On the benefits of automated tuning of hyper-parameters: an experiment related to temperature prediction on UAV computers

Renato de Sousa Maximiano[1], **Valdivino Alexandre de Santiago Júnior**[1], **Elcio Hideiti Shiguemori**[2]

[1]Instituto Nacional de Pesquisas Espaciais (INPE),
São José dos Campos, SP, 12227-010, Brazil

[2]Instituto de Estudos Avançados (IEAV), Trevo Coronel Aviador José Alberto Albano do Amarante 01 - Putim, SP, 12228-001, Brazil

{renato.maximiano,valdivino.santiago}@inpe.br, elcio@ieav.cta.br

Abstract. *Finding the best configuration of a neural network to solve a problem has been challenging given the numerous possibilities of values of the hyper-parameters. Thus, tuning of hyper-parameters is one important approach and researchers suggest doing this automatically. However, it is important to verify when it is suitable to perform automated tuning which is usually very costly financially and also in terms of hardware infrastructure. In this study, we analyze the advantages of using a hyper-parameter optimization framework as a way of optimizing the automated search for hyper-parameters of a neural network. To achieve this goal, we used data from an experiment related to temperature prediction of computers embedded in unmanned aerial vehicles (UAVs), and the models Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) to perform these predictions. In addition, we compare the hyper-parameter optimization framework to the hyper-parameter exhaustive search technique varying the size of the training dataset. Results of our experiment shows that designing a model using a hyper-parameter optimizer can be up to 36.02% better than using exhaustive search, in addition to achieving satisfactory results with a reduced dataset.*

1. Introduction

Unmanned aerial vehicles (UAVs) have been gaining space and notoriety in a significant range of applications, such as search/rescue [Castellano et al. 2020], public safety [Amato et al. 2020], construction/infrastructure [Munawar et al. 2021], and forest monitoring [Kinaneva et al. 2019]. Artificial intelligence (AI) techniques/algorithms, particularly deep neural networks (DNNs), and computer vision approaches have been embedded into computers of UAVs. Due to its computational power demands and also the time of use, these techniques can cause the computer on-board the UAV to overheat. This can compromise the mission, reduce the life of the computer, and even cause a burn on the board due to overheating [Benoit-Cattin et al. 2020, Prathaban et al. 2019, Bandopadhyay et al. 2022]. Therefore, performing computer temperature predictions while using AI/computer vision algorithms embedded into UAV computers is extremely important to avoid possible overheating during the flights.

Time series forecasting methods can be classified into two main categories: traditional statistical methods and methods based on machine learning (ML) [Elsworth and Güttel 2020]. Recent studies show significant results when using artificial neural networks for time series prediction, more specifically Long Short-Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] and Gated Recurrent Unit (GRU) [Chung et al. 2014] recurrent neural networks. Such neural networks have been successfully applied to several distinct domains such as financial market forecasting [Moghar and Hamiche 2020] [de Caux et al. 2020], weather forecasting [Tukymbekov et al. 2021] [Cañar et al. 2020], Covid-19 transmission [Patidar et al. 2021], prediction of failures in aeronautical components [Chui et al. 2021], among others.

Some issues exist when relying on artificial neural networks since they usually require a training phase in addition to being sensitive to the high number of hyper-parameters [Elsworth and Güttel 2020]. Thus, training and hyper-parameter tuning are in general considered the most critical aspects which are usually very costly financially and in terms of hardware infrastructure, demanding the use of graphical processing units (GPUs), specially if we consider deep neural networks (DNNs). Getting to the design of a model that achieves good results can be a difficult task with this high number of hyper-parameters and their values.

Recent studies have addressed the use of structures and techniques for automated hyper-parameter search optimization. In [Kong et al. 2017], the effect of the automatic adjustment of hyper-parameters for residential load prediction via deep learning (DL) is analyzed. In [Hamida et al. 2020], hyper-parameter optimization is used to improve the prediction of patients infected with Covid-19. In [Ekundayo 2020a], hyper-parameter optimization is used to design a prediction model for electricity consumption.

Since tuning of hyper-parameters is costly, specially if we rely on automated tuning via optimization frameworks, we must wonder whether it is truly useful to accomplish such a procedure because, eventually, the benefits might not be justified in terms of the performance of the models.

