A Comparative Study of Genetic-Based Approaches for Enhanced Hourly Temperature Predictions

Fabrizio Sandri
University of Trento, Italy

***Abstract*—Nowadays, time series forecasting is widely applied across various domains. However, addressing the challenges of time series forecasting proves difficult due to factors that extend beyond the data. In this context, approaches based on deep learning and metaheuristics emerge to confront the complexities of time series forecasting by harnessing the inherent data. This study examines two strategies for predicting the outdoor temperature in the upcoming hour: a deep learning model, optimized through Genetic Algorithms, and an interpretable alternative employing Genetic Programming. The analysis aims to compare the results attained through these differing methodologies.**

***Index Terms*—Time Series Forecasting, Genetic Programming, Genetic Algorithms, LSTM.**

I. Introduction

TIME series forecasting finds its applications across diverse domains, ranging from economic contexts for predicting stock prices to medical scenarios where it aids in forecasting health conditions or diseases over time. Additionally, it plays a crucial role in the realm of weather and climate forecasting, which is the focal point of this study. Specifically, this research delves into utilizing historical data from a weather station to predict future meteorological conditions, particularly the forecast of outdoor weather temperature. The main challenge lies in the amount of factors influencing temperature changes, such as wind, ocean currents, humidity, and other environmental variables. Addressing this problem is challenging not only due to the individual factors but also because of the intricate relationships among them.

In this particular scenario, the adoption of deep learning is deemed appropriate due to its capacity to effectively capture and model intricate relationships inherent in the dataset. The research methodology involves implementing a well-established deep learning model based on recurrent neural networks, with the aim of optimizing its performance by identifying the most suitable hyperparameters through the use of a Genetic Algorithms[1] for hyperparameter tuning. One notable limitation of deep learning models is their black box nature, where they operate on complex mathematical models that are interpretable only by computers, not humans. Recognizing this lack of interpretability, this study explores an alternative model that can deliver similar results to complex deep neural networks while providing interpretability. This alternative allows for a better understanding of how outputs are derived from inputs, offering transparency in the forecasting process.

II. Problem Statement

Consider a weather time series dataset represented as $\{x_1, x_2, \ldots, x_N\}$, where N is the number of samples, and each x_i is a set of relevant features recorded at timestep i. The goal is to develop a model that can estimate the outdoor temperature of the next hour based on historical observations. This can be formulated as the task of learning a function f such that:

$$\hat{y}_{t+1} = f(x_t, x_{t-1}, \ldots, x_{t-p+1}) \tag{1}$$

where $\hat{y}_{t+1}$ is the predicted value of the temperature at time $t+1$, p is the number of prior samples used for prediction, and x_t is the feature vector at time t.

The objective is to find a model that minimizes the discrepancy between the predicted temperature value and the actual temperature. This discrepancy is quantified using the Mean Absolute Error:

$$\mathcal{L} = \frac{1}{N} \sum_{t=1}^{N} |y_t - \hat{y}_t| \tag{2}$$

III. Methodology

In this study, a baseline algorithm is presented for comparative purposes. Subsequently, two valid alternative approaches are thoroughly examined. The first approach involves employing a customized Genetic Algorithm for hyperparameter tuning of a LSTM model, while the second approach utilizes Genetic Programming to directly infer potential relationships among variables from the data.

A. Baseline

The most basic model involves predicting the temperature for the next hour using a simple approach: returning the last observed temperature. This straightforward model is encapsulated by the equation:

$$f(x_t, x_{t-1}, \ldots, x_{t-p+1}) = x_t \tag{3}$$

B. LSTM tuning using Genetic Algorithms

The intricate relationships concealed within time series data pose a challenge for conventional neural networks, which often find it difficult to effectively model such sequences. In response to this challenge, Recurrent Neural Networks(RNN) emerge as a valuable solution, enabling the processing of

sequential data and the integration of learned parameters from preceding time steps. Numerous studies have demonstrated the limitations of standard neural networks in handling weather-related time series data[2], emphasizing the superior performance of more advanced RNN-based models, such as Long Short-Term Memory(LSTM) networks.

