Neuroevolutionary Transfer Learning for Time Series Forecasting

Aymeric Vellinger[2](✉), J. F. Torres[1], Federico Divina[1], and Wim Vanhoof[2]

[1] Pablo de Olavide University, Sevilla, Spain
{jftormal,fdivina}@upo.es

[2] University of Namur, Namur, Belgium
aymeric.vellinger@student.unamur.be, wim.vanhoof@unamur.be

Abstract. In this paper, we propose a neuroevolution technique specifically designed for evolving LSTM networks. The proposed technique uses a grammar-based approach to evolve LSTM neural networks for time series prediction tasks, and is based on a previous technique which was designed in order to evolve CNN networks.

We use transfer learning in order to reduce the computational time of our approach. We have compared results obtained with other state of the art time series forecasting techniques on twenty time series, which contains data generated by sensors placed on a number of Iberian pigs. Results obtained confirm the effectiveness of the strategy proposed in this work.

Overall, we showcase the potential of our proposal in producing precise and efficient deep learning models for time series prediction, as well as the adaptability of transfer learning to new datasets.

Keywords: Time series forecasting · Neuroevolution · Deep Learning

1 Introduction

Evolutionary algorithms (EAs), deep learning (DL), and Long Short-Term Memory networks (LSTM) have gained significant interest in recent years for data science applications, including time series prediction and animal movement tracking [8,9]. EAs utilize principles inspired by biological evolution, employing techniques like natural selection, mutation, and genetic recombination to optimize machine learning models such as deep learning models, e.g., [5].

DL is a machine learning technique that uses deep neural networks to model and analyze data. These neural networks can have multiple layers and are trained using large amounts of data to improve their accuracy in prediction. DL has been used in object identification in images [4,17], energy prediction [7] or fraud detection [6], among others. The computational cost of training in DL is typically high. For this reason transfer learning is gaining more and more popularity [13,18]. Transfer learning is a powerful machine learning technique that improves performance on a target task by leveraging knowledge learned from a source task.

In transfer learning, a pre-trained model is used as a starting point for a new task, rather than training a new model from scratch. This pre-trained model has typically been trained on a large, diverse dataset and has learned a set of general features that can be applied to many tasks.

Hyperparameter are parameters that are not learned directly from the data, but rather set by the researcher or programmer. Their optimization is a critical component of DL model development. In fact, such parameters affect the behavior of the model during training and can significantly impact the performance and generalization capability of the model. Therefore, selecting the optimal set of hyperparameters is crucial for achieving the best possible performance of the deep learning model. By optimizing such hyperparameters, DL models can achieve better accuracy and faster convergence, making them more suitable for real-world applications [10,14]. In time series prediction, LSTM networks have become a popular technique. The reason being that such networks are capable of modeling temporal dependence in data and learning complex patterns in time series [16].

The above observations motivates us to introduce a neuroevolutionary transfer techniques in order to optimize LSTM networks. Our proposal is based on Deep Evolutionary Network Structured Representation (DENSER) [3], which uses GAs and Dynamic Structured Grammatical Evolution in order to evolve DL networks. The original proposal is used to evolve CNN networks, while in this work we use a grammar for evolving LSTM networks. We use the neuroevolutionary technique to obtain an initial model for a domain, and then we fine tune the weights of the initial model on a target dataset.

In particular, we tested our proposal on a domain originating from the field of livestock farming, where it is highly interesting to monitor the condition of animals. For instance, it is common practice to utilize LSTM networks to analyze pig growth [15], predict the required amount of feed [1], or evaluate milk production in cows based on their diet [11]. In this article, we will explore the use of these techniques in predicting Iberian pig activity. Accurate activity prediction can be useful in order to assess the health status of the animals promptly, giving in this way the possibility to tackle possible health issues in early stages.

We compare the performance of our proposal with other state of the art prediction models.

The remainder of the article is structured as follows. Section 2 describes DENSER and outlines the main proposal of this article. Section 3 describes the dataset used, and report the experimentation performed. Finally, in Sect. 4 we draw the main conclusions and identify possible future developments.

