Evoluindo Arquiteturas para Previsão de Séries Temporais via Neuroevolução Gramatical

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

[1] Programa de Pós-graduação em Computação Aplicada
Instituto Federal do Espírito Santo (IFES) – Campus Serra
Av. dos Sabiás, 330 – Serra – ES – 29166-630 – Brazil

{jefferson.andrade,kkomati}@ifes.edu.br

[2]Instituto Federal do Espírito Santo (IFES) – Campus Serra
Av. dos Sabiás, 330 – Serra – ES – 29166-630 – Brazil

{heitor.oliveira,caio.coronel,luiz.elizeta}@estudante.ifes.edu.br

Abstract. *This work presents SANDL (Structured Artificial Neural Network Definition Language), a declarative and hierarchical language for describing artificial neural network architectures. Based on this language, we propose a Grammatical Neuroevolution (GNE) system to automatically design and optimize architectures for multivariate time series forecasting, replacing manual design with an automated discovery process. Evaluated on a synthetic multivariate time series dataset, the approach outperformed a traditional statistical model, producing efficient architectures adapted to complex dependencies.*

Resumo. *Este trabalho apresenta a SANDL (Structured Artificial Neural Network Definition Language), uma linguagem declarativa e hierárquica para descrever arquiteturas de redes neurais artificiais. A partir dessa linguagem, propõe-se um sistema de Grammatical Neuroevolution (GNE) para projetar e otimizar automaticamente arquiteturas voltadas à previsão de séries temporais multivariadas, substituindo o projeto manual por um processo de descoberta automatizado. Avaliada em um conjunto sintético de séries temporais multivariadas, a abordagem superou um modelo estatístico tradicional, gerando arquiteturas eficientes e adaptadas a dependências complexas.*

1. Introdução

A previsão de séries temporais multivariadas (STM) consiste na estimativa de valores futuros a partir de múltiplas variáveis observadas ao longo do tempo. Essa abordagem é empregada em diferentes áreas da engenharia e da ciência, incluindo a estimativa de carga em redes elétricas e a modelagem de processos industriais [Sivadasan et al. 2025]. Em STM, cada variável evolui ao longo do tempo e pode apresentar relações de dependência com as demais, sejam elas diretas ou indiretas. Essas interdependências podem ocorrer de forma simultânea, quando o valor de uma variável em um dado instante está correlacionado ao valor de outra no mesmo instante, ou defasada, quando o valor de uma variável influencia outra em instantes futuros [Wang et al. 2019].

Na literatura clássica, prevalecem métodos estatísticos para previsão de séries temporais, especialmente os autorregressivos vetoriais (VAR) [Box et al. 2015]. Entretanto, a dependência de pressupostos rígidos de linearidade e estacionariedade limita sua aplicabilidade a dinâmicas complexas do mundo real. As Redes Neurais Artificiais (ANNs) superam parte dessas restrições ao aprender padrões não lineares e interações complexas [Lim and Zohren 2021], mas seu desempenho está fortemente condicionado ao projeto da arquitetura, etapa que continua sendo um desafio não trivial [Elsken et al. 2019]. A escolha dos hiperparâmetros é um fator decisivo para o desempenho final de modelos baseados em neuroevolução [Galván and Mooney 2021]. Elementos como o número de neurônios em cada camada, o número total de épocas de treinamento, a taxa de aprendizado e o número de camadas compõem um espaço de busca extremamente amplo. Além disso, o efeito desses parâmetros não é trivial: pequenas alterações podem levar ao *overfitting*, *underfitting* ou a uma convergência mais lenta, impactando diretamente o modelo.

A neuroevolução aplica conceitos inspirados na evolução biológica — como populações de indivíduos (arquiteturas), seleção dos mais aptos e operadores de variação (cruzamento e mutação) — para explorar, de forma paralela e aleatória, diferentes topologias de redes [Stanley et al. 2019]. O desempenho dessa abordagem depende, sobretudo, de como as arquiteturas candidatas são representadas e manipuladas pelo algoritmo evolutivo. Nesse ponto, a Grammatical Evolution (GE) [Soltanian et al. 2022] se destaca como um método estruturado para definir o espaço de busca, sendo o foco deste trabalho.