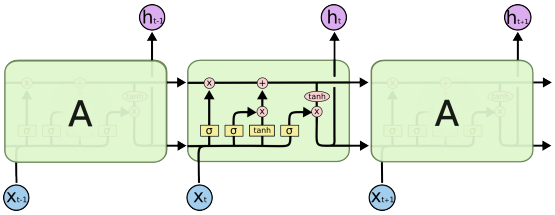

Fig. 1: The architecture of a LSTM.

Without entering into specific details, LSTM networks are explicitly crafted to capture and model long-term dependencies within sequences of information. The network incorporates loops, enabling the propagation and retention of information over time, essentially serving as a form of memory. In this specific context, a sequence refers to a subset of a weather time series with a length of p, denoted as $\{x_1, x_2, \ldots, x_p\}$, where $p < N$ represents the sequence length corresponding to the number of preceding weather samples used for prediction. At the end of the LSTM network, there is a linear layer that compresses the output into a single floating-point value, representing the ultimate temperature forecast. The entire architecture is trained via backpropagation, and the hyperparameters are detailed as follows:

- **Sequence length**: the length of the sequence corresponding to the number of prior weather samples used for prediction.
- **LSTM hidden size**: the number of hidden units or neurons in each LSTM layer, determining the capacity of the model to capture complex patterns.
- **LSTM number of layers**: the number of stacked LSTM layers.
- **Optimizer learning rate**: the step size at which the optimizer adjusts the model weights during training.
- **Extra parameters list**: additional parameters beyond the outdoor temperature included as inputs, providing supplementary information for the model. Each element in the input sequence fed to the LSTM can be a vector of multiple features. Therefore, this hyperparameter aims to identify the weather parameters that enable the model to achieve optimal accuracy.

To employ genetic algorithms, various aspects must be defined, including the representation of the genotype for each individual, the selection mechanisms and the methods for individual mutation and crossover.

1) Encoding of the Genotype: In this specific context of hyperparameter tuning, the genotype is represented as an array containing the aforementioned hyperparameters, maintaining the same order. The order within the array is significant, as both the mutation and crossover operators are specifically tailored for this specific order.

2) Crossover: Crossover plays a pivotal role in emphasizing exploitation by combining optimal parts from different individuals. In this context, a uniform crossover operator is employed, selecting elements from the first and second individuals with equal probability.

3) Mutation: The first three parameters of the genotype (sequence length, LSTM hidden size, and LSTM number of layers) are integers, while the remaining ones are a float (the learning rate) and an array (the set of extra parameters). Considering the distinct nature of each gene in the genotype, a custom mutation operator has been implemented. It introduces an integer/float perturbation following a normal distribution to the integer/float genes, respectively, with a magnitude proportional to a predefined standard deviation provided by the user. For the last element in the genotype, the set of extra parameters, one element is probabilistically removed and added based on a certain probability.

4) Selection: In the context of parent selection, tournament selection has been employed, wherein individuals are randomly sampled and compared against each other in tournaments of size K. The fittest individual from each tournament is then chosen as a parent for the genetic operations. Moreover, survivor selection follows an elitism strategy, aiming to preserve the best individuals from the previous generation.

5) Algorithm: The genetic algorithm can be outlined as follows:

1) Randomly sample an initial population of individuals.
2) Evaluate the fitness of all individuals in the population.
3) Select parents for reproduction through tournament selection(parent selection).
4) With a probability p_c, choose two individuals from the parents' pool and return a single individual by applying the crossover operator. The new individual is then reintroduced into the parent pool.
5) With a probability p_m, pick one individual from the parents' pool and subject it to mutation.
6) Select individuals for the next generation by employing elitism as the survivor selection method.
7) Repeat from step 2 until a termination condition is met, such as reaching a specified number of generations.