2 Our Proposal

As mentioned in the previous section, in this paper we propose a neuroevolutionary transfer strategy for predicting livestock activity. Our proposal consists in two phases. In the first phase, we use a neuroevolution technique, called DENSER, in order to obtain an initial model, using some data from the domain. In a second phase, this initial model is fine tuned on the final target datasets.

$$\begin{aligned}
&< start >::=< lstm > \;\mid\; < lstm >< dropout >\\
&< lstm >::=layer : lstm[units, int, 1, 1, 20]\\
&\qquad\qquad < activation - function >< recurrent - function >\\
&\qquad\qquad < bias >< kernel - initializer >< recurrent - initializer >\\
&\qquad\qquad < forget - bias >\\
&< dropout >::=layer : dropout[rate, float, 1, 0, 0.7]
\end{aligned}$$

Fig. 1. Grammar used for the encoding of LSTM and Dropout layers

$$\begin{aligned}
< bias >::=&bias : True \mid bias : False\\
< activation - function >::=& act : sigmoid \mid act : relu \mid act : softmax\\
&\mid act : softplus \mid act : softsign \mid act : tanh\\
&\mid act : selu \mid act : elu\\
< recurrent - function >::=&rec - act : sigmoid \mid rec - act : relu\\
&\mid rec - act : softmax \mid rec - act : softplus\\
&\mid rec - act : softsign \mid rec - act : tanh\\
&\mid rec - act : selu\\
&\mid rec - act : elu
\end{aligned}$$

Fig. 2. Grammar used for the activation and recurrent functions.

Deep Evolutionary Network Structured Representation (DENSER) [3] is a deep learning architecture that uses evolutionary optimization techniques to automatically discover and learn the optimal neural network structure and parameters for a given task by combining the basic principles of GAs with Dynamic Structured Grammatical Evolution (DSGE) [2]. The sequence of layers is encoded using the GA, and the parameterisation of each layer using DSGE.

In DENSER, the grammar is used to define the structure of the evolved networks. The grammar specifies a set of production rules that describe how the network can be built.

A version of DENSER was used for evolving Convolutional Neural Networks. In our proposal, we use grammar designed in order to generate genotypes describing the topology and hyperparameters of LSTM networks. The grammar targets either a single-layered or stacked LSTM network, with potentials Dropout layer to prevent overfitting. A mandatory Dense layer is included to enable the network to learn complex patterns and relationships in the data.

The choice of minimum and maximum number of recursions in the grammar depends on the complexity of the task and the amount of available training data. Therefore, a parameter in the grammar stand for defining the minimum and maximum number of times a LSTM layer can be used in the neural network.

Figure 1 displays the start symbol, which is used as the recursive parameter for the architecture of the network, along with a grammar in Backus-Naur Form

$$\begin{aligned}
< forget-bias > &::= f-bias : True \mid f-bias : False \\
< kernel-initializer > &::= kernel-init : glorot-uniform \\
&\mid kernel-init : glorot-normal \\
&\mid kernel-init : he-normal \\
&\mid kernel-init : he-uniform \\
&\mid kernel-init : random-normal \\
&\mid kernel-init : random-uniform \\
< recurrent-initializer > &::= recu-init : glorot-uniform \\
&\mid recu-init : glorot-normal \\
&\mid recu-init : he-normal \\
&\mid recu-init : he-uniform \\
&\mid recu-init : random-normal \\
&\mid recu-init : random-uniform
\end{aligned}$$

Fig. 3. Grammar used for the initializers of LSTM layers.