Este trabalho propõe uma solução automatizada para gerar arquiteturas de redes neurais otimizadas para tarefas específicas de previsão de STM. Para isso, apresentamos a Structured Artificial Neural Network Definition Language (SANDL), uma linguagem declarativa e hierárquica para descrever arquiteturas de redes neurais artificiais. A SANDL foi projetada para permitir tanto flexibilidade quanto controle na definição de modelos, sendo adequada para manipulação automática por algoritmos de Evolução Gramatical (EG). Com base nessa linguagem, desenvolvemos um sistema de neuroevolução gramatical (NEG) que automatiza a criação e otimização de arquiteturas, produzindo modelos eficientes ajustados às características das séries temporais analisadas.

A estrutura do artigo é a seguinte: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 descreve a SANDL e o processo de GNE; a Seção 4 relata e discute os resultados obtidos; por fim, a Seção 5 conclui o artigo e indica direções para pesquisas futuras.

2. Related Works

O artigo de Soltanian et al. [Soltanian et al. 2022] apresenta uma proposta, denominada Modular Grammatical Evolution (MGE), com o objetivo de validar a hipótese de que restringir o espaço de soluções da Neuroevolução a redes neurais modulares e que permite a geração eficiente de redes menores e mais estruturadas. A representação do MGE é modular, de forma que cada indivíduo possui um conjunto de genes, e cada gene é mapeado para um neurônio por meio de regras gramaticais. São definidas e avaliadas cinco formas diferentes de estruturas, com e sem modularidade, utilizando o MGE, sendo constatado que módulos de camada única sem acoplamento são mais produtivos. O método proposto foi validado em dez *benchmarks* de classificação, com diferentes tamanhos, quantidades de atributos e classes de saída. Os resultados experimentais indicam que o MGE oferece acurácia superior em relação aos métodos existentes de neuroevolução e gera classifica-

dores mais simples do que aqueles produzidos por outras abordagens de aprendizado de máquina.

O artigo de Winter e Teahan [Winter and Teahan 2025] apresenta o Neuvo GEAF, uma abordagem inovadora que utiliza a evolução gramatical para evoluir automaticamente funções de ativação adaptadas a arquiteturas específicas de redes neurais e conjuntos de dados. Embora funções padrão como a ReLU sejam amplamente utilizadas, elas podem não oferecer resultados ideais para todas as tarefas. Experimentos realizados em múltiplos conjuntos de dados de classificação binária mostram melhorias estatisticamente significativas na medida F1 (de 2,4% a 9,4%) em relação à ReLU, sem aumento no tamanho do modelo. Esses resultados apoiam o desenvolvimento de redes eficientes para dispositivos de borda com restrições de recursos e sugerem que a evolução de funções de ativação específicas para cada tarefa pode melhorar substancialmente o desempenho em classificação.

3. Framework de Neuroevolução Gramatical (GENNAS)

Este trabalho apresenta um *framework* de NEG para o projeto automatizado de ANNs voltadas à previsão de séries temporais multivariadas, o GENNAS.

3.1. Evolução Gramatical (EG)

O mecanismo central da EG é um processo de mapeamento que traduz um genótipo simples em um fenótipo complexo e funcional. O processo se baseia em três componentes principais: uma gramática formal, o genótipo e o fenótipo.

A gramática, normalmente definida na forma Backus-Naur Form (BNF), estabelece o universo de todas as arquiteturas de redes neurais válidas que podem ser geradas. Ela é composta por símbolos não terminais — marcadores abstratos que representam escolhas a serem feitas (por exemplo, $\langle$*hidden_layer*$\rangle$) — e símbolos terminais, que são os blocos construtivos concretos e finais do modelo (por exemplo, '`LSTM`', 128, '`Adam`'). O genótipo é uma representação de baixo nível composta por um vetor de inteiros (codons), geralmente inicializado de forma aleatória. O fenótipo, por sua vez, é a expressão executável final do genótipo: uma arquitetura completa de rede neural.

