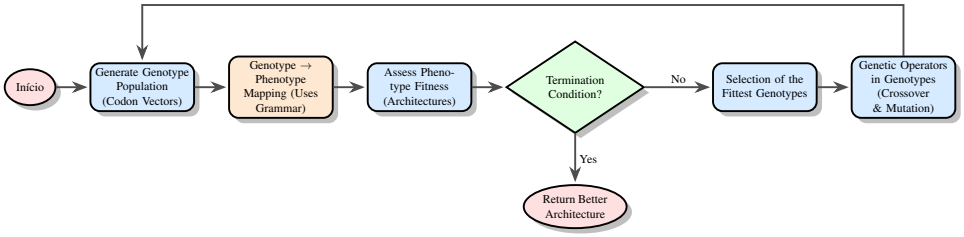
```
<hidden_layer>  ::= <layer_type> '{' ( <layer_param> ';' )* '}'
<layer_type>    ::= 'dense' | 'lstm' | 'gru' | 'conv1d' | 'tcn' | ...
<layer_param>   ::= <activation> | <dropout> | <l1> | <l2> | ...
<activation>    ::= 'activation' '=' <activation_fn>
<activation_fn> ::= 'linear' | 'relu' | 'sigmoid' | 'tanh' | ...
<dropout>       ::= 'dropout' '=' FLOAT
```

Figure 1. Small excerpt of SANDL's grammar.

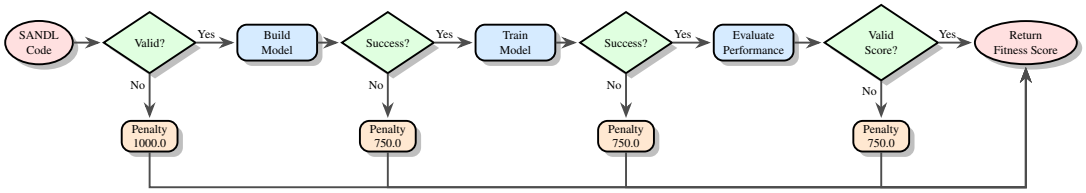
This grammar allows the evolutionary process to construct architectures by combining different layers and parameters. SANDL is a declarative and portable language that focuses on what the architecture should be rather than how to build it, enhancing readability, portability, and integration. It adopts a hierarchical structure, using nested blocks ({}) to group parameters by layer or section, mirroring the modular design of neural networks. Each block contains explicit `key = value` pairs for direct hyperparameter configuration, enabling precise and transparent definition of both network-wide and layer-specific settings.

3.1 The Grammatical Neuroevolution Process

The GNE process integrates the principles of GE with the SANDL specification to create a fully automated workflow for designing neural network architectures. This entire framework is illustrated in Figure 2, which details both the high-level evolutionary loop and the fine-grained fitness evaluation procedure.



(a) Main evolutionary loop where genotypes are mapped to neural network phenotypes and evaluated for fitness.



(b) The fitness evaluation process for assessing neural architecture quality.

Figure 2. GENNAS framework architecture.

The main evolutionary cycle, illustrated in Figure 2a, starts by initializing a population of genotypes, each represented as a vector of integers (codons). For each individual, the GE mapping process translates the genotype into a phenotype—an ANN architecture defined in SANDL—using the grammar's production rules. The resulting architecture undergoes a fitness evaluation to assess its quality. The algorithm then checks whether the termination condition (e.g., maximum generations) has been reached. If not, the fittest individuals are selected, and genetic operators such as crossover and mutation are applied to generate a new population. This process repeats iteratively, guiding the search toward better architectures.

The fitness evaluation, illustrated in Figure 2b, is a multi-stage process designed to ensure that architectures are syntactically valid, functionally viable, and performant. The evaluation begins with validation, where the SANDL code is parsed; if it is syntactically invalid, a severe penalty is applied and the evaluation terminates. If the code is valid, the next step attempts to construct the corresponding neural network. Failures during model construction, such as incompatible layer connections, also result in a high penalty. Successfully built models proceed to the training phase, where numerical instabilities or training failures trigger additional penalties. Finally, the model's predictive performance is assessed on a validation set. If this evaluation yields an invalid score, such as NaN, another penalty is assigned. Otherwise, standard metrics including RMSE, MAE, and R^2 are returned as the fitness score.

4 Case Study

The experiment was conducted using a synthetic time series dataset with 500 daily observations spanning from January 1, 2020, to May 14, 2021. It includes three input features (`feat_1`, `feat_2`, `feat_3`) and one continuous target variable, simulating a forecasting scenario common in real-world applications. Data preprocessing followed standard time series practices. For the neural models, features were standardized using z-score normalization to ensure stable training. The data was segmented into sequences of 20 timesteps with a 1-step prediction horizon, generating input-output pairs for supervised learning. A sequential split allocated 80% of the data (400 samples) for training and 20% (100 samples) for testing, preserving temporal structure.

As a statistical baseline, we used the AutoRegressive Integrated Moving Average (ARIMA) model. The optimal model order was selected via grid search based on the Akaike Information Criterion (AIC) [8]. The Augmented Dickey-Fuller test indicated non-stationarity (p-value = 0.085), requiring first differencing. The best configuration was ARIMA(3,0,3), with AIC = -559.41 and BIC = -527.48, balancing model complexity and fit. Candidate neural architectures were evaluated using RMSE on validation data, with penalties for invalid models or failed training. Key configuration parameters included a population size of 50, a maximum of 100 generations, crossover probability of 0.8, mutation probability of 0.1, and a fitness evaluation timeout of 5 minutes. Early stopping was employed with a patience of 10 epochs.

