



Overfitting Dynamics in Recurrent Neural Networks: A Statistical and Experimental Approach

Orgeta Gjermëni

Department of Mathematics and Physics,
University of Vlora "Ismael Qemali",
Vlora 9400, Albania

Received: 5 January 2026 / Revised: 20 February 2026 / Accepted: 3 March 2026 / Published: 25 March 2026
© 2026 Orgeta Gjermëni

Doi: 10.56345/ijrdv13n116

Abstract

Overfitting remains a key challenge in applying Recurrent Neural Networks (RNNs) to sequential forecasting tasks, including carbon emissions modeling. While Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) architectures are widely used, comparative analyses of their overfitting behavior across architectural variants remain limited. The present analysis addresses this gap by examining how structural variables such as model architecture, layer depth, architecture type, model family, and the number of hidden units, together with their two-way interactions with unit configuration, influence overfitting behaviour in RNNs applied to a univariate time series of carbon dioxide (CO₂) emissions from land-use change in Albania. The dataset covers the period from 1850 to 2022. Preprocessing steps included Isolation Forest-based outlier detection, LSTM-based imputation, first differencing, Yeo-Johnson transformation, and Min-Max normalization. Six RNN architectures were evaluated, including single-layer models (GRU and LSTM) and their homogeneous and hybrid two-layer variants. Each architecture was trained across hidden unit values, resulting in 402 model instances under a unified configuration. Overfitting ratios were calculated as the ratio of test to training values for each of the four performance metrics: root mean squared error, symmetric mean absolute percentage error, mean absolute scaled error, and normalized mean absolute error. Their distributional properties were also assessed. A non-parametric multivariate analysis of variance was conducted to examine both main effects and two-way interactions, followed by pairwise comparisons within statistically significant structural factors. The results showed that model architecture, layer depth, architecture type, and model family significantly influence overfitting behaviour. Although the number of hidden units did not have a significant main effect, consistent interaction effects suggest that their impact depends on the architectural configuration. Pairwise comparisons revealed that hybrid and homogeneous architectures differed significantly from simple models. No significant difference was found between hybrid and homogeneous architectures, indicating greater similarity between the latter two. These findings emphasize the importance of aligning architectural design with hyperparameter selection when developing RNN-based forecasting models. Methodologically, the study demonstrates the utility of multivariate non-parametric analysis in characterizing generalization behavior. The research insights provided practical guidance for constructing more robust and generalizable RNNs for environmental forecasting and similar applications.

Keywords: CO₂ emissions; Hyperparameter interaction; Model architecture; Overfitting;
Recurrent neural networks; Time series forecasting

1. Introduction

Climate change and its associated impacts on public health and ecosystems are increasingly linked to carbon dioxide (CO₂) emissions. Although multiple sectors contribute to environmental degradation, CO₂ remains the dominant greenhouse gas, making its mitigation central to global climate strategies (Gavurova et al., 2021).

Albania's economic structure, which includes mining, industry, agriculture, forestry, and tourism, also influences its environmental footprint. Despite maintaining the lowest average CO₂ emissions per capita among Balkan countries (Mitic et al., 2020), Albania represents a critical context for examining emission dynamics and guiding early-stage mitigation policies to sustain its low-carbon pathway. Land-use dynamics are crucial in the global carbon cycle, functioning as sources and sinks of atmospheric CO₂. Deforestation, reforestation, land abandonment, and wood harvesting significantly alter CO₂ fluxes between land and atmosphere, underscoring the importance of land management for carbon policy and sustainability planning (Li et al., 2024; deB Richter Jr & Houghton, 2011).

Recurrent Neural Networks (RNNs) are a key class of Deep Learning (DL) models in the broader Machine Learning (ML) domain, particularly suited to modeling sequential and temporal data (Goodfellow et al., 2016). Among RNN variants, Long Short-Term Memory (LSTM) networks (Hochreiter & Schmidhuber, 1997) and Gated Recurrent Unit (GRU) networks (Cho et al., 2014) have gained widespread adoption due to their ability to capture long-term dependencies by overcoming the vanishing and exploding gradient problems. A brief theoretical background on LSTM and GRU can be found in Mateus et al. (2021), Pierre et al. (2023), and Shahi et al. (2020). Comparative studies suggest that performance differences between LSTM and GRU vary with task characteristics: some report superior results for GRU (Abumohsen et al., 2023; Kurniasari et al., 2024; Mateus et al., 2021; Yang et al., 2022), others favor LSTM (Kristiani, et al., 2022), and several indicate comparable performance (Cican et al., 2023; Gao et al., 2021; Shahi et al., 2020). Recent findings also highlight the potential of hybrid architectures that combine GRU and LSTM cells (Trivedi & Patel, 2022; Widiyasari & Efendi, 2024).

Forecasting carbon emissions is one of the many areas where deep learning models, particularly RNNs, have been applied with encouraging results (Aamir, et al., 2022; Nassef et al., 2023). Nonetheless, much of the existing literature primarily focuses on predictive accuracy, with limited attention to systematically quantifying overfitting phenomena across different RNN architectures (Nassef et al., 2023; Luo et al., 2023; Gao et al., 2021).

Although overfitting is a critical concern in ML, where models may perform well on training data but poorly generalize to unseen data, few studies have rigorously assessed generalization behavior using multiple metrics and statistical methods. The interaction among model architecture, structural complexity (e.g., the number of layers), model family distinctions (e.g., GRU-based versus LSTM-based designs), and hyperparameter configurations (e.g., the number of hidden units) remains insufficiently explored, particularly in studies that treat architecture and unit configuration as explicit experimental factors. Understanding these relationships is crucial for designing resilient forecasting systems, including applications involving nonlinear univariate time series, such as CO₂ emissions from land-use change.