$$\begin{aligned}
< learning > &::= < learner > [batchSize, int, 1, 16, 100] \\
< learner > &::= < gradient-descent > \mid < rmsprop > \mid < adam > \\
< gradient-descent > &::= learning : opt : gradient-descent[lr, float, 1, 0.0001, 0.1] \\
&[momentum, float, 1, 0.68, 0.99][decay, float, 1, 0.000001, 0.001] \\
&< nesterov > \\
< nesterov > &::= nesterov : True \mid nesterov : False \\
< adam > &::= learning : opt : adam[lr, float, 1, 0.0001, 0.1] \\
&[beta1, float, 1, 0.5, 1][beta2, float, 1, 0.5, 1] \\
&[decay, float, 1, 0.000001, 0.001] \\
< amsgrad > &::= amsgrad : True \mid amsgrad : False \\
< rmsprop > &::= learning : opt : rmsprop[lr, float, 1, 0.0001, 0.1] \\
&[rho, float, 1, 0.5, 1][decay, float, 1, 0.000001, 0.001]
\end{aligned}$$

Fig. 4. Grammar used for the optimizers

(BNF) form that outlines the hyperparameters and structure of a LSTM layer. Eventually, a Dropout layer has to be placed between LSTMs layers and at the end of the network before the Dense layer. The hyperparameters of such layer are also evolved.

Figure 2 shows the grammar used for the activation functions and the recurrent functions.

The proposed grammar also provides a comprehensive description of the forget bias hyperparameter (represented as a boolean), as well as the kernel and recurrent initializer. Figure 3 outlines the available initialization methods.

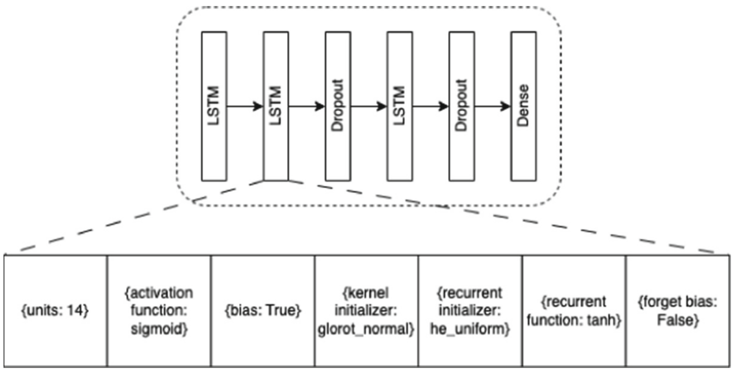

LSTM
LSTM
Dropout
LSTM
Dropout
Dense
{units: 14}
{activation function: sigmoid}
{bias: True}
{kernel initializer: glorot_normal}
{recurrent initializer: he_uniform}
{recurrent function: tanh}
{forget bias: False}

Fig. 5. Example of genotype generated by the grammar of a candidate solution with a number of recursions set to 3.

The optimizer and the batch size must also be specified, in order to allow the model to fit in a more efficient way. This can be done with the fragment of grammar shown in Fig. 4. Other parameters for the training could be added to the grammar, e.g, the number of epochs or the maximum time of computation per individual.

For the Dense layer, the number of units is fixed at the same size as the prediction horizon (24), as described in the data preparation part 3.

The grammar is then used to encode the genotype of each candidate solution with an ordered sequence of derivation steps in two different level; the first one for representing the macro structure of the networks and the second one to encodes the parameters associated to a layer, i.e., the hyperparmeters; an example of a candidate genotype is presented in Fig. 5.

GAs can then be applied to evolve the individuals of the population through variation operators. Each offspring is generated by applying the standard DENSER crossover and mutation operators on the selected parents. Each individual undergoes 500 epochs of training, utilizing the Mean Squared Error (MSE) (1) measurement as the loss function. Additionally, elitism has been applied to ensure that the best model is preserved in each generation. Mean Squared Error (MSE) is used as fitness function and to select the final result of this phase, which is the initial model that is further trained in the second phase, where the initial network is further trained for 500 epochs more on the target dataset.

3 Results

To assess the validity of our proposal, we have tested it on 20 time series originating from Rodríguez-Baena et al. [12]. Each time series was generated by triaxial accelerometer placed on twenty Iberian pigs.