This study presents an evaluation aiming at perceiving the benefits of using an automated hyper-parameter optimization framework, Optuna [Akiba et al. 2019], in the design of LSTM and GRU models for temperature prediction of UAV computers when object detection models are running. Optuna looks for the optimal values of hyper-parameters via a Bayesian optimization algorithm. The tool relies on a historical record of results to determine which hyper-parameters to suggest on the next attempt, and then always estimates promising regions of hyper-parameters. Temperature data was collected from the Raspberry Pi 4's processor (usually found in UAVs), while running object detection algorithms. We analyzed eight You Only Look Once (YOLO) [Redmon et al. 2016, Bochkovskiy et al. 2020] and an R-CNN Mask [He et al. 2017]. The results obtained with Optuna are compared to an exhaustive search technique for the same number of iterations. In addition, we compared the influence of the size of the training dataset.

This article is organized as follows. Section 2 presents some relevant related studies. Section 3 presents the methodology used to carry out our experimentation process. Results and discussion are presented in Section 4. Section 5 presents conclusions and

future directions.

2. Related work

Although there are approaches that study the overheating of single-board computers, when they are running computer vision algorithms like [Benoit-Cattin et al. 2020], [Prathaban et al. 2019], and [Bandopadhyay et al. 2022] and approaches that study cooling techniques like [Machowski and Dzieńkowski 2021], [Joseph 2019], and [Manganiello 2021]. There is still a gap regarding studies that predict possible overheating.

According to [Kong et al. 2017] the common problem of DL, the selection of many hyper-parameters is rarely discussed, so the authors propose a residential load prediction framework based on LSTM networks with automatic hyper-parameter adjustment, where for this approach a improvement of about 10% when compared to random searches and exhaustive searches. [Hamida et al. 2020] also confirms that the selection of appropriate hyper-parameters for classification or prediction is a difficult and fundamental task to improve results in the design of models. In [Turner et al. 2021] the authors present the results and insights of the Black Box Optimization Challenge (BBO) in NeurIPS 2020, a challenge that emphasized the importance of evaluating derivatives-free optimizers to adjust the hyper-parameters of machine learning models. The challenge decisively demonstrated the benefits of Bayesian optimization over random search where top submissions showed sample efficiency gains of over 100 times compared to random search. The same effect is seen in [Parsa et al. 2019], where it is shown how Bayesian-based hyper-parameter optimization can take advantage compared to other simpler techniques, not only regarding the result but also the search time for the best set of hyper-parameters.

In several articles involving time series forecasting, the use of Optuna has been used as an automatic hyper-parameter optimization framework. In [Ekundayo 2020b] Optuna is used to optimize a CNN-LSTM model for predicting residential energy consumption, in [Nishitsuji and Nasseri 2022] it is used to optimize forgetting gates for lithofacies prediction, [Zegarra et al. 2021] uses Optuna to optimize LSTM architectures for tool wear estimation, and in [dos Santos Antoniassi 2022] it is used to optimize GRU models for river level prediction in the hydrographic region.

Our study differs from these others because it applies the hyper-parameter optimization framework in an approach still little explored, through an experiment related to temperature prediction in computers embedded in UAVs, comparing the use of Optuna with an exhaustive search. In addition, we analyze the impact that increasing the training dataset has on this approach.

3. Methodology

The methodology applied in this work is divided into 5 main steps, as show in Figure 1: i) Data collection; ii) organization of the training datasets; iii) model training using an exhaustive search of hyper-parameters; iv) model training using the Optuna automated hyper-parameter optimization framework; and v) test of the best models. We detail each step of the methodology in the subsections below.