This study contributes a controlled inferential comparison of architecture-dependent overfitting behavior, rather than an accuracy-focused forecasting benchmark. Given historical constraints on series length, a single univariate design is used to isolate architecture, depth, family, and unit effects while limiting confounding from multivariate feature engineering and cross-dataset heterogeneity. Thus, conclusions are framed as within-study evidence on relative overfitting dynamics in this context.

Empirically, the analysis evaluates how key structural factors influence the overfitting behavior of RNNs, specifically focusing on model architecture ("Model"), layer depth ("Layers"), architecture type ("Architecture Type"), model family ("Family"), and the number of hidden units ("Units"). Six model architectures were evaluated: GRU, GRU-GRU, GRU-LSTM, LSTM, LSTM-GRU, and LSTM-LSTM, representing single-layer, homogeneous two-layer, and hybrid two-layer designs. These were further categorized using the previously mentioned structural descriptors "Layers" (single vs. two-layer), "Architecture Type" (simple, homogeneous, hybrid), and "Family" (GRU-based vs. LSTM-based). In addition to analyzing these main effects, the study investigates two-way interactions between unit configuration and each structural factor.

Accordingly, the present study is not framed as a forecasting benchmark against classical statistical baselines. Such benchmarking is outside the scope of the present study. Within this scope, the following research questions are addressed:

- (RQ1): Does model architecture significantly affect overfitting ratios across multiple performance metrics?
- (RQ2): Which specific pairs of model architectures differ significantly in overfitting behavior?
- (RQ3): Does the number of hidden units independently influence overfitting behavior?
- (RQ4): Is there a significant interaction effect between model architecture and the number of units?
- (RQ5): Do single-layer and two-layer models differ in generalization behavior?
- (RQ6): Is there a significant interaction between layer depth and the number of units?
- (RQ7): Do hybrid architectures generalize differently than homogeneous or simple ones?
- (RQ8): Is there a significant interaction between architecture type and unit configuration?

- (RQ9): Do GRU-based and LSTM-based model families differ in overfitting behavior?
 (RQ10): Is there a significant interaction between model family and unit configuration?

The rest of the paper is organized as follows: the Methods section describes the dataset, preprocessing steps, model configurations, and statistical procedures used to ensure comparability across architectures. The Results section presents empirical findings on overfitting behavior across the six model configurations using non-parametric multivariate analysis. The Discussion section interprets the findings in relation to the research questions, acknowledges limitations mainly related to external validity, and outlines future directions. Finally, the Conclusion summarizes key findings and contributions.

2. Methods

The current study adopted a structured experimental design. The research framework included dataset acquisition and preprocessing, model development and training, performance evaluation, and statistical analysis of patterns of overfitting.

2.1 Dataset and Preprocessing

The dataset utilized in this analysis comprises annual CO₂ emissions from land-use change per capita (CO₂-E-LUC) for Albania, covering the period from 1850 to 2022. These data were obtained from Our World in Data (2023), a source also used in prior studies (Mitic et al., 2020; Nassef et al., 2023). The dataset includes two variables, “year” and “annual CO₂-E-LUC per capita”, resulting in 173 annual observations.

Preliminary investigations into the time series characteristics (Gjermëni, 2024), revealed that the series was initially both non-stationary and non-linear. While first differencing successfully induced stationarity, the non-linear structure remained evident even after transformation.

A structured preprocessing workflow was applied to ensure data quality and compatibility with RNN architectures, as summarized in Table 1. Outlier detection was conducted using the Isolation Forest algorithm (Liu et al., 2008).

The detected outlier values were subsequently replaced with missing values (NA). Imputation of missing values was performed using an LSTM-based model (Allaire & Cholet, 2024; Allaire & Tang, 2024; Hochreiter & Schmidhuber, 1997; Hyndman & Koehler, 2006), detailed in Table 2.

Table 1. Preprocessing steps for annual CO₂-E-LUC time series

Step	Preprocessing	Technical description
1	Outlier detection via Isolation Forest	Applied <code>isolation_forest()</code> (package: <code>isotree</code> (Cortes, 2024)) with <code>ntrees = 50, 100, 150, 200</code> ; selected optimal trees (100) based on max outliers detected; used 95th percentile as threshold
2	Outlier replacement	Outliers identified were replaced with NA in the original series
3	LSTM-based imputation	Further details are given in Table 2
4	First differencing	Applied <code>diff()</code> function to the imputed series
5	Yeo-Johnson transformation	Applied <code>yeojohnson()</code> from the <code>bestNormalize</code> package
6	Min-Max normalization	Transformed using: $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$

The effectiveness of the LSTM-based imputation was quantitatively validated using multiple performance metrics. A root mean squared error of 0.0981, a symmetric mean absolute percentage error of 2.11%, and a mean absolute scaled error of 0.2018 were achieved. The normalized mean absolute error was 0.4074%, and the coefficient of determination based on residuals reached 99.29%.

The series was differenced once to achieve stationarity, as confirmed by the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test (Pfaff, 2024; Kwiatkowski et al., 1992), then transformed using the Yeo-Johnson method (Yeo & Johnson, 2000; Peterson, 2021) to reduce skewness, and scaled to [0, 1] via Min-Max normalization.