O processo de mapeamento, ou derivação, começa a partir do símbolo inicial da gramática (por exemplo, $\langle$*ann*$\rangle$). O algoritmo percorre a expressão da esquerda para a direita e, ao encontrar o primeiro não terminal, consome o próximo códon do genótipo para selecionar uma regra de produção. A escolha é feita utilizando o operador módulo:

$$\textit{Alternativa} = \textit{Códon} \bmod n$$

onde n é o número de alternativas disponíveis para uma dada regra. O não terminal é então substituído pelo resultado da regra escolhida. Esse processo se repete até que não restem símbolos não terminais na expressão. Caso o genótipo seja totalmente consumido antes do término da derivação, um mecanismo de *wrapping* é ativado, e os codons são lidos novamente desde o início do vetor.

Esse mecanismo de mapeamento confere à EG vantagens. A validade sintática é garantida, pois o processo de derivação só pode gerar soluções que estejam de acordo com

as regras da gramática. A modularidade é obtida ao permitir que símbolos não terminais representem subestruturas complexas e reutilizáveis, possibilitando que a evolução opere com blocos construtivos de alto nível.

3.2. A Linguagem SANDL

Para representar e manipular arquiteturas de redes neurais, desenvolvemos a SANDL — um formato declarativo e legível por humanos que atua como representação fenotípica dessas arquiteturas. A SANDL é otimizada para a geração e modificação programática de redes neurais voltadas à previsão de séries temporais. A estrutura de qualquer programa SANDL válido é definida formalmente por uma gramática na Backus-Naur Form (BNF), que serve de base para o processo de Evolução Gramatical. Isso garante que qualquer arquitetura gerada seja sintaticamente válida. Um trecho dessa gramática, que ilustra seu design baseado em componentes, é apresentado na Fig. 1.

```
<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
<l1>            ::= 'l1' '=' FLOAT
<l2>            ::= 'l2' '=' FLOAT
```

Figura 1. Pequeno trecho da gramática da SANDL.

Uma camada começa pela indicação do seu tipo, que pode ser, por exemplo, `dense`, `lstm`, `gru`, `conv1d` ou `tcn`. Em seguida, entre chaves, são listados zero ou mais parâmetros, separados por ponto e vírgula. Esses parâmetros especificam propriedades da camada, como a função de ativação (`activation=relu`), a taxa de *dropout* (`dropout=0.3`) ou os coeficientes de regularização L1 e L2 (`l1=0.0001, l2=0.0005`). A função de ativação, por sua vez, deve ser escolhida entre um conjunto pré-definido, como `linear`, `relu`, `sigmoid` ou `tanh`. Essa especificação formal assegura que toda camada descrita na SANDL siga um formato sintaticamente válido. Assim, é possível representar desde camadas com múltiplos parâmetros — como `dense activation=relu; dropout=0.25;` — até camadas sem parâmetros — como `tcn { }`.

A gramática também possibilita a inclusão de novos tipos de camada e parâmetros, o que confere flexibilidade para representar diferentes arquiteturas. Essa estrutura permite que o processo evolutivo construa arquiteturas complexas por meio da combinação de múltiplas camadas e configurações. O Código 1 apresenta um exemplo completo e válido de arquitetura definida na SANDL.

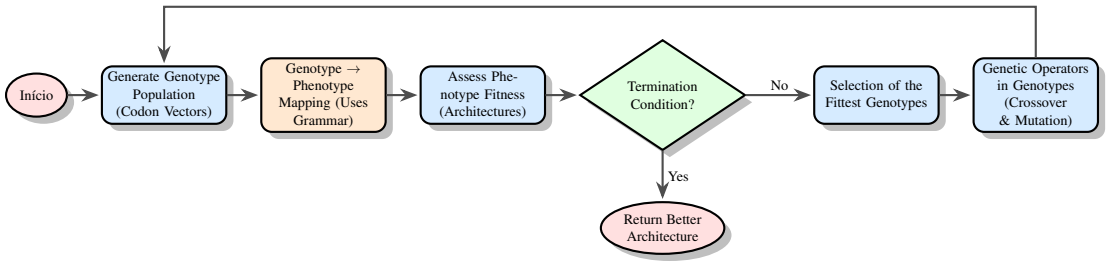
Listing 1. Example of a neural network architecture defined with SANDL.

```
neuralnet {
    input { features = 5; }
    dense { units = 10; activation = relu; }
    output { units = 3; activation = softmax; }
}
```