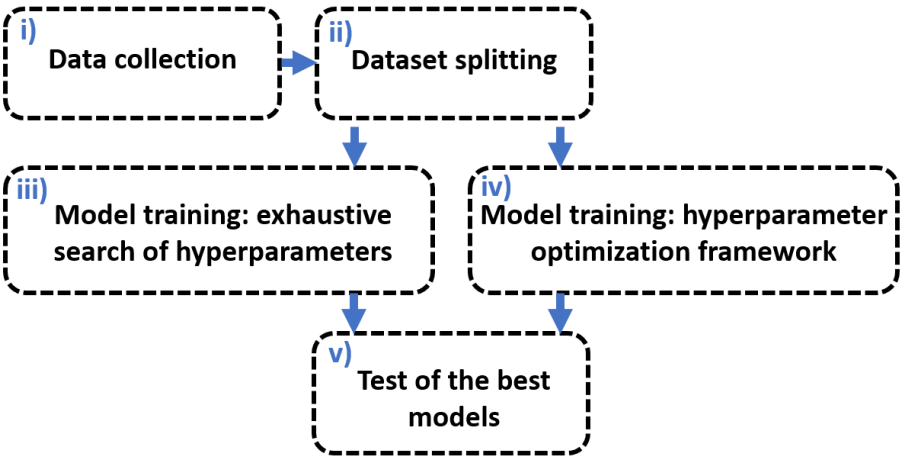

Figure 1. Methodology steps

3.1. Data collection

Processor temperature data were collected from simulations of the use of the Raspberry Pi 4 [Raspberry-Pi 2022], running object detection algorithms. These simulations were performed based on the flight characteristics of the Phantom 4, a commercial UAV developed by Dà-Jiang Innovations Science and Technology (DJI) [Dji 2022], which has an opeational autonomy of 30 minutes.

The object detection algorithms used for simulation are pre-trained networks with the COCO dataset [cocodataset 2022]. For this work we used 9 different models embedded into the RaspberryPi 4. The models used were: i) YOLO family [AlexeyAB 2022], with different input sizes at activation (YOLOv3 320x320, YOLOv3 416x416; YOLOv4 320x320, YOLOv4 416x416, YOLOv4 512x512 ; YOLOv4 -tiny 320x320, YOLOv4 -tiny 416x416 YOLOv4 - tiny 512x512); ii) Mask-R-CNN [escoladeestudantes 2022].

Altogether, 10 $\times$ 30-minute simulations were performed for each selected algorithm configuration. Therefore, in total, 90 tables were extracted with 1800 points of the processor temperature, with a time step equal to 1 second, totaling 162 thousand points collected.

3.2. Dataset splitting

After data acquisition, we split the data as follows: 80% for the training dataset and 20% for the test dataset. The training set was further divided into four subsets: i) Taking 25% of the training dataset; ii) 50% of the training dataset; iii) 75% of training dataset; and iv) 100% of training data. This latter subdivision was necessary to support our goal to realize about the benefits of the hyper-parameter optimization framework when facing different sizes of the dataset.

3.3. Model training: exhaustive search of hyper-parameters

In this step we performed an exhaustive search (Grid search) of some hyper-parameters for LSTM and GRU models. Altogether, 108 LSTM models and 108 GRU models were trained for each subset of the training data (25%, 50%, 75% and 100%), totaling 864 models, with the objective of predicting the Raspberry Pi processor temperature, while