Table 2. LSTM imputation model settings

Category	Parameter	Value
Environment setup	R seed	<code>set.seed(12345678)</code>
	TensorFlow seed	<code>tf\$random\$set_seed(12345678)</code>
	Determinism mode	<code>tf\$config\$experimental\$enable_op_determinism()</code>

Category	Parameter	Value
	Clear session	tf.keras.backend.clear_session()
	CPU-only	Not enforced (no CUDA_VISIBLE_DEVICES)
	Threads	"OMP_NUM_THREADS" = 1, "TF_NUM_INTEROP_THREADS" = 1, "TF_NUM_INTRAOP_THREADS" = 1
Model initialization	Initializer	initializer_glorot_uniform(seed = 12345678)
Input configuration	Timesteps	1
	Input shape	c(timesteps, 1)
LSTM layer	Units	30
	Activation	'tanh'
	Recurrent activation	'sigmoid'
	Kernel initializer	initializer_glorot_uniform(seed = 12345678)
	Recurrent initializer	initializer_glorot_uniform(seed = 12345678)
	Return sequences	FALSE
Output (Dense) layer	Units	1
	Activation	'linear'
	Kernel initializer	initializer_glorot_uniform(seed = 12345678)
Training configuration	Loss function	'mean_absolute_error'
	Optimizer	Adam (learning_rate = 0.001)
	Epochs	100
	Batch size	16
	Early stopping	patience = 3, monitor = "val_loss"
Imputation method	Strategy	Recursive prediction using a trained model

2.2 Experimental Setup and Architecture Configuration

Experiments were conducted on a 64-bit Windows system with an Intel(R) Core (TM) i7-10750H CPU at 2.60 GHz and 16 GB RAM, operating in a CPU-only environment. All models were implemented in R (R Core Team, 2023) using the Keras API and the TensorFlow backend, ensuring deterministic execution by using fixed random seeds and environment settings.

As introduced earlier, six RNN model architectures were evaluated. These configurations represent single-layer models (GRU, LSTM), two-layer homogeneous models (GRU-GRU, LSTM-LSTM), and two-layer hybrid models (GRU-LSTM, LSTM-GRU). The hidden layer configurations were kept consistent across models, with the number of hidden units varying across a predefined range. A unified model configuration was applied across all architectures, as summarized in Table 3.

Table 3. Unified configuration for RNN architectures

Layer type	Parameter	Value
First hidden layer (GRU or LSTM) layer_gru() / layer_lstm()	units	{10,20,30,40, ...,600} and { 2 ⁴ , 2 ⁵ , 2 ⁶ , ... 2 ¹⁰ }.
	input_shape	c(timesteps, 1)
	dropout	0.1
	recurrent_dropout	0.1
	kernel_initializer	initializer_glorot_uniform(seed = 123456789)
	recurrent_initializer	initializer_glorot_uniform(seed = 123456789)
	activation	'tanh'
	recurrent_activation	'sigmoid'
Second hidden layer (GRU or LSTM) layer_gru() / layer_lstm()	units	{10,20,30,40, ...,600} and { 2 ⁴ , 2 ⁵ , 2 ⁶ , ... 2 ¹⁰ }.
	dropout	0.1
	recurrent_dropout	0.1
	kernel_initializer	initializer_glorot_uniform(seed = 123456789)
	recurrent_initializer	initializer_glorot_uniform(seed = 123456789)
	activation	'tanh'
	recurrent_activation	'sigmoid'
	Output layer (Dense) layer_dense()	units
kernel_initializer		initializer_glorot_uniform(seed = 123456789)
kernel_regularizer		regularizer_l1_l2(l1 = 0.001, l2 = 0.001)
activation		'relu'

To ensure reproducibility and deterministic behavior during model training, fixed random seeds were applied using both R's `set.seed()` function and TensorFlow's `tf.random.set_seed()`. GPU computation was disabled (`Sys.setenv("CUDA_VISIBLE_DEVICES" = "-1")`) and TensorFlow's experimental deterministic mode was enabled. Table 4 summarizes the reproducibility setup. Separate random seeds were used: 12345678 for the LSTM imputation phase and 123456789 for the modelling phase.

Table 4. Reproducibility setup

Component	Configuration details
Random seed setup	<code>Sys.setenv("CUDA_VISIBLE_DEVICES" = "-1"),</code> <code>set.seed(123456789),</code> <code>tf.random.set_seed(123456789)</code>
Deterministic execution	<code>tf.config.experimental.enable_op_determinism()</code>
Weight initialization	<code>initializer_glorot_uniform(seed = 123456789)</code> applied across all model layers

2.3 Training and Evaluation Strategy

The models were trained to minimize the mean absolute error between predicted and actual values. Based on prior studies, train-test split ratios such as 60/40 (Abduljabbar et al., 2021), 70/30 (Raubitzek & Neubauer, 2021; Kurniasari et al., 2024; Nassef et al., 2023; Mateus et al., 2021; Cican et al., 2023), and 80/20 (Gao et al., 2021; Widiarsari & Efendi, 2024) have been explored. A range of training/testing ratios (from 10/90 to 90/10) was systematically evaluated for ML models using root mean squared error, mean absolute error, and the correlation coefficient, with findings indicating that the 70/30 split provided the highest predictive performance (Nguyen et al., 2021).

A 75/25 train-test split was adopted in our study, based on preliminary exploratory experiments that indicated superior predictive performance. The supervised learning setup employed a single lag (timesteps = 1). Model training used the Adam optimizer with a fixed learning rate of 0.0001 and employed backpropagation through time to propagate gradients across temporal dependencies. Training was limited to 2000 epochs (Keydana et al., 2022), with early stopping applied after three consecutive epochs without improvement in validation loss to prevent overfitting. A batch size of 32 and a 20% validation split from the training set were used to monitor model performance during training (Goodfellow et al., 2016). Finally, a recursive forecasting strategy was implemented, feeding the predicted value at each timestep as input for the next.

2.4 Evaluation Metrics

Four commonly used evaluation metrics were selected: root mean squared error (RMSE), symmetric mean absolute percentage error (sMAPE), mean absolute scaled error (MASE), and normalized mean absolute error (NMAE). RMSE provides a higher penalty to large errors; sMAPE handles percentage errors symmetrically; MASE, as recommended in (Hyndman & Koehler, 2006) and supported further in (Franses, 2016). MASE is a reliable metric because it compares model results to a simple baseline. Values below one show better performance than the baseline. NMAE was also included because it standardizes error by the data's range, making it easier to compare performance across different datasets (Yoo & Oh, 2020).