5 Results

The GENNAS evolutionary process converged to an optimal neural network architecture after systematic exploration of the design space. The automatically discovered architecture, expressed in SANDL notation, is presented in Listing 1. The evolved architecture exhibits several characteristics that reflect effective design principles for time series modeling. The stacked LSTM configuration with decreasing unit counts (64 $\rightarrow$ 32) implements a hierarchical feature extraction strategy, where the first layer captures low-level temporal patterns and the second layer learns higher-level abstractions. The inclusion of dropout layers with varying rates (0.2, 0.1, 0.1) provides adaptive regularization to prevent overfitting while maintaining model expressiveness. The architecture comprises 30,369 trainable parameters, representing a substantial increase in model complexity compared to the 6-parameter ARIMA baseline. However, as demonstrated in the performance results, this increased complexity is justified by significant improvements in predictive accuracy and generalization capability.

Listing 1. GENNAS-generated neural architecture in SANDL format.

```
neuralnet {
    input { features = 3; sequence_length = 20; }
    lstm { units = 64; dropout = 0.2; return_sequences = true; }
    lstm { units = 32; dropout = 0.1; return_sequences = false; }
    dense { units = 16; activation = relu; }
    dropout { rate = 0.1; }
    output { units = 1; activation = linear; }
}
```

Table 1. Performance comparison between ARIMA and GENNAS-generated neural network

Model	**RMSE**	**MAE**	**R^2**	**Parameters**
ARIMA(3,0,3)	0.2813	0.2426	-0.1971	6
GENNAS Neural Net	**0.1157**	**0.0871**	**0.8045**	30,369
Improvement	**+58.85%**	**+64.09%**	**+508.07%**	–



Figure 3. Unlike the ARIMA model's poor tracking of actual values, the GENNAS neural network's predictions are more accurate, leading to improved performance metrics (RMSE, MAE, R2) and a better overall model fit.

Table 1 presents a comparison of model performance across all evaluation metrics. The results demonstrate the clear superiority of the GENNAS-generated neural architecture over the traditional statistical approach. The

neural network achieved remarkable performance improvements across all metrics. The RMSE decreased by 58.85% from 0.2813 to 0.1157, indicating more accurate predictions. Similarly, the MAE improved by 64.09%, demonstrating consistent reduction in prediction errors. Most notably, the coefficient of determination improved from -0.1971 to 0.8045, representing a transformation from poor predictive capability (worse than a simple mean predictor) to strong explanatory power.

The GENNAS-generated architecture converged efficiently, completing training in 16 epochs with early stopping. The final training loss was 0.0247, and the validation loss stabilized at 0.0133, indicating good generalization without overfitting. This rapid convergence suggests that the evolutionary process identified an architecture well-adapted to the data. Figure 3 compares both models using time series plots and performance visualizations. The neural network shows superior alignment with actual values, especially in capturing complex temporal patterns that the linear ARIMA model could not represent adequately.

6 Conclusion

This paper presented GENNAS, a framework for automated neural architecture search tailored to time series forecasting. By integrating GE with the SANDL domain-specific language, GENNAS automates the design of effective neural networks without requiring manual hyperparameter tuning or specialized expertise. Experimental results show that GENNAS significantly outperforms traditional statistical methods, achieving a 58.85% reduction in RMSE compared to the ARIMA baseline. Additionally, the use of SANDL provides interpretable and reproducible architecture specifications. Key contributions include: (1) a human-readable language (SANDL) for representing neural architectures; (2) a grammar-based search strategy that efficiently explores the architecture space while respecting domain constraints; and (3) a robust fitness evaluation framework to assess model performance under realistic conditions.

Future research directions include extending the framework to handle multivariate forecasting scenarios, incorporating uncertainty quantification mechanisms, and developing specialized grammars for specific application domains such as financial time series or sensor data analysis. Additionally, investigating hybrid approaches that combine evolutionary search with gradient-based optimization could further enhance the efficiency and effectiveness of the architecture discovery process.

Acknowledgements. Professor Komati thanks CNPq for grantDT-2 (nº 302726/2023-3) and for project nº 407742-/2022-0, and also FAPES nº 1023/2022 P:2022-8TZV6.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] E. Sivadasan, N. M. Sundaram, and R. Santhosh. Deep learning for energy forecasting using gated recurrent units and long short-term memory. *Journal of Intelligent Systems & Internet of Things*, vol. 14, n. 1, 2025.

[2] B. Lim and S. Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, vol. 379, n. 2194, pp. 20200209, 2021.

[3] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, vol. 20, n. 55, pp. 1–21, 2019.

[4] E. Galván and P. Mooney. Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Transactions on Artificial Intelligence*, vol. 2, n. 6, pp. 476–493, 2021.

[5] K. Soltanian, A. Ebnenasir, and M. Afsharchi. Modular grammatical evolution for the generation of artificial neural networks. *Evolutionary computation*, vol. 30, n. 2, pp. 291–327, 2022.

[6] M. Mahdinejad and A. Murphy. Generalisability of u-net models evolved using grammatical neuroevolution. *IET Conference Proceedings*, vol. 2024, pp. 307–310, 2024.

[7] B. D. Winter and W. J. Teahan. Task-specific activation functions for neuroevolution using grammatical evolution, 2025.

[8] A. Khalid and A. I. Sarwat. Unified univariate-neural network models for lithium-ion battery state-of-charge forecasting using minimized akaike information criterion algorithm. *Ieee Access*, vol. 9, pp. 39154–39170, 2021.