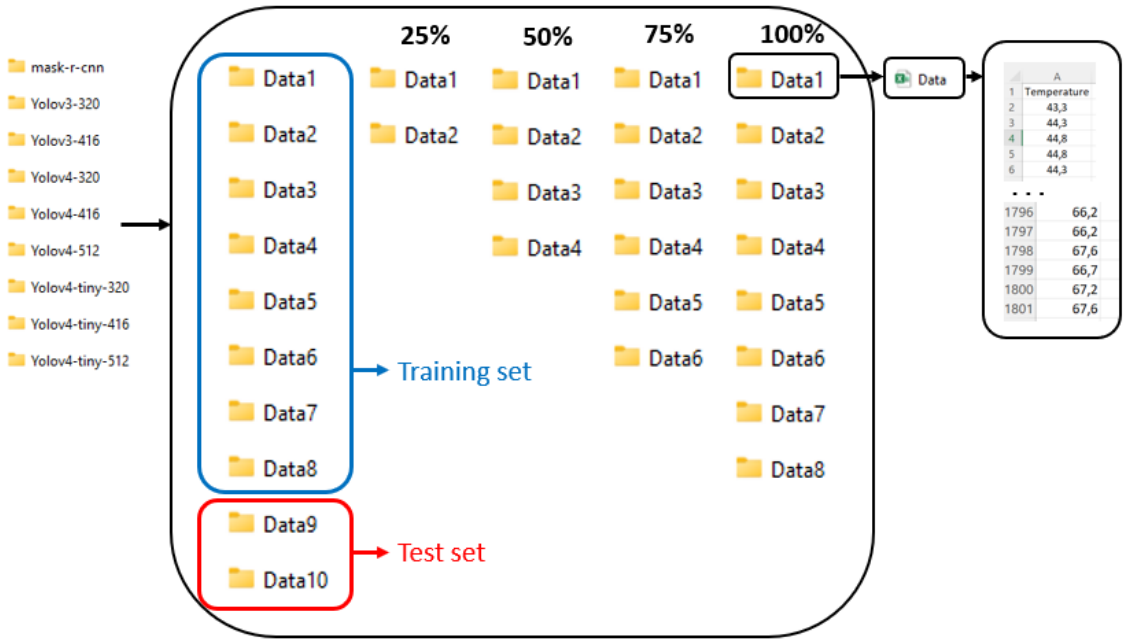

Figure 2. Organization of training datasets

object detection algorithms were running. Notice that for each subset of training data, 20% of the data was randomly separated for validation during training. For this step, the number of layers of the network is changed in 1, 3 or 5 layers, and the number of neurons of layers 1, 2 and 3 when demanded, a dropout layer is also changed after the first layer of the network, in addition to the windows of time that are taken into account before the forecast. Fixed and changed hyper-parameters are highlighted below.

As for the models it was fixed:

- Optimizer: Adam;
- Activation function: Hyperbolic tangent;
- Recurrence function: Sigmoid;
- Loss: Mean squared error (MSE);
- Epochs: 100, Patience: 10;
- Neurons layer 4 : 32;
- Neurons layer 5 : 32;
- Batch size: 1024;
- Forecast points: 15.

As for the models it was varied:

- Number of layers: {1,3,5};
- Neurons layer 1 : {32,64,128};
- Neurons layer 2 : {64,128};
- Neurons layer 3 : {64,32};
- Dropout: {0, 0.3};
- Points before: {15, 30}.

3.4. Model training: hyper-parameter optimization framework

In this step, we also considered 108 LSTM models and 108 GRU models trained for each subset of the training data (25%, 50%, 75% and 100%). But instead of performing an exhaustive search among the previously highlighted hyper-parameters, we used Optuna as a hyper-parameter optimization framework. By not needing to go through all the variables, in this step we increase the variation range of the hyper-parameters, where the number of layers is varied from 1 to 5 layers, as well as the number of neurons in layers 1, 2, 3, 4 and 5. Note which here is varied through ranges instead of working with categorical values.Fixed and changed hyper-parameters are highlighted below.

As for the models it was fixed:

- Optimizer: Adam;
- Activation function: Hyperbolic tangent;
- Recurrence function: Sigmoid;
- Loss: MSE;
- Epochs: 100, Patience: 10;
- Batch size: 1024;
- Forecast points: 15.

As for the models it was varied:

- Number of layers: Range{1 to 5};
- Neurons layer 1 : Range{32 to 128};
- Neurons layer 2 : Range{32 to 128};
- Neurons layer 3 : Range{32 to 128};
- Neurons layer 4 : Range{32 to 128};
- Neurons layer 5 : Range{32 to 128};
- Dropout: Range{0 to 0.3};
- Points before: Range{15 to 30}.

3.5. Test of the best models

In the last step, we separated the best models from each training set, hyper-parameter search technique, and network type (LSTM or GRU), based on the results obtained in the validation during training, where we used as a metric the MSE described in:

$$MSE = \frac{1}{n}\sum_{t=1}^{n} e_t^2 \tag{1}$$

where n stands for the number of elements, $\sum$ notation for sum, and e_t^2 the square of the difference between the actual value and the predicted value. We then got 16 different models to perform the activation with the test data.

4. Results and discussion

Table 1 shows the results of our experiment where LB means the input window size, Cn means the number of neurons per layer, and D means the dropout. In blue bold we show the top approach and in red the worst.

Table 1. Results of the experiment. Using the test subset data.