2.5 Statistical Analysis

The overfitting ratio (*OR*) was defined as the ratio between the test and training RMSE in (Fiorentini et al., 2022):

$$OR = \frac{RMSE_{test}}{RMSE_{train}} \quad (1)$$

where an *OR* value close to one suggests good generalization, while significantly higher values indicate overfitting. Extending this approach, the present study computes overfitting ratios concretely for the four performance metrics RMSE, sMAPE, MASE, and NMAE, as follows:

$$OR_{Metric} = \frac{Metric_{test}}{Metric_{train}} \quad (2)$$

This is consistent with the metric-based overfitting ratio framework discussed in Avrutskiy (2020). Let $M = \{M_1, M_2, \dots, M_6\}$ denote the set of tested model architectures, where $M_1 = GRU, M_2 = GRU - GRU, M_3 = GRU -$

$LSTM$, $M_4 = LSTM$, $M_5 = LSTM - GRU$, and $M_6 = LSTM - LSTM$. Each model architecture was trained and tested across a set of hidden units, $K = \{10, 20, 30, 40, \dots, 600\} \cup \{2^4, 2^5, 2^6, \dots, 2^{10}\}$. Consequently, a total of 402 models were evaluated. The set of error metrics is defined as $E = \{e_1, e_2, e_3, e_4\}$ corresponding to RMSE, sMAPE, MASE, and NMAE, respectively. The overfitting ratio OR_{ij}^k was calculated for each model M_i^k , and each metric e_j . The overfitting vector was defined as:

$$OR_i^k = (OR_{i1}^k, OR_{i2}^k, OR_{i3}^k, OR_{i4}^k). \quad (3)$$

For every fixed pair (i, j) , the distributions $\{OR_{ij}^k\}_{k \in K}$ were subjected to the Anderson-Darling normality test (Anderson & Darling, 1952). Descriptive statistics were computed for each distribution, including measures of central tendency (mean μ , median Q_2), measures of dispersion (standard deviation σ , interquartile range (*IQR*), minimum, and maximum, measures of shape (skewness γ_1 , kurtosis γ_2), and relative variability (coefficient of variation, *CV*).

Further, a multivariate non-parametric approach was then employed to examine whether overfitting ratio profiles significantly varied across structural model variables and their interactions. Given that the sets $\{OR_{ij}^k\}_{k \in K}$ deviate from normality, non-parametric methods were selected. A multivariate non-parametric analysis was conducted using permutational multivariate analysis of variance (PERMANOVA) (Anderson M. J., 2001), employing Bray–Curtis dissimilarity as the distance metric. The PERMANOVA tests were performed using the *adonis2* function from the *vegan* package (Oksanen et al., 2025) in R, with 999 permutations to assess statistical significance.

The structural categorical variables considered to characterize RNN models analyzed in this study are as follows: “Model”, “Layers”, “Architecture Type”, “Family”, and “Units”. Specifically, “Model” referred to the six model architectures evaluated ($M_1, M_2, M_3, M_4, M_5, M_6$); “Layers” distinguished single-layer architectures (M_1, M_4) from two-layer architectures (M_2, M_3, M_5, M_6); “Architecture Type” classified models as homogeneous (M_2, M_6), hybrid (M_3, M_5), or simple (single-layer models M_1, M_4); “Family” grouped models into GRU-based (M_1, M_2, M_3) or LSTM-based (M_4, M_5, M_6) categories. The global PERMANOVA model evaluated the main effects of “Model”, “Layers”, “Architecture Type”, “Family”, and “Units”, as well as their two-way interaction terms (“Model \times Units”, “Layers \times Units”, “Architecture Type \times Units”, and “Family \times Units”).

To detect specific between-group differences, pairwise PERMANOVA comparisons were conducted within each structural variable (“Model”, “Layers”, “Architecture Type”, “Family”, and “Units”), only when the corresponding global PERMANOVA test was statistically significant. All comparisons were based on Bray–Curtis dissimilarities and 999 permutations. P-values were adjusted using Bonferroni correction (Shaffer, 1995), and values below 0.05 were considered statistically significant. Interaction terms were not decomposed into pairwise comparisons due to the complexity and sparsity of subgroup combinations.

3. Results

3.1 Preliminary Distributional Analysis

Before proceeding to the multivariate non-parametric analyses, a preliminary distributional assessment was conducted on the overfitting ratios OR_{ij}^k across the selected models and performance metrics. Table 5 summarizes the results of the Anderson–Darling normality tests and descriptive statistics, calculated separately for each model-metric combination.

The descriptive analysis showed variation in the distributional characteristics of overfitting ratios across models and metrics. While several combinations showed moderate skewness and kurtosis, others deviated from normality, as indicated by the Anderson-Darling test results. Except for two cases, all other p-values from the Anderson-Darling tests fell below the 0.05 threshold, suggesting that the distributions of overfitting ratios significantly deviated from normality. Standard deviation and interquartile range varied across models, suggesting heterogeneity in the stability of model generalization behaviour across different error metrics.

Table 5. Distributional and descriptive analysis of overfitting ratios by model and metric, for $k \in K$

Model	Dataset	μ	Q_2	σ	<i>IQR</i>	<i>min</i>	<i>max</i>	γ_1	γ_2	<i>CV</i>	AD statistic	p-value
M_1	OR_{11}^k	1.200	1.167	0.069	0.000	1.143	1.333	1.394	0.013	0.058	15.206	0.000***
	OR_{12}^k	1.164	1.197	0.090	0.136	0.956	1.261	-0.839	-0.681	0.077	3.638	0.000***
	OR_{13}^k	1.233	1.264	0.107	0.161	1.000	1.354	-0.739	-0.764	0.087	2.766	0.000***
	OR_{14}^k	1.529	1.568	0.131	0.199	1.246	1.671	-0.745	-0.781	0.086	2.968	0.000***