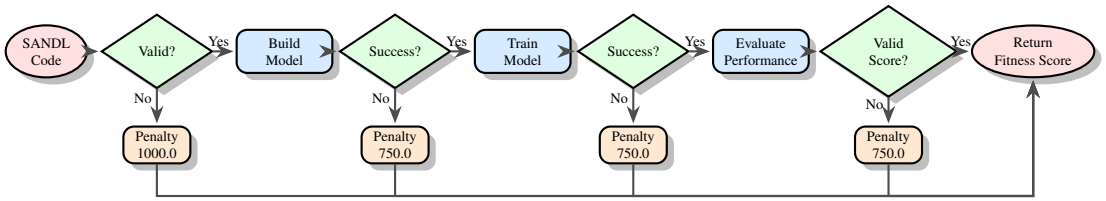
SANDL é uma linguagem declarativa que descreve o que a arquitetura deve ser, sem especificar como construí-la. Sua estrutura é hierárquica e utiliza blocos aninhados para agrupar parâmetros por camada ou seção, seguindo o modelo modular das redes neurais. Cada bloco contém pares `chave=valor` para configuração de hiperparâmetros, permitindo a definição de ajustes globais ou específicos de cada camada. A BNF define a estrutura, enquanto a verificação de valores, a compatibilidade de parâmetros e a coerência do modelo é realizada por regras semânticas aplicadas posteriormente no processo de compilação ou interpretação.

3.3. O Processo de Neuroevolução Gramatical

O processo de NEG integra os princípios da EG com a especificação SANDL, criando um fluxo totalmente automatizado para o projeto de arquiteturas de redes neurais. Esse funcionamento geral é ilustrado na Figura 2, que apresenta tanto o ciclo evolutivo de alto nível quanto as etapas detalhadas de avaliação de *fitness*.



(a) *Loop* principal, no qual genótipos são mapeados para fenótipos de redes neurais e avaliados quanto ao *fitness*.



(b) Processo de avaliação de *fitness* que apresenta o procedimento detalhado para verificar a qualidade de arquiteturas neurais por meio da análise do código SANDL, construção do modelo, treinamento e medição de desempenho.

Figura 2. GENNAS framework architecture.

O ciclo evolutivo principal, mostrado na Figura 2a, inicia-se com a criação de uma população de genótipos, onde cada indivíduo é representado por um vetor de inteiros (Codons Vectors). Para cada indivíduo, o processo central de mapeamento da GE converte o genótipo em um fenótipo — uma arquitetura de rede neural definida em SANDL — aplicando as regras de produção da gramática formal. Essa arquitetura gerada é então avaliada para medir sua qualidade. O ciclo segue até que seja atendida uma condição de parada, como atingir o número máximo de gerações. Caso a condição não seja satisfeita, os genótipos mais bem avaliados são selecionados (por exemplo, via seleção por torneio), e operadores genéticos, como *crossover* de ponto único e mutação, são aplicados para gerar uma nova população. Esse processo se repete, buscando arquiteturas cada vez mais adequadas.