Model	Dataset	LB	C1	D	C2	C3	C4	C5	MSE
LSTM-Grid	100%	30	128	0.3	64	64	32	32	2.6767
LSTM-Grid	75%	30	128	0.3	64	64	32	32	2.7703
LSTM-Grid	50%	30	128	0.3	64	32	32	X	3.0777
LSTM-Grid	25%	30	128	0.3	64	64	32	32	6.9458
LSTM-Optuna	100%	25	83	0.1733	105	86	39	39	**2.6482**
LSTM-Optuna	75%	25	82	0.1503	42	47	84	82	2.6579
LSTM-Optuna	50%	29	32	0.1385	41	93	59	X	2.6651
LSTM-Optuna	25%	27	36	0.1719	42	98	81	87	4.4498
GRU-Grid	100%	30	128	0.3	64	64	32	32	2.6851
GRU-Grid	75%	30	128	0.3	64	32	X	X	2.7652
GRU-Grid	50%	30	128	0.3	64	64	32	32	2.9768
GRU-Grid	25%	30	64	0.3	64	64	32	32	3.6939
GRU-Optuna	100%	29	114	0.2331	115	89	40	89	2.6495
GRU-Optuna	75%	30	67	0.0796	66	X	X	X	2.8317
GRU-Optuna	50%	28	66	0.1863	98	39	X	X	2.8583
GRU-Optuna	25%	30	102	0.1929	30	95	X	X	3.0865

We can see that the LSTM model presents contrary performances. It got the best (lower) MSE with 100% of the training dataset with the Optuna framework, but it also delivered the worst outcome with the exhaustive search and 25% of the training dataset. Moreover, the GRU-Optuna configuration with 100% of the training set got the second best performance, quite close to the top approach.

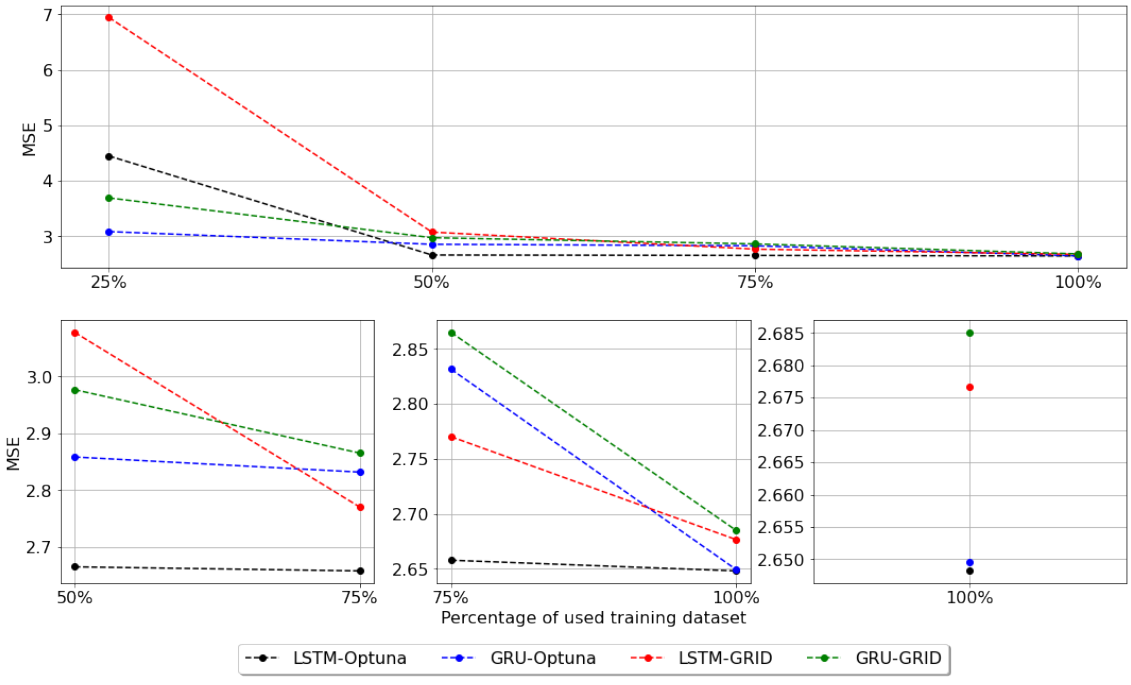

Figure 3. Results based on the size of training dataset

Figure 3 shows the MSE obtained by each model where on the x axis we have

the amount of data used for training. Analyzing Table 1 and Figure 3, we noticed that when using Optuna, as an automated hyper-parameter optimization structure, we got better results than performing an exhaustive search. With 25% of the training dataset, GRU-Optuna was the best approach followed GRU-Grid. Closer to 50% of the complete size of the training dataset, LSTM-Optuna surpassed all the other approaches and remained as the top solution up to 100% of the training set.