Model	Dataset	μ	Q_2	σ	<i>IQR</i>	<i>min</i>	<i>max</i>	γ_1	γ_2	<i>CV</i>	<i>AD statistic</i>	<i>p-value</i>
M_2	OR_{21}^k	1.180	1.200	0.095	0.083	1.000	1.400	-0.858	0.093	0.080	5.484	0.000***
	OR_{22}^k	1.017	1.014	0.120	0.229	0.870	1.231	0.249	-1.345	0.118	2.108	0.000***
	OR_{23}^k	1.104	1.069	0.144	0.280	0.933	1.364	0.273	-1.357	0.131	2.376	0.000***
	OR_{24}^k	1.367	1.331	0.178	0.340	1.164	1.662	0.285	-1.366	0.130	2.601	0.000***
M_3	OR_{31}^k	1.175	1.200	0.097	0.083	1.000	1.400	-0.892	-0.184	0.083	7.333	0.000***
	OR_{32}^k	1.007	1.000	0.103	0.170	0.878	1.236	0.545	-0.610	0.102	1.372	0.001***
	OR_{33}^k	1.091	1.073	0.130	0.242	0.932	1.349	0.467	-0.923	0.119	1.587	0.000***
	OR_{34}^k	1.351	1.332	0.160	0.307	1.161	1.669	0.489	-0.883	0.119	1.781	0.000***
M_4	OR_{41}^k	1.192	1.167	0.056	0.033	1.143	1.400	3.117	9.102	0.047	11.950	0.000***
	OR_{42}^k	1.092	1.071	0.090	0.155	0.954	1.237	0.144	-1.418	0.082	1.829	0.000***
	OR_{43}^k	1.164	1.135	0.114	0.206	0.987	1.356	0.145	-1.445	0.098	2.007	0.000***
	OR_{44}^k	1.442	1.406	0.141	0.247	1.226	1.667	0.142	-1.451	0.098	1.984	0.000***
M_5	OR_{51}^k	1.201	1.200	0.062	0.033	1.000	1.400	1.099	3.916	0.051	8.804	0.000***
	OR_{52}^k	1.165	1.197	0.076	0.090	0.927	1.260	-1.343	1.230	0.065	3.517	0.000***
	OR_{53}^k	1.260	1.292	0.099	0.117	0.972	1.375	-1.268	0.802	0.079	3.588	0.000***
	OR_{54}^k	1.562	1.608	0.121	0.126	1.204	1.673	-1.352	0.887	0.078	4.463	0.000***
M_6	OR_{61}^k	1.183	1.200	0.084	0.033	1.000	1.400	-0.794	1.499	0.071	7.176	0.000***
	OR_{62}^k	1.088	1.078	0.078	0.111	0.900	1.244	0.000	-0.465	0.072	0.615	0.105
	OR_{63}^k	1.181	1.167	0.104	0.143	0.953	1.354	-0.073	-0.846	0.088	0.749	0.049*
	OR_{64}^k	1.461	1.451	0.127	0.182	1.175	1.664	-0.080	-0.784	0.087	0.636	0.093

Significance codes: ***** $p \leq 0.001$, **** $0.001 < p \leq 0.01$, *** $0.01 < p \leq 0.05$.

3.2 Global Multivariate Analysis (PERMANOVA)

Table 6 summarizes the results of the global PERMANOVA analysis, reporting the main effects of “Model”, “Layers”, “Architecture Type”, “Family”, and “Units”, along with their two-way interactions with “Units”. Figure 1 shows the corresponding R^2 effect sizes, providing a visual comparison of the relative variance explained by each main effect and interaction term. Statistically significant p-values (≤ 0.001) were observed for “Model” ($R^2 = 0.23072$), “Layers” ($R^2 = 0.03718$), “Architecture Type” ($R^2 = 0.04738$), and “Family” ($R^2 = 0.05153$). “Units” item alone was not statistically significant ($p = 0.143$). All two-way interactions involving “Units” were significant, with the strongest effect observed for “Model \times Units” ($R^2 = 0.41769$), followed by “Architecture Type \times Units” ($R^2 = 0.16453$), “Layers \times Units” ($R^2 = 0.15190$), and “Family \times Units” ($R^2 = 0.10721$).

Table 6. PERMANOVA results for main and interaction effects

Effect	Df	Sum of Squares	R^2	F	<i>p-value</i>
Model	5	0.21028	0.23072	23.754	0.001***
Layers	1	0.03389	0.03718	15.446	0.001***
Architecture Type	2	0.04318	0.04738	9.922	0.001***
Family	1	0.04697	0.05153	21.732	0.001***
Units	1	0.00494	0.00542	2.182	0.143
Model \times Units	11	0.38068	0.41769	25.431	0.001***
Layers \times Units	3	0.13844	0.15190	23.761	0.001***
Architecture Type \times Units	5	0.14996	0.16453	15.597	0.001***
Family \times Units	3	0.09771	0.10721	15.931	0.001***

Significance codes: ***** $p \leq 0.001$.

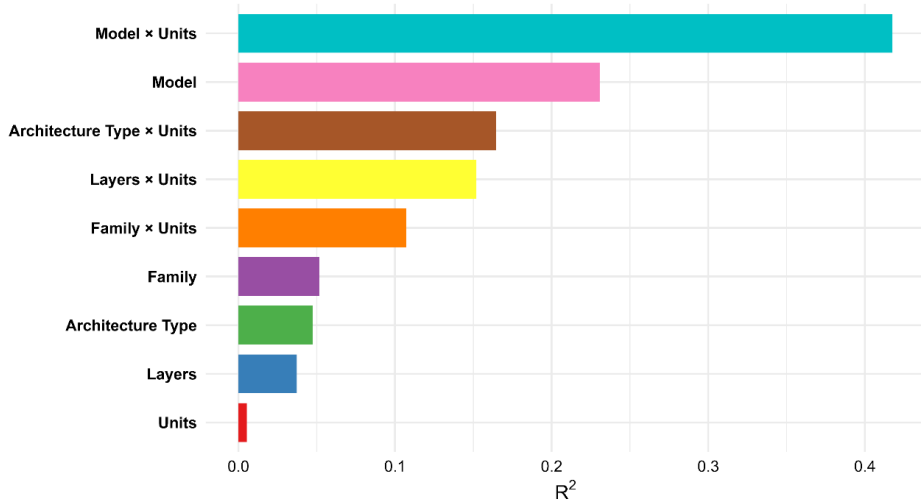


Figure 1. Visual summary of PERMANOVA R^2 values reported in Table 6 for main and interaction effects.