C. Genetic Programming

The prior approach, which uses deep learning to address the time series weather temperature forecasting problem, lacks a crucial aspect: interpretability. This lack push the exploration of a second methodology centered on Genetic Programming, with the primary goal of establishing an interpretable model for the problem at hand.

In Genetic Programming, individuals are characterized by tree-based structures that represent a program or equation. The objective is to evolve a computer program or equation capable of solving the problem by directly deducing relationships within the data, without any prior assumptions about

the model. However, a significant challenge encountered in Genetic Programming lies in the requirement for a large population of individuals. The dimensions of the evolved programs or equations can quickly increase, necessitating a substantial number of individuals within the population to adequately navigate and explore the search space.

In this study, the objective of Genetic Programming is to evolve an arithmetic expression that effectively models the intricate relationships among various weather parameters. This evolved tree-based structure representing an expression should have the capacity to predict the outdoor temperature for the subsequent hour, relying solely on prior observations.

The tree-based structure consist of nodes, where:

- The leaves of the tree derive values from the **terminal set** T, encompassing constants and all variables relevant to the problem. In the problem examined in this study, the terminal set is defined as $\{x_t, x_{t-1}, \ldots, x_{t-p+1}, e\}$, where p represents the number of preceding samples used for prediction, and e is an ephemeral constant randomly initialized.
- The inner nodes represent functions that establish relationships among nodes in the terminal set. Each function has an arity, indicating the number of arguments it accepts. In this instance, the **function set** is denoted as F and contains the four basic arithmetic operations and the trigonometric functions, $\sin$ and $\cos$. Due to the impracticality of a priori checking all potential inputs with Genetic Programming, the division operator has been redefined to prevent a zero division error.

Since the genotype is represented as a tree-based structure, standard mutation and crossover operations need to be tailored accordingly. In the case a one-point crossover is used, wherein a random crossover point is chosen in each individual, and the sub-trees rooted at the selected point are exchanged between the individuals. Regarding mutations, a customized mutation operator is employed, incorporating three mutation strategies randomly chosen with equal probability:

- **Shrink**: a random node in the tree is selected, and the entire sub-tree is replaced with a single node.
- **Increase**: this strategy inserts a new branch at a random position in the mutated individual.
- **Ephemeral Mutation**: this mutation alters one of the ephemeral variables by applying a random mutation.

The rationale behind this mutation operator is to mitigate the bloat problem. This approach gives trees the flexibility to grow or shrink with an equal probability, thus preventing excessive growth.

IV. Experiments

The experiments are carried out using weather data from the year 2022 obtained from my personal weather station in northern Italy[3]. The station records various weather parameters, including outdoor temperature, humidity, atmospheric pressure, wind speed, and dewpoint, at two-second intervals. Due to the substantial size of the dataset, approximately 5.5 GB, the data is resampled. Specifically, the temperature recordings are aggregated to an hourly frequency, guided by the assumption that temperature changes within a few minutes are not characterized by abrupt fluctuations. This process reduces the dataset to approximately 9000 samples.

The dataset is partitioned into three sets for training, validation, and testing purposes. During the hyperparameter tuning of the LSTM using Genetic Algorithms, the training set is employed for training the model through backpropagation by minimizing the error on the training set. The validation set is used to assess the model's generalization capabilities, and the accuracy metric based on the Mean Absolute Error is calculated on the validation set, serving as the fitness function for Genetic Algorithms. Finally, the test set is used to compare the three models.

For the method involving Genetic Programming, the size of the training set is reduced to the first 100 samples due to the impracticality of using the entire training set given the number of individuals in the population, which would require excessive time. The validation set is employed, similar to the other method, to measure the model's ability to generalize and select the model that exhibits better generalization without overfitting the training data.