In order to apply the technique proposed in this paper, each time series is converted into a matrix, the size of which depends on two parameters: the historical window (w) and the prediction horizon (h). w represents the number of prior entries that are utilized to train the algorithm and generate a model, which is then used to forecast the subsequent h values.

After having tested various values, in this work, h has been set to 24, corresponding to a period of 4 hours, while w has been set to 168. We use 30% of the data as test set, and the rest as training set.

In order to obtain the initial network, we have run the GA for 20 generations and with a population size of 20. We used the DENSER selection strategy, crossover probability is set to 0.1, while mutation probability to 0.3. Each individual is trained for 500 epochs. We have used the grammar detailed in Sect. 2, with a maximum recursion level of 2. At the end of the GA, the best individual, according to the MSE, is returned and used as initial model in the second phase, where it is further trained for 500 epochs on the final dataset. In order to obtain the initial model, we have use the data relative to animal 2. This choice was randomly made. The best architecture found is a two-layered stacked LSTM neural network. The first layer has 13 units and uses a sigmoid activation function with a hyperbolic tangent recurrent activation function. It uses He normal and Glorot uniform initializers for the kernel and recurrent weights, respectively, and the forget bias is set to false. The second layer is similar but includes a bias term initialized with random normal and random uniform initializers for the kernel and recurrent weights. The dropout rate of the Dropout layer is approximately 0.015039 as determined by the GA.

In order to evaluate the results obtained, we used the mean squared error (MSE) (1), mean absolute error (MAE) (2), mean absolute percentage error (MAPE)(3), and mean relative error (MRE) (4), which are defined as follows:

$$MSE : (y, \hat{y}) = \frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N} \tag{1}$$

$$MAPE : (y, \hat{y}) = \frac{100\%}{N} \sum_{i=0}^{N-1} \frac{y_i - \hat{y}_i}{y_i} \tag{3}$$

$$MAE : (y, \hat{y}) = \frac{\sum_{i=0}^{N-1} |y_i - \hat{y}_i|}{N} \tag{2}$$

$$MRE : (y, \hat{y}) = \frac{1}{N} \sum_{i=0}^{N-1} \frac{|y_i - \hat{y}_i|}{|y_i|} \tag{4}$$

Table 1 reports the results obtained on animal 2, which was used for obtaining the initial model, the best and worst results achieved (on animal 17 and animal 1, respectively), and the average results on the 20 time series considered. We can notice that the initial model transferred quite well on all the cases, and in

Table 1. Results obtained on test data for animal 2, best results obtained (animal 17), worst results obtained (animal 1), and average results.

	MAPE	MAE	MSE	MRE
animal 2	0.01205	0.01253	0.00029	1.20500
animal 17	0.00688	0.00712	0.00008	1.2552
animal 1	0.01680	0.01771	0.00054	1.68071
average	0.012544	0.01086	0.00025	1.05544