3.3 Pairwise Comparisons

Table 7 presents the pairwise PERMANOVA comparisons conducted within each significant structural factor (“Model”, “Layers”, “Architecture Type”, and “Family”), as shown by the global PERMANOVA analysis. Within the “Model” factor, most pairwise comparisons were significant (adjusted $p = 0.015$), except M_1 vs M_5 (adjusted $p = 1.000$), M_2 vs M_3 (adjusted $p = 1.000$), and M_4 vs M_6 (adjusted $p = 1.000$). The comparison between single-layer and two-layer models under “Layers” was significant (adjusted $p = 0.001$) and hybrid vs. single (adjusted $p = 0.033$) were significant, while homogeneous vs. hybrid was not (adjusted $p = 0.135$). Under “Family”, GRU-based and LSTM-based models showed a significant difference (adjusted $p = 0.001$). Across model-level contrasts, R^2 values varied widely, indicating that some architectural pairs exhibit stronger separation in overfitting profiles than others.

Table 7. Pairwise PERMANOVA comparisons results among models, layers, architecture types, and families

Comparison	F.model	R^2	p-value	Adjusted p-value
Between Models				
M_1 vs M_2	40.718	0.23575	0.001	0.015*
M_1 vs M_3	53.297	0.28763	0.001	0.015*
M_1 vs M_4	12.401	0.08588	0.001	0.015*
M_1 vs M_5	2.672	0.01984	0.095	1.000
M_1 vs M_6	12.298	0.08523	0.002	0.030*
M_2 vs M_3	-0.015	-0.00011	0.919	1.000
M_2 vs M_4	11.136	0.07780	0.001	0.015*
M_2 vs M_5	53.300	0.28764	0.001	0.015*
M_2 vs M_6	13.847	0.09494	0.001	0.015*
M_3 vs M_4	15.785	0.10681	0.001	0.015*
M_3 vs M_5	68.635	0.34209	0.001	0.015*
M_3 vs M_6	19.244	0.12724	0.001	0.015*
M_4 vs M_5	24.388	0.15594	0.001	0.015*
M_4 vs M_6	1.587	0.01188	0.191	1.000
M_5 vs M_6	19.862	0.13079	0.001	0.015*
Between Layers				

Comparison	F.model	R ²	p-value	Adjusted p-value
single-layer vs two-layer	15.446	0.03718	0.001	0.001***
Between Architecture Types				
homogeneous vs hybrid	3.715	0.01377	0.045	0.135
homogeneous vs single	21.404	0.07447	0.001	0.003**
hybrid vs single	6.629	0.02432	0.011	0.033*
Between Families				
GRU-based vs LSTM-based	21.732	0.05153	0.001	0.001***

Significance codes: "****" $p \leq 0.001$, "***" $0.001 < p \leq 0.01$, "**" $0.01 < p \leq 0.05$.

4. Discussion

This statistical analysis indicated that the model architecture ("Model") factor had a statistically significant effect on RNN overfitting (RQ1). Multiple pairs of model architectures showed statistically significant differences, suggesting heterogeneous overfitting dynamics across model architectures (RQ2). The main effect of "Units" was not statistically significant, indicating that the number of hidden units alone does not account for meaningful variation in overfitting behavior (RQ3). A significant "Model × Units" interaction was observed, highlighting that the impact of unit configuration depends on the specific model architecture (RQ4). The main effect of "Layers" was significant, supporting the conclusion that layer depth affects overfitting tendencies (RQ5). The "Layers × Units" interaction was significant, indicating that the influence of unit configuration varies with layer depth (RQ6).

The main effect of "Architecture Type" was statistically significant, and pairwise comparisons showed that hybrid architectures differed significantly from simple ones. In contrast, the difference between hybrid and homogeneous architectures was not statistically significant. This suggests that the generalization behaviour of hybrid models is more similar to that of homogeneous models than to simple (single-layer) models (RQ7). A significant interaction between "Architecture Type" and "Units" was identified, suggesting that unit configuration plays different roles across architecture types (RQ8). The "Family" effect was significant, indicating that GRU-based and LSTM-based architectures exhibit different generalization behavior (RQ9). The "Family × Units" interaction was also significant, suggesting that the role of unit configuration is modulated by model family (RQ10).

These findings emphasize the critical role of architectural design in shaping generalization in RNNs. Rather than treating hidden units as independent tuning parameters, model developers should consider their interaction with deeper structural variables. This is particularly relevant in forecasting tasks involving non-linear, univariate time series, such as CO₂ emissions from land-use change, where optimal generalization depends on the alignment between structural choices and hyperparameter configurations.

The conducted analysis is subject to several limitations that should be acknowledged. First, it focused on a single univariate time series with a relatively small sample size and a dataset characterized by high uncertainty, particularly compared with other components of the global carbon cycle (Ganzenmüller et al., 2022). Second, the analysis was restricted to six specific RNN model architectures, which may limit the generalizability of the findings to other or more recent recurrent network designs. Third, the exploration of hyperparameters was limited: only one optimizer was used, and a restricted set of configurations was examined, focusing mainly on the number of hidden units, without systematically varying the learning rate, batch size, or regularization strategies.