A avaliação de fitness é um elemento central do método, garantindo que as arquiteturas sejam sintaticamente corretas, treináveis e eficazes para a tarefa de previsão. Para cada fenótipo SANDL, o processo ocorre em quatro etapas:

1. Validação – O código SANDL é analisado para verificar sua validade sintática. Em caso de erro, é atribuída uma penalidade alta, e a avaliação do indivíduo é encerrada.
2. Construção do Modelo – Se o código for válido, o sistema tenta construir a rede neural correspondente (por exemplo, em Keras/TensorFlow). Falhas, como conexões incompatíveis entre camadas, resultam em penalização.
3. Treinamento – O modelo construído é treinado com dados de séries temporais. Problemas como instabilidade numérica também levam a penalidades.
4. Avaliação de Desempenho – O modelo treinado é avaliado em um conjunto de validação. Se o resultado for inválido (por exemplo, NaN), aplica-se nova penalidade; caso contrário, o valor obtido para a métrica definida (como RMSE) é registrado como *fitness* final.

Esse esquema de avaliação, baseado em penalizações, direciona a busca evolutiva para arquiteturas corretas, treináveis e com desempenho adequado na tarefa de previsão.

4. Estudo de Caso e Resultados

Para validar a eficácia do *framework* GENNAS, foi realizado um experimento com dados sintéticos, comparando um modelo estatístico tradicional (ARIMA) e uma arquitetura de rede neural gerada automaticamente pelo sistema proposto.

4.1. Configuração do Experimento

O experimento foi realizado utilizando um conjunto de dados sintético de séries temporais, com 500 observações diárias no período de 1º de janeiro de 2020 a 14 de maio de 2021. O conjunto contém três variáveis de entrada (nomeadas como `feature_1`, `feature_2`, `feature_3`) e uma variável alvo contínua, simulando um cenário de previsão comum em aplicações reais. O pré-processamento seguiu práticas usuais para séries temporais: nas abordagens com redes neurais, as variáveis foram padronizadas utilizando normalização z-score para garantir estabilidade no treinamento. Os dados foram segmentados em sequências de 20 passos temporais, com horizonte de previsão de 1 passo, gerando pares de entrada e saída para aprendizado supervisionado. A divisão foi sequencial, preservando a ordem temporal, com 80% dos dados (400 amostras) destinados ao treinamento e 20% (100 amostras) ao teste.

4.1.1. Configuração do ARIMA

Como referência estatística, utilizou-se o modelo AutoRegressive Integrated Moving Average (ARIMA). A ordem ótima do modelo foi selecionada por meio de *grid search*, com base no Akaike Information Criterion (AIC) [Khalid and Sarwat 2021]. O teste de Dickey-Fuller aumentado indicou não estacionariedade (`valor-p = 0,085`), exigindo primeira diferenciação. A melhor configuração encontrada foi ARIMA(3,0,3), com AIC=-559,41 e BIC=-527,48, equilibrando complexidade do modelo e qualidade do ajuste.

As arquiteturas candidatas de redes neurais foram avaliadas com base no RMSE sobre os dados de validação, aplicando penalizações para modelos inválidos ou com falha no treinamento. Os principais parâmetros de configuração incluíram tamanho de

população de 50 indivíduos, máximo de 100 gerações, probabilidade de *crossover* de 0,8, probabilidade de mutação de 0,1 e tempo limite de avaliação de *fitness* de 5 minutos. Utilizou-se *early stopping* com paciência de 10 épocas.

4.1.2. Configuração no GENNAS

O *framework* GENNAS foi configurado com uma gramática desenvolvida especificamente para arquiteturas de previsão de séries temporais. O processo evolutivo utilizou uma estratégia de busca baseada em população para explorar o espaço de possíveis configurações de redes neurais. A função de *fitness* avaliou as arquiteturas candidatas com base no desempenho de Root Mean Square Error (RMSE) nos dados de validação, aplicando penalidades para arquiteturas inválidas ou falhas no treinamento.

Os principais parâmetros de configuração incluíram:

- Tamanho da população: 50 indivíduos
- Número máximo de gerações: 100
- Probabilidade de *crossover*: 0,8
- Probabilidade de mutação: 0,1
- Tempo limite para avaliação de *fitness*: 5 minutos por avaliação
- *Early stopping*: paciência de 10 épocas durante o treinamento

O desempenho do modelo foi avaliado utilizando três métricas complementares: RMSE, Mean Absolute Error (MAE) e Coeficiente de Determinação (R^2). Essas métricas fornecem uma avaliação abrangente da precisão das previsões (RMSE, MAE) e da variância explicada (R^2), permitindo uma comparação consistente entre diferentes abordagens de modelagem.