A. Implementation

The implementation of both approaches discussed in this paper is implemented in Python and is publicly available on GitHub [4]. Specifically:

- The Genetic Algorithms employed for hyperparameter tuning was developed from scratch leveraging PyTorch[5] as the backend for the deep learning part. PyTorch provides advanced tools for both automatic gradient tracking and backpropagation, along with a simple interface for implementing LSTM models.
- For the Genetic Programming model, the DEAP[6] open-source library is utilized. It has been adapted to suit the specific requirements of this study. More precisely, the evolutionary algorithm governing genetic programming is modified to calculate the accuracy on the validation set at the conclusion of each generation. This change is crucial for evaluating the model's generalization capabilities.

V. Results

The performance of the proposed models has been assessed across various configurations to leverage their capabilities. The outcomes for each model are presented in the subsequent sections, categorized by model. The results for the baseline are reported in the concluding section dedicated to the comparison, as this model do not have specific configurations.

A. LSTM Model

This is the slowest algorithm, as the evaluation of each individual in the population involves training a LSTM model for a specified number of epochs—set to 15 in this instance. The initial population consists of 20 randomly initialized individuals, and the algorithm runs for 30 generations. The tournament size K is set to 3 and the number of elites preserved at each generation is set to 2. The crossover rate

p_c and mutation rate p_m are respectively set to $p_c = 0.3$ and $p_m = 0.7$. The entire experiment takes several minutes, and the fitness progress over the generations is depicted in Figure 2, where the plot illustrates the best model's fitness evaluated on the validation set at the end of each generation.

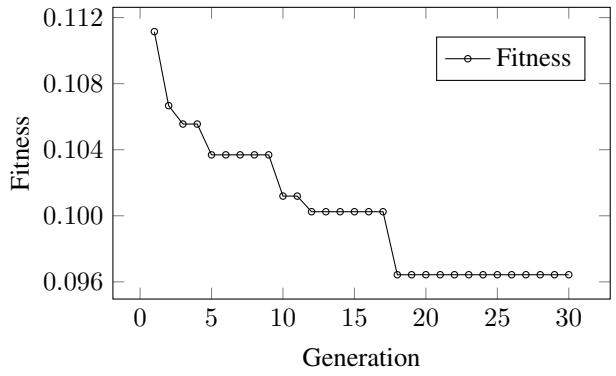

Fig. 2: Change in fitness over generations.

The best individual reported by Genetic Algorithms has a fitness of 0.0984 on the validation set, and the optimal configuration found is `[61, 33, 1, 0.0044261263, []]`. Interestingly, the model did not incorporate any additional weather parameters beyond the outdoor temperature, indicating that these omitted parameters may not have played a crucial role in the forecasting process.

B. Genetic Programming

In this particular case, the problem space is large, necessitating a substantial population. Specifically, the population size is configured with 500 individuals, and the tournament size K is set to 5. A range of values for the crossover rate p_c and mutation rate p_m has been explored, yielding notably diverse outcomes. The optimal tree is derived with $p_c = 0.8$ and $p_m = 0.1$, executing over a span of 20 generations. The visual representation of the evolved tree is provided in Figure 3, where $\{x_0, x_1, x_2, x_3\}$ denotes the past $p = 4$, with x_3 being the latest one, and $e_0 = 0.17488$ and $e_1 = 0.97058$ represent ephemeral variables. The fitness progress over the generations is depicted in Figure 4, where the plot illustrates the best model's fitness evaluated on the training set at the end of each generation as well as the average size of the evolved trees.

An interesting observation arises from the evolved tree structure: only the initial and final temperature observations are employed to forecast the temperature of the next hour while the intermediate measurements of the temperature are disregarded in this predictive process. This distinctive characteristic adds nuance to the understanding of how the Genetic Programming model interprets and utilizes past observations and ephemeral variables in forecasting.

C. Comparison

All the three models were tested on the test split of the dataset, containing the data of the last half of the year, obtaining the results reported in Table I. The results detailed in Table I, distinctly demonstrate the LSTM-based model's superior performance in terms of Mean Absolute Error. However, in terms of complexity, Genetic Programming produced an interpretable model for the given problem, characterized by basic arithmetic operations. In contrast, the LSTM model involved a more intricate structure with a total of 3501 parameters.