Future research should extend this framework by incorporating larger and more heterogeneous datasets, exploring a broader range of architectures and hyperparameter settings, and applying alternative optimization algorithms and training regimes. Moreover, expanding the analysis to multivariate time series and integrating regularization techniques such as dropout and weight decay could yield deeper insights into the mechanisms of overfitting in recurrent neural networks across diverse domains.

5. Conclusion

This study systematically examined how structural factors and unit configuration affect overfitting behavior in RNNs. The findings revealed that model architecture, layer depth, architecture type, and model family significantly influence overfitting behaviour. At the same time, the number of hidden units alone does not have a substantial independent impact. However, the consistent presence of significant interaction effects underscores that the influence of unit configuration is conditional on structural characteristics. Pairwise comparisons confirmed that hybrid and homogeneous architectures generalize differently from simple models; meanwhile, their mutual similarity suggests that increased depth,

rather than architectural heterogeneity alone, may be the key driver of improved generalization.

By clarifying the role of structural design in shaping overfitting dynamics, this study contributes to a more nuanced understanding of model behavior in deep learning, particularly for univariate forecasting tasks involving nonlinear temporal patterns such as CO₂ emissions from land-use change. The results advance methodological knowledge by demonstrating the value of multivariate non-parametric analysis in distinguishing generalization profiles among RNN architectures.

The research undertaken offers practical guidance for model developers, highlighting the need to integrate architectural considerations into hyperparameter tuning processes. These contributions support the development of more robust and generalizable forecasting models in environmental and other real-world domains where overfitting poses significant challenges.

References

- Amir, M., Bhatti, M. A., Bazai, S. U., Marjan, S., Mirza, A. M., et al. (2022). Predicting the environmental change of carbon emission patterns in South Asia: A deep learning approach using BiLSTM. *Atmosphere*, 13(12), 2011. <https://doi.org/10.3390/atmos13122011>
- Abduljabbar, R. L., Dia, H., & Tsai, P.-W. (2021). Unidirectional and bidirectional LSTM models for short-term traffic prediction. *Journal of Advanced Transportation*, 2021, 5589075. <https://doi.org/10.1155/2021/5589075>
- Abumohsen, M., Owda, A. Y., & Owda, M. (2023). Electrical load forecasting using LSTM, GRU, and RNN algorithms. *Energies*, 16(5), 2283. <https://doi.org/10.3390/en16052283>
- Allaire, J. J., & Chollet, F. (2024). *keras: R interface to 'Keras' (Version 2.15.0)* [R package]. Comprehensive R Archive Network (CRAN). <https://CRAN.R-project.org/package=keras>
- Allaire, J. J., & Tang, Y. (2024). *tensorflow: R interface to 'TensorFlow' (Version 2.16.0)* [R package]. Comprehensive R Archive Network (CRAN). <https://CRAN.R-project.org/package=tensorflow>
- Anderson, M. J. (2001). A new method for non-parametric multivariate analysis of variance. *Austral Ecology*, 26(1), 32–46. <https://doi.org/10.1111/j.1442-9993.2001.01070.pp.x>
- Anderson, T. W., & Darling, D. A. (1952). Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *The Annals of Mathematical Statistics*, 23(2), 193–212. <https://doi.org/10.1214/aoms/1177729437>
- Avrutskiy, V. I. (2020). Preventing overfitting by training derivatives. In K. Arai, R. Bhatia, & S. Kapoor (Eds.), *Advances in Intelligent Systems and Computing* (Vol. 1069, pp. 144–163). Springer. https://doi.org/10.1007/978-3-030-32520-6_12
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., et al. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. <https://doi.org/10.3115/v1/D14-1179>
- Cican, G., Buturache, A.-N., & Mirea, R. (2023). Applying machine learning techniques in air quality prediction: A Bucharest city case study. *Sustainability*, 15(11), 8445. <https://doi.org/10.3390/su15118445>
- Cortes, D. (2024). *isotree: Isolation-based outlier detection (Version 0.6.1-1)* [R package]. Comprehensive R Archive Network (CRAN). <https://CRAN.R-project.org/package=isotree>
- deB Richter, D., & Houghton, R. A. (2011). Gross CO₂ fluxes from land-use change: Implications for reducing global emissions and increasing sinks. *Carbon Management*, 2(1), 41–47. <https://doi.org/10.4155/cmt.10.43>
- Fiorentini, N., Pellegrini, D., & Losa, M. (2022). Overfitting prevention in accident prediction models: Bayesian regularization of artificial neural networks. *Transportation Research Record*, 2677(2), 1455–1470. <https://doi.org/10.1177/03611981221111367>
- Franses, P. H. (2016). A note on the mean absolute scaled error. *International Journal of Forecasting*, 32(1), 20–22. <https://doi.org/10.1016/j.ijforecast.2015.03.008>
- Ganzenmüller, R., Bultan, S., Winkler, K., Fuchs, R., Zabel, F., et al. (2022). Land-use change emissions based on high-resolution activity data substantially lower than previously estimated. *Environmental Research Letters*, 17, 064050. <https://doi.org/10.1088/1748-9326/ac70d8>
- Gao, Y., Wang, R., & Zhou, E. (2021). Stock prediction based on optimized LSTM and GRU models. *Scientific Programming*, 2021, 4055281. <https://doi.org/10.1155/2021/4055281>
- Gavurova, B., Rigelsky, M., & Ivankova, V. (2021). Greenhouse gas emissions and health in the countries of the European Union. *Frontiers in Public Health*, 9, 756652. <https://doi.org/10.3389/fpubh.2021.756652>
- Gjermëni, O. (2024). Assessing non-linearity and stationarity in the time series of Albania's annual emissions of CO₂ from land-use change. *Science & Technology Asia*, 29(4), 39–50. <https://ph02.tci-thaijo.org/index.php/SciTechAsia/article/view/255207>
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. <http://www.deeplearningbook.org>
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/ncoc.1997.9.8.1735>
- Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4), 679–688. <https://doi.org/10.1016/j.ijforecast.2006.03.001>