As for this study, we also notice that the GRU Network achieved better results with a small amount of data, when compared to the LSTM. On the other hand, LSTM had an advantage over GRU when the training dataset was increased. This conclusion is interesting to suggest practitioners to rely on GRU if their amount of data in the training dataset is not considerable large and LSTM otherwise.

In Figure 4, we can observe the decay of the error difference between the exhaustive search of hyper-parameters and the Optuna, when increasing the data set.

We compared the two hyper-parameter search techniques for each of the networks (LSTM and GRU) separately, and for each subset of training data. When using the hyper-parameter optimizer, for the set of 25% of the data for training, we have an improvement of 36.02% for LSTM models and 16.64% for GRU models. Using 50% of the data, we observed a difference of 13.4% for LSTM and 3.98% for GRU. For 75% we have a difference of 4.05% for LSTM and 1.16% for GRU. As for 100% of the data, we observed a difference of 1.06% for LSTM and 1.32% for GRU.

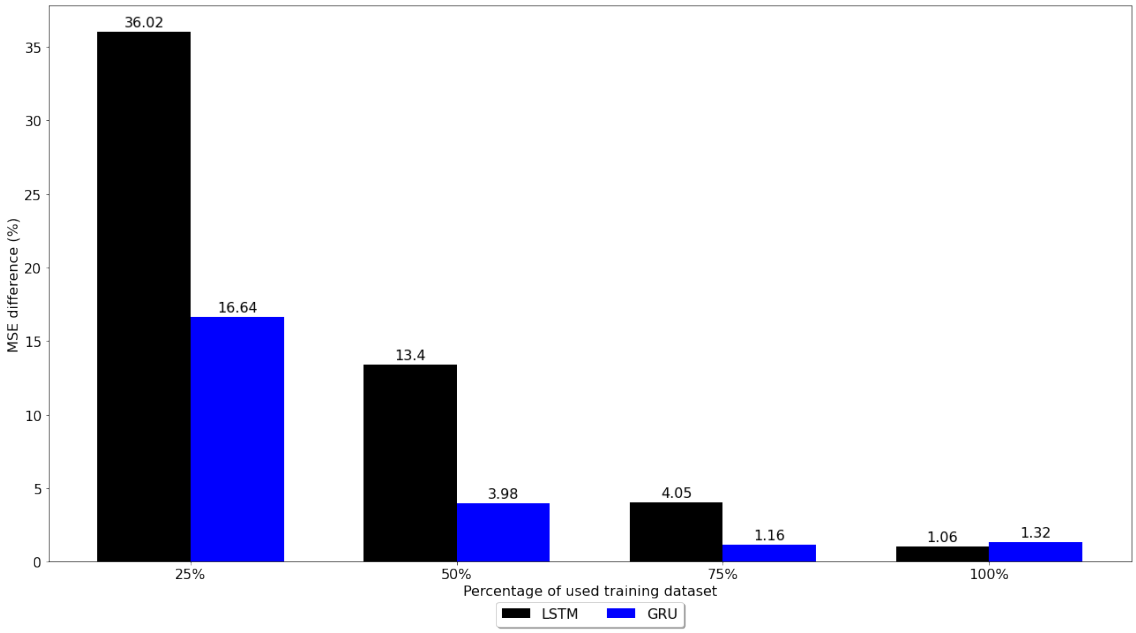

Figure 4. Error difference between the exhaustive search of hyper-parameters and Optuna

We also compared the average training time for the LSTM and GRU models considering the different subsets of training data. The models were trained using one Bull Sequana X1120 node of the SDumont supercomputer (SDumont) [lncc 2022] which has 4 $\times$ NVIDIA Volta V100 GPUs. We used 4 GPUs of this node. The results can be seen in

Table 2 where $\overline{LSTM}$ and $\overline{GRU}$ means the average of the average times for LSTM and GRU, respectively.

Table 2. Mean training time for the LSTM and GRU models

Model	Dataset	Time(s)
LSTM	100%	681.16
LSTM	75%	579
LSTM	50%	435.73
LSTM	25%	254.91
$\overline{LTSM}$	-	487.7
GRU	100%	595.06
GRU	75%	514.07
GRU	50%	396.25
GRU	25%	226.8
$\overline{GRU}$	-	433.04

We observe that the average training speed of the models increases significantly when decreasing the size of the training dataset. Moreover, the GRU network has a better training time than LSTM according to this result.