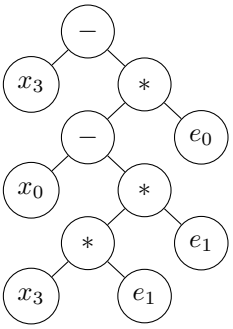

Fig. 3: Evolved tree generated by Genetic Programming.

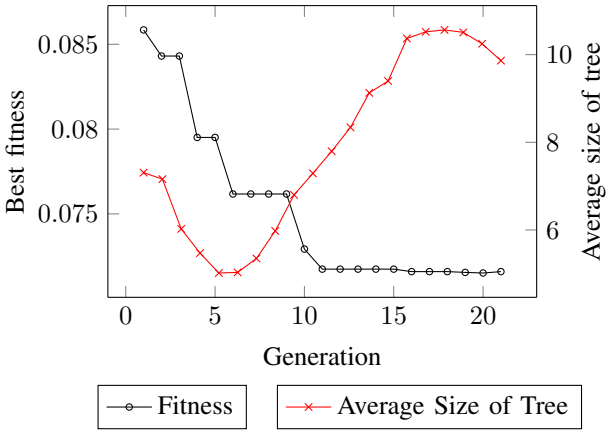

Fig. 4: Fitness trend and average tree size over generations.

	Baseline	LSTM	GP
MAE	0.1126	0.08005	0.0929

TABLE I: MAE values for different models

VI. Conclusion

In summary, the obtained results reveal that Genetic Programming produced satisfactory outcomes, albeit with a slightly inferior performance compared to LSTM in terms of MAE. Despite LSTM's superior performance, the evolved mathematical expression might be favored for its simplicity and interpretability, particularly in scenarios where computational cost and efficiency are fundamental. It is crucial to note that this study exclusively utilized outdoor temperature, neglecting other parameters. Acknowledging the potential existence of intrinsic relationships among various weather parameters, future research endeavors should encompass a more comprehensive exploration of the search space to unearth improved solutions by exploiting the interdependencies within the complete set of weather data.

REFERENCES

[1] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimed. Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021.

[2] J. M. Han, Y. Q. Ang, A. Malkawi, and H. W. Samuelson, "Using recurrent neural networks for localized weather prediction with combined use of public airport data and on-site measurements," *Building and Environment*, vol. 192, p. 107601, 2021.

[3] F. Sandri, "Trentino weather station dataset," https://github.com/FabrizioSandri/bio-inspired-ai-project/tree/main/data, 2022.

[4] F. Sandri, "Bio-inspired ai project," https://github.com/FabrizioSandri/bio-inspired-ai-project, 2023.

[5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[6] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, no. 1, p. 2171–2175, jul 2012.