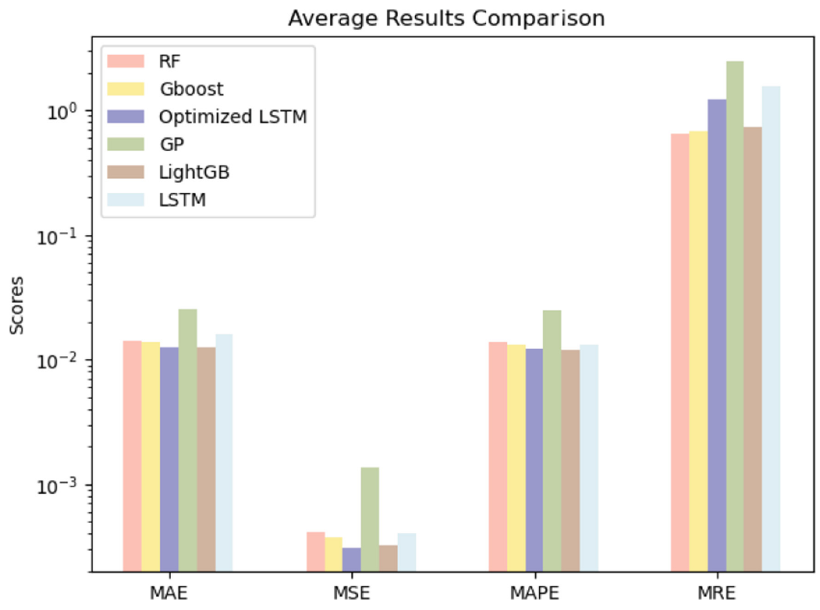


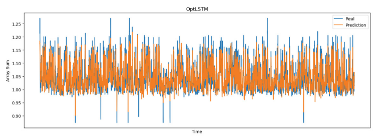
Fig. 6. Average results obtained on the 20 time series.

some cases, like on animal 17, it even achieve better results than for the animal on which the initial model was obtained.

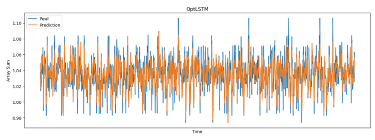
Upon examining the evaluations of the models trained with the data from the 20 different datasets, it can be concluded that the results obtained are quite satisfactory, as can be seen from the last row of the Table.

To assess model performance, we compared them with commonly used methods for time series prediction, including Gaussian Process, Gradient Boosting, Random Forest, LightGBM, and a manually configured LSTM with specific architecture and hyperparameters. The results, shown in Fig. 6, demonstrate that our optimized LSTM achieved the lowest Mean Squared Error (MSE) among the methods, indicating better approximation of true values and higher accuracy in predictions. The optimized model also performed well in terms of Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE). While it did not outperform other methods in terms of MRE, it still outperformed the manually configured LSTM, highlighting the significance of the neuroevolutionary phase.

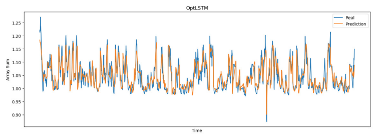
Finally, in Fig. 7 we show the best and worst predictions, obtained on animal 17 and 1, respectively, against the real values. We can see that in both cases the predictions are really close to the actual values, and it is interesting to notice that our proposal could handle the peaks of activity that are more evident in animal 17 then in animal 1.



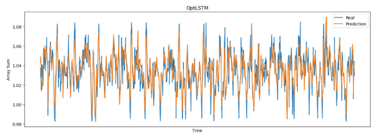
(a) All animal 1



(b) All animal 17



(c) Subset animal 1



(d) Subset animal 17

Fig. 7. Examples of best and worst predictions vs real values. Best predictions were achieved on animal 17, worst on animal 1.

4 Conclusions

In this paper, we have proposed a variant of DENSER with the aim of evolving LSTM networks. A transfer learning strategy has been used in order to increase the efficiency of our proposal.

The identification of topology and hyperparameters has yielded remarkable results in time series prediction, outperforming manually optimized neural networks, and transfer learning has shown great adaptability. These outcomes showcase the potential of the DENSER methodology in producing precise and efficient deep learning models for time series prediction.

It is worth noting that the DENSER methodology involves an iterative process of training and evaluating multiple neural networks, which can be computationally expensive. Additionally, LSTM models, in general, are known to have high time complexity due to their recurrent nature. As a result, training and testing these models can require significant computational resources, which can be a limitation for some applications. Despite the high computational cost, the remarkable outcomes attained through the DENSER methodology validate its use.

Future research can be directed towards exploring techniques such as threading to decrease the program's time complexity. Additionally, increasing the size of computations with more individuals and generations could lead to even better model performance. Furthermore, incorporating more detailed grammar could improve hyperparameter tuning. Lastly, smoothing the data could eliminate high and low spikes, making it more suitable for LSTM prediction.

Acknowledgments. The authors would like to thank the Spanish Ministry of Science and Innovation for the support under the project PID2020-117954RB-C21 and the European Regional Development Fund and Junta de Andalucía for projects PY20-00870 and UPO-138516.