4.2. Resultados

O processo evolutivo do GENNAS convergiu para uma arquitetura de rede neural ótima após a exploração sistemática do espaço de projeto. A arquitetura descoberta automaticamente, expressa na notação SANDL, é apresentada no Listing 2.

Listing 2. Arquitetura de rede neural gerada pelo GENNAS no formato SANDL.

```
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; }
}
```

O bloco `input` define que o modelo receberá 3 variáveis de entrada (`features = 3`) e que cada amostra de entrada será composta por uma janela temporal de 20 passos (`sequence_length = 20`), isto é, o modelo observa um histórico de 20 *timesteps* para prever o próximo valor.

A primeira camada LSTM contém 64 unidades e taxa de *dropout* de 0,2, com `return_sequences = true`. Isso significa que ela retorna a sequência completa

de saídas, preservando a informação temporal para ser processada pela próxima camada LSTM. Essa camada atua na extração inicial de padrões temporais complexos, como tendências de curto e médio prazo. A segunda camada LSTM reduz para 32 unidades e aplica *dropout* de 0,1, com `return_sequences = false`. A redução no número de unidades implementa uma forma de compressão hierárquica: as representações temporais extraídas pela primeira LSTM são condensadas, enfatizando as informações mais relevantes e descartando ruídos e o retorno indica que apenas o último estado oculto é passado para as camadas seguintes, adequado para tarefas de previsão de passo único (*one-step ahead forecasting*). A arquitetura evoluída apresenta uma configuração de duas camadas LSTM, com redução progressiva no número de unidades ($64 \rightarrow 32$), implementa uma estratégia hierárquica de extração de características, em que a primeira camada captura padrões temporais mais detalhados, enquanto a segunda aprende abstrações de nível mais alto.

A seguir, a camada *dense* com 16 unidades e ativação RELU transforma a representação extraída pelas LSTMs em um espaço latente de menor dimensão, permitindo a combinação não linear das características aprendidas. Essa camada funciona como um mapeador de alto nível entre a representação temporal e a saída final. Ainda aplica-se uma camada *dropout* com taxa de 0,1, reforçando a regularização antes da saída. Esse segundo ponto de *dropout* ajuda a reduzir a sobreajuste, já que atua sobre representações mais compactas e abstratas do modelo. A utilização de camadas *dropout* com taxas diferentes (0,2; 0,1; 0,1) fornece regularização adaptativa, reduzindo o risco de *overfitting* sem comprometer a capacidade de representação do modelo.

Finalmente, a camada *output* define que haverá 1 unidade de saída com ativação linear, apropriada para tarefas de regressão onde a previsão é um valor contínuo. No total, a arquitetura possui 30.369 parâmetros treináveis, valor significativamente superior aos 6 parâmetros do modelo ARIMA utilizado como referência. No entanto, como demonstrado nos resultados de desempenho, esse aumento de complexidade é compensado por ganhos expressivos na precisão preditiva e na capacidade de generalização.

Tabela 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%**	–

A Tabela 1 apresenta uma comparação abrangente do desempenho dos modelos em todas as métricas de avaliação. Os resultados demonstram a superioridade da arquitetura neural gerada pelo GENNAS em relação à abordagem estatística tradicional. A rede neural obteve ganhos expressivos em todas as métricas. O RMSE apresentou redução de 58,85%, passando de 0,2813 para 0,1157, o que indica previsões mais precisas. O MAE melhorou 64,09%, refletindo redução consistente nos erros de previsão. O coeficiente de determinação (R^2) apresentou o avanço mais significativo, subindo de -0,1971 para 0,8045 — uma mudança de um desempenho inferior ao de um preditor baseado na média simples para um modelo com forte capacidade explicativa. A arquitetura gerada

pelo GENNAS convergiu em 16 épocas com *early stopping*. A perda final no treinamento foi de 0,0247, enquanto a perda de validação estabilizou em 0,0133, sugerindo boa capacidade de generalização sem ocorrência de *overfitting*.