- Keydana, S., Chollet, F., Kalinowski, T., & Allaire, J. J. (2022). *Deep learning with R* (2nd ed.). Manning Publications.
- Kristiani, E., Lin, H., Lin, J.-R., Chuang, Y.-H., Huang, C.-Y., et al. (2022). Short-term prediction of PM2.5 using LSTM deep learning methods. *Sustainability*, 14(4), 2068. <https://doi.org/10.3390/su14042068>
- Kumiasari, D., Nuraini, M. E., Wamiliana, W., & Nisa, R. K. (2024). A case study: Comparison of LSTM and GRU methods for forecasting oil, non-oil, and gas export values in Indonesia. *Jurnal Teknik Informatika*, 17(2). <https://doi.org/10.15408/jti.v17i2.39098>
- Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1–3), 159–178. [https://doi.org/10.1016/0304-4076\(92\)90104-Y](https://doi.org/10.1016/0304-4076(92)90104-Y)
- Li, L., Awada, T., Zhang, Y., & Paustian, K. (2024). Global land use change and its impact on greenhouse gas emissions. *Global Change Biology*, 30, e17604. <https://doi.org/10.1111/gcb.17604>
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the Eighth IEEE International Conference on Data Mining* (pp. 413–422). IEEE Computer Society. <https://doi.org/10.1109/ICDM.2008.17>
- Luo, R., Wang, J., & Gates, I. (2023). Machine learning for accurate methane concentration predictions: short-term training, long-term results. *Environmental Research Communications*, 5(8), 081003. <https://doi.org/10.1088/2515-7620/acf0a3>
- Mateus, B. C., Mendes, M., Farinha, J. T., Assis, R., & Cardoso, A. M. (2021). Comparing LSTM and GRU models to predict the condition of a pulp paper press. *Energies*, 14(21), 6958. <https://doi.org/10.3390/en14216958>
- Mitić, P., Kostić, A., Petrović, E., & Cvetanović, S. (2020). The relationship between CO₂ emissions, industry, services and gross fixed capital formation in the Balkan countries. *Engineering Economics*, 31(4), 425–436. <https://doi.org/10.5755/j01.ee.31.4.24833>
- Nassef, A. M., Olabi, A. G., Rezk, H., & Abdelkareem, M. A. (2023). Application of artificial intelligence to predict CO₂ emissions: Critical step towards sustainable environment. *Sustainability*, 15(9), 7648. <https://doi.org/10.3390/su15097648>
- Nguyen, Q. H., Ly, H.-B., Ho, L. S., Al-Ansari, N., Le, H. V., et al. (2021). Influence of data splitting on performance of machine learning models in prediction of shear strength of soil. *Mathematical Problems in Engineering*, 2021, 4832864. <https://doi.org/10.1155/2021/4832864>
- Oksanen, J., Simpson, G. L., Blanchet, F. G., Kindt, R., Legendre, P., et al. (2025). *vegan: Community ecology package* (Version 2.6-10) [R package]. Comprehensive R Archive Network (CRAN). <https://CRAN.R-project.org/package=vegan>
- Our World in Data. (2023). *Data on CO₂ and greenhouse gas emissions*. Retrieved June 7, 2024, from <https://github.com/owid/co2-data>
- Peterson, R. A. (2021). Finding optimal normalizing transformations via bestNormalize. *The R Journal*, 13(1), 310–329. <https://doi.org/10.32614/RJ-2021-041>
- Pfaff, B. (2024). *urca: Unit root and cointegration tests for time series data* (Version 1.3-4) [R package]. Comprehensive R Archive Network (CRAN). <https://CRAN.R-project.org/package=urca>
- Pierre, A. A., Akim, S. A., Semenyio, A. K., & Babiga, B. (2023). Peak electrical energy consumption prediction by ARIMA, LSTM, GRU, ARIMA–LSTM and ARIMA–GRU approaches. *Energies*, 16(12), 4739. <https://doi.org/10.3390/en16124739>
- R Core Team. (2023). *R: A language and environment for statistical computing* [Computer software]. R Foundation for Statistical Computing. <https://www.R-project.org/>
- Raubitzek, S., & Neubauer, T. (2021). Taming the chaos in neural network time series predictions. *Entropy*, 23(11), 1424. <https://doi.org/10.3390/e23111424>
- Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology*, 46, 561–584. <https://doi.org/10.1146/annurev.ps.46.020195.003021>
- Shahi, T. B., Shrestha, A., Neupane, A., & Guo, W. (2020). Stock price forecasting with deep learning: A comparative study. *Mathematics*, 8(9), 1441. <https://doi.org/10.3390/math8091441>
- Trivedi, D. V., & Patel, S. (2022). An analysis of GRU-LSTM hybrid deep learning models for stock price prediction. *International Journal of Scientific Research in Science, Engineering and Technology*, 9(3), 47–51. <https://doi.org/10.32628/IJSRSET229264>
- Widiasari, I. R., & Efendi, R. (2024). Utilizing LSTM-GRU for IoT-based water level prediction using multi-variable rainfall time series data. *Informatics*, 11(4), 73. <https://doi.org/10.3390/informatics11040073>
- Yang, C.-H., Chen, B.-H., Wu, C.-H., Chen, K.-C., & Chuang, L.-Y. (2022). Deep learning for forecasting electricity demand in Taiwan. *Mathematics*, 10(14), 2547. <https://doi.org/10.3390/math10142547>
- Yeo, I.-K., & Johnson, R. A. (2000). A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4), 954–959. <https://doi.org/10.1093/biomet/87.4.954>
- Yoo, T.-W., & Oh, I.-S. (2020). Time series forecasting of agricultural products' sales volumes based on seasonal long short-term memory. *Applied Sciences*, 10(22), 8169. <https://doi.org/10.3390/app10228169>