For a reduced dataset, the GRU-type networks stood out in relation to the LSTM, however, by the LSTM-type networks with only 50% of the training dataset achieved great performance, being similar when using 100% of the data, but with a much higher training speed, taking an average of 435.73 seconds to train a model against 595.06 seconds for the GRU using 100% of the training dataset. It is then concluded that it is possible through the use of a hyper-parameter optimization framework to improve the results for predictions of this work in addition to reducing the average training time of the models when linked to a reduction of the data set.

To sum up, we firstly should recognize the importance of the tuning of hyper-parameters. Notice that even the exhaustive search is still a tuning process even if not automated.

Secondly, an automated hyper-parameter optimization framework like Optuna is valuable. We believe that if there is an available and adequate computing infrascture (with GPUs) it is worth the effort. Notice that the worst mean time to train the models took less than 12 minutes to finish (see LSTM with 100% of the training set in Table 2).

5. Conclusions

This study showed the benefits of using a hyper-parameter optimization framework in an automated way, in the design of models for predicting the temperature of computers embedded in UAVs. We were also able to analyze the influence of the size of the training dataset on the results, in addition to comparing LSTM and GRU neural networks for this type of prediction.

The use of a hyper-parameter optimization framework proved to be more efficient than performing an exhaustive search for the same number of iterations. We showed that by reducing the training dataset, the difference between automated hyper-parameter tuning and performing an exhaustive search increases significantly.

We observed that for a reduced dataset, the GRU networks managed to achieve a smaller error when compared to the LSTM type. However, when increasing the dataset for training the model, the LSTM type networks stood out, achieving excellent results when linked to automated hyper-parameter search, getting an MSE of 2.6651 for the 50% subset of the training data, 2.6579 for 75% of the training data, and 2.6482 for 100% of the training dataset.

As future directions, we intend to compare other types of time series forecasting techniques, as well as different problems and databases. Hence, we will include more hyper-parameter ranges and perform a search in a larger number of iterations.

Acknowledgments

This research was developed within the **IDeepS** project (https://github.com/vsantjr/IDeepS) which is supported by the *Laboratório Nacional de Computação Científica* (LNCC/MCTI, Brazil) via resources of the SDumont supercomputer. This research was also supported by CAPES, grant #88887.610576/2021-00.

References

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

AlexeyAB (2022). `https://github.com/AlexeyAB/darknet`. [Last accessed: 4 JUN 2022].

Amato, G., Falchi, F., Gennaro, C., Massoli, F. V., and Vairo, C. (2020). Multi-resolution face recognition with drones. In *2020 3rd International Conference on Sensors, Signal and Image Processing*, pages 13–18.

Bandopadhyay, D., Jha, V., Bandyopadhyay, A., Roy, P., Halder, R., and Majhi, S. (2022). Automated people monitoring system using opencv and raspberry pi. In *ICT Analysis and Applications*, pages 905–913. Springer.

Benoit-Cattin, T., Velasco-Montero, D., and Fernández-Berni, J. (2020). Impact of thermal throttling on long-term visual inference in a cpu-based edge device. *Electronics*, 9(12):2106.

Cañar, R. L., Fontaine, A., Morillo, P. L., and El Yacoubi, S. (2020). Deep learning to implement a statistical weather forecast for the andean city of quito. In *2020 IEEE ANDESCON*, pages 1–6. IEEE.

Castellano, G., Castiello, C., Mencar, C., and Vessio, G. (2020). Preliminary evaluation of tinyyolo on a new dataset for search-and-rescue with drones. In *2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, pages 163–166. IEEE.

Chui, K. T., Gupta, B. B., and Vasant, P. (2021). A genetic algorithm optimized rnn-lstm model for remaining useful life prediction of turbofan engine. *Electronics*, 10(3):285.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

cocodataset (2022). https://cocodataset.org/#home. [Last accessed: 08 FEB 2022].

de Caux, M., Bernardini, F., and Viterbo, J. (2020). Short-term forecasting in bitcoin time series using lstm and gru rnns. In *Anais do VIII Symposium on Knowledge Discovery, Mining and Learning*, pages 97–104. SBC.