References

1. Ahnaf, M.S., Kurniawati, A., Anggana, H.D.: Forecasting pet food item stock using arima and lstm. In: 2021 4th International Conference of Computer and Informatics Engineering (IC2IE), pp. 141–146 (2021)
2. Assunçao, F., Lourenço, N., Machado, P., Ribeiro, B.: Towards the evolution of multi-layered neural networks: a dynamic structured grammatical evolution approach. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 393–400 (2017)
3. Assunção, F., Lourenço, N., Machado, P., Ribeiro, B.: Denser: deep evolutionary network structured representation. Genet. Program Evolvable Mach. **20**, 5–35 (2019)
4. Degu, M.Z., Simegn, G.L.: Smartphone based detection and classification of poultry diseases from chicken fecal images using deep learning techniques. Smart Agricul. Technl. **4**, 100221 (2023)
5. Divina, F., Torres, J.F., García-Torres, M., Martínez-Álvarez, F., Troncoso, A.: Hybridizing deep learning and neuroevolution: application to the Spanish short-term electric energy consumption forecasting. Appl. Sci. **10**(16), 5487 (2020)
6. Habibpour, M., et al.: Uncertainty-aware credit card fraud detection using deep learning. Eng. Appl. Artif. Intell. **123**, 106248 (2023)
7. Hadjout, D., Torres, J.F., Troncoso, A., Sebaa, A., Martínez-Álvarez, F.: Electricity consumption forecasting based on ensemble deep learning with application to the Algerian market. Energy **243**, 123060 (2022)
8. Hu, H., Xia, X., Luo, Y., Zhang, C., Nazir, M.S., Peng, T.: Development and application of an evolutionary deep learning framework of LSTM based on improved grasshopper optimization algorithm for short-term load forecasting. J. Building Eng. **57**, 104975 (2022)
9. Martínez-Álvarez, F.: Coronavirus optimization algorithm: A bioinspired metaheuristic based on the Covid-19 propagation model. Big Data **8**(4), 308–322 (2020)
10. Morteza, A., Yahyaeian, A.A., Mirzaeibonehkhater, M., Sadeghi, S., Mohaimeni, A., Taheri, S.: Deep learning hyperparameter optimization: Application to electricity and heat demand prediction for buildings. Energy Buildings **289**, 113036 (2023)
11. Nguyen, Q.T., Fouchereau, R., Frénod, E., Gerard, C., Sincholle, V.: Comparison of forecast models of production of dairy cows combining animal and diet parameters. Comput. Electron. Agric. **170**, 105258 (2020)
12. Rodriguez-Baena, D.S., et al.: Identifying livestock behavior patterns based on accelerometer dataset. J. Comput. Sci. **41**, 101076 (2020)
13. Sarti, S., Laurenço, N., Adair, J., Machado, P., Ochoa, G.: Under the hood of transfer learning for deep neuroevolution. In: Applications of Evolutionary Computation: 26th European Conference, EvoApplications 2023, pp. 640–655. Springer (2023). https://doi.org/10.1007/978-3-031-30229-9_41
14. Shin, D., Ko, D., Han, J., Kam, T.: Evolutionary reinforcement learning for automated hyperparameter optimization in EEG classification. In: 2022 10th International Winter Conference on Brain-Computer Interface (BCI), pp. 1–5 (2022)
15. Taylor, C., Guy, J., Bacardit, J.: Prediction of growth in grower-finisher pigs using recurrent neural networks. Biosys. Eng. **220**, 114–134 (2022)
16. Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A.: Deep learning for time series forecasting: A survey. Big Data **9**(1), 3–21 (2021)

17. Wang, Y., Kang, X., He, Z., Feng, Y., Liu, G.: Accurate detection of dairy cow mastitis with deep learning technology: a new and comprehensive detection method based on infrared thermal images. Animal **16**(10), 100646 (2022)
18. Ye, R., Dai, Q.: Implementing transfer learning across different datasets for time series forecasting. Pattern Recogn. **109**, 107617 (2021)