APPENDIX A
QUALITATIVE RESULTS

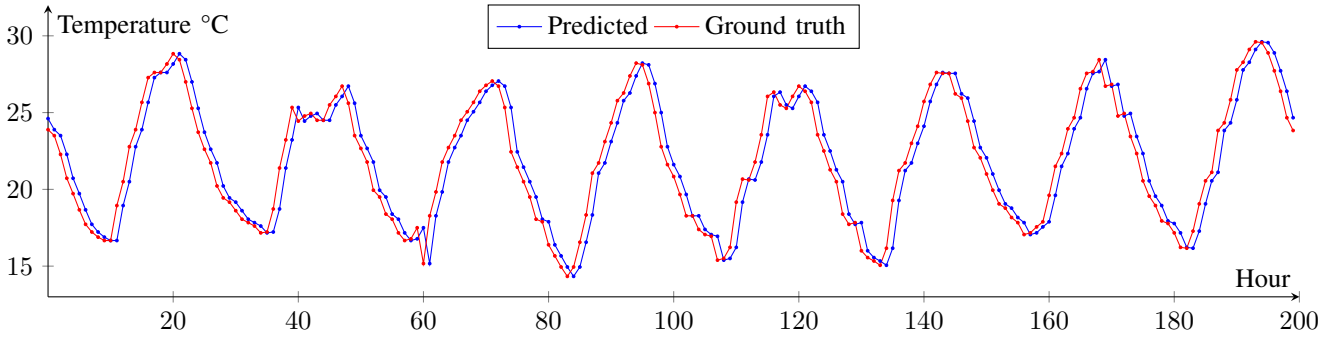

Fig. 5: **Baseline** model evaluated on the first 200 samples of the test set.

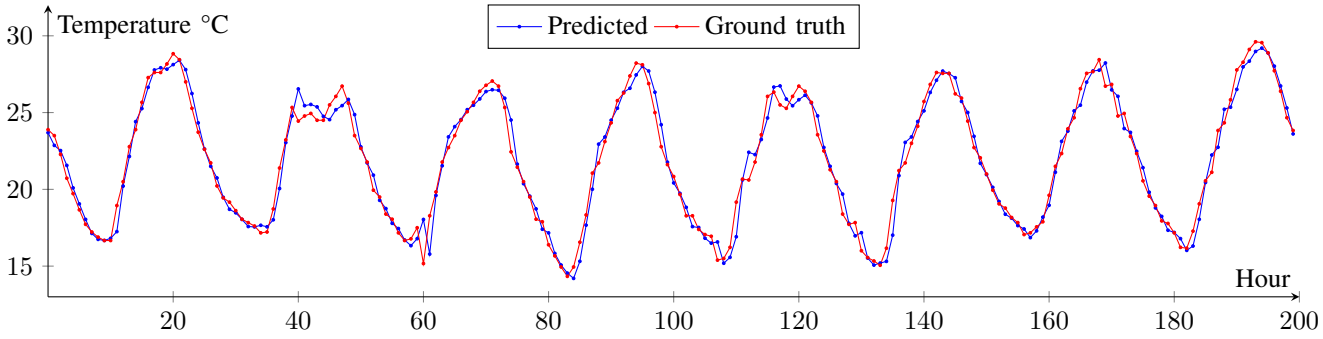

Fig. 6: **LSTM** model evaluated on the first 200 samples of the test set.

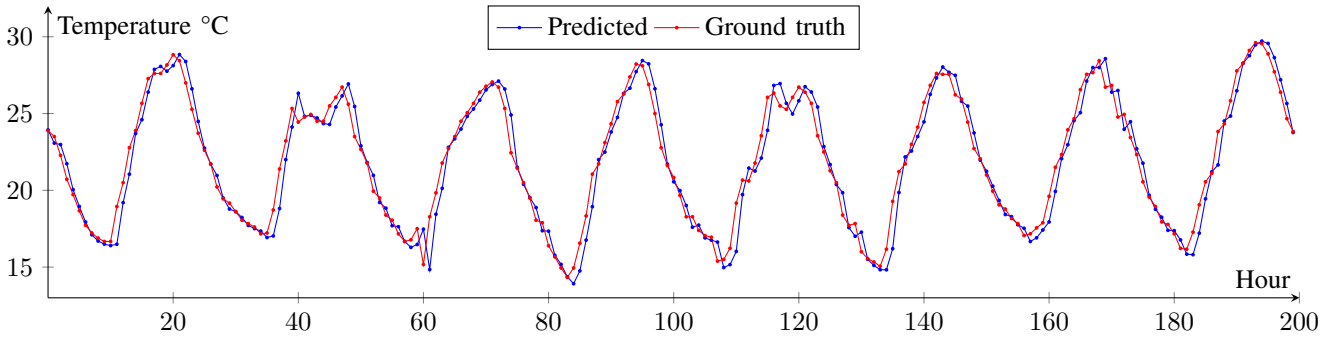

Fig. 7: **Genetic Programming** model evaluated on the first 200 samples of the test set.