Dji (2022). https://www.dji.com/br. [Last accessed: 25 FEB 2022].

dos Santos Antoniassi, R. A. (2022). Predição de nível de rios da região hidrográfica do rio paraguai utilizando algoritmos de aprendizado de máquina.

Ekundayo, I. (2020a). *Optuna optimization based cnn-lstm model for predicting electric power consumption*. PhD thesis, Dublin, National College of Ireland.

Ekundayo, I. (2020b). *Optuna optimization based cnn-lstm model for predicting electric power consumption*. PhD thesis, Dublin, National College of Ireland.

Elsworth, S. and Güttel, S. (2020). Time series forecasting using lstm networks: A symbolic approach. *arXiv preprint arXiv:2003.05672*.

escoladeestudantes (2022). https://github.com/escoladeestudantes/opencv/tree/main/22_ObjectDetection_Mask-RCNN_Inception_v2_COCO. [Last accessed: 4 JUN 2022].

Hamida, S., El Gannour, O., Cherradi, B., Ouajji, H., and Raihani, A. (2020). Optimization of machine learning algorithms hyper-parameters for improving the prediction of patients infected with covid-19. In *2020 ieee 2nd international conference on electronics, control, optimization and computer science (icecocs)*, pages 1–6. IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Joseph, F. J. J. (2019). Iot based weather monitoring system for effective analytics. *International Journal of Engineering and Advanced Technology*, 8(4):311–315.

Kinaneva, D., Hristov, G., Raychev, J., and Zahariev, P. (2019). Early forest fire detection using drones and artificial intelligence. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1060–1065. IEEE.

Kong, W., Dong, Z. Y., Luo, F., Meng, K., Zhang, W., Wang, F., and Zhao, X. (2017). Effect of automatic hyperparameter tuning for residential load forecasting via deep learning. In *2017 australasian universities power engineering conference (aupec)*, pages 1–6. IEEE.

lncc (2022). https://sdumont.lncc.br/. [Last accessed: 4 JUN 2022].

Machowski, J. and Dzieńkowski, M. (2021). Selection of the type of cooling for an overclocked raspberry pi 4b minicomputer processor operating at maximum load conditions. *Journal of Computer Sciences Institute*, 18:55–60.

Manganiello, F. (2021). Computer vision on raspberry pi. In *Computer Vision with Maker Tech*, pages 159–225. Springer.

Moghar, A. and Hamiche, M. (2020). Stock market prediction using lstm recurrent neural network. *Procedia Computer Science*, 170:1168–1173.

Munawar, H. S., Ullah, F., Heravi, A., Thaheem, M. J., and Maqsoom, A. (2021). Inspecting buildings using drones and computer vision: A machine learning approach to detect cracks and damages. *Drones*, 6(1):5.

Nishitsuji, Y. and Nasseri, J. (2022). Lstm with forget gates optimized by optuna for lithofacies prediction.

Parsa, M., Mitchell, J. P., Schuman, C. D., Patton, R. M., Potok, T. E., and Roy, K. (2019). Bayesian-based hyperparameter optimization for spiking neuromorphic systems. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 4472–4478. IEEE.

Patidar, S., Jindal, R., and Kumar, N. (2021). Streamed covid-19 data analysis using lstm—a deep learning technique. In *Soft Computing for Problem Solving*, pages 493–504. Springer.

Prathaban, T., Thean, W., and Sazali, M. I. S. M. (2019). A vision-based home security system using opencv on raspberry pi 3. In *AIP Conference Proceedings*, volume 2173, page 020013. AIP Publishing LLC.

Raspberry-Pi (2022). `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/`. [Last accessed: 25 FEB 2022].

Tukymbekov, D., Saymbetov, A., Nurgaliyev, M., Kuttybay, N., Dosymbetova, G., and Svanbayev, Y. (2021). Intelligent autonomous street lighting system based on weather forecast using lstm. *Energy*, 231:120902.

Turner, R., Eriksson, D., McCourt, M., Kiili, J., Laaksonen, E., Xu, Z., and Guyon, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS 2020 Competition and Demonstration Track*, pages 3–26. PMLR.

Zegarra, F. C., Vargas-Machuca, J., and Coronado, A. M. (2021). Comparison of cnn and cnn-lstm architectures for tool wear estimation. In *2021 IEEE Engineering International Research Conference (EIRCON)*, pages 1–4. IEEE.