Figura 3. Análise comparativa do desempenho do ARIMA e da rede neural GENNAS: (a) previsões do modelo ARIMA apresentando baixo acompanhamento dos valores reais, com previsões relativamente planas; (b) previsões da rede neural demonstrando maior precisão na captura de padrões temporais; (c) comparação de métricas de desempenho evidenciando melhorias obtidas pela arquitetura gerada pelo GENNAS nas métricas RMSE, MAE e R^2 em relação ao ARIMA; (d) distribuição dos resíduos mostrando erros mais próximos de uma distribuição normal para a rede neural em comparação ao modelo ARIMA.

A Figura 3 ilustra a qualidade das previsões de ambos os modelos por meio de gráficos de séries temporais e visualizações de desempenho. As previsões da rede neural apresentam melhor acompanhamento dos valores reais, especialmente na captura de padrões temporais complexos que o modelo linear ARIMA não conseguiu representar de forma adequada, na melhoria das métricas e na análise dos resíduos revela que a rede neural produz erros mais próximos de uma distribuição normal e com variância reduzida em comparação ao modelo ARIMA, indicando melhor ajuste.

5. Conclusão

Este artigo apresentou o GENNAS, um novo *framework* para busca automatizada de arquiteturas neurais, desenvolvido especificamente para aplicações de previsão de séries temporais. Por meio da integração da evolução gramatical com a linguagem de domínio específico SANDL, nossa abordagem enfrenta o desafio central de projetar arquiteturas de redes neurais eficazes sem exigir amplo conhecimento especializado ou ajuste manual de hiperparâmetros. A validação experimental demonstra benefícios significativos da busca evolutiva de arquiteturas neurais em relação a métodos estatísticos tradicionais. A arquitetura gerada pelo GENNAS obteve uma melhoria de 58,85% no RMSE em comparação com a linha de base ARIMA.

O trabalho apresenta três contribuições principais. Primeiro, a linguagem SANDL oferece um formato estruturado e legível por humanos para representar arquiteturas neurais, facilitando a reprodutibilidade e a transferência de conhecimento. Segundo, a abordagem baseada em evolução gramatical possibilita explorar de forma eficiente o espaço de arquiteturas. Terceiro, o *framework* abrangente de avaliação de *fitness* assegura uma análise robusta das arquiteturas em condições realistas.

Como trabalhos futuros, propõe-se ampliar o *framework* para lidar com cenários reais de previsão multivariada, incorporar mecanismos de quantificação de incerteza e desenvolver gramáticas especializadas para domínios de aplicação específicos, como séries temporais financeiras ou análise de dados de sensores. Além disso, investigar abordagens híbridas que combinem busca evolutiva com otimização baseada em gradiente.

Agradecimentos

A professora Komati agradece ao CNPq pela bolsa DT-2 (nº 302726/2023-3) e pelo projeto nº 407742/2022-0; e agradece à FAPES pelo projeto nº 1023/2022 P:2022-8TZV6.

Referências

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

Galván, E. and Mooney, P. (2021). Neuroevolution in deep neural networks: Current trends and future challenges. *IEEE Transactions on Artificial Intelligence*, 2(6):476–493.

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

Lim, B. and Zohren, S. (2021). Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209.

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

Soltanian, K., Ebnenasir, A., and Afsharchi, M. (2022). Modular grammatical evolution for the generation of artificial neural networks. *Evolutionary computation*, 30(2):291–327.

Stanley, K. O., Clune, J., Lehman, J., and Miikkulainen, R. (2019). Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35.

Wang, H., Lei, Z., Zhang, X., Zhou, B., and Peng, J. (2019). A review of deep learning for renewable energy forecasting. *Energy Conversion and Management*, 198:111799.

Winter, B. D. and Teahan, W. J. (2025). Task-specific activation functions for neuroevolution using grammatical evolution. *arXiv:cs.NE,e2503.10879*.