Meta-Learning for hyperparameters tuning in CNNs for Chest Images

Jesús García-Ramírez[1][0000−0002−1583−7554], Rodrigo Ramos Díaz[1][0009−0008−4368−6731], Jimena Olveres[1,2][0000−0002−1514−4520], and Boris Escalante-Ramírez(✉)[1,2][0000−0003−4936−8714]

[1] Facultad de Ingeniería,
Universidad Nacional Autónoma de México, Mexico City, Mexico
[2] Centro de Estudios en Computación Avanzada,
Universidad Nacional Autónoma de México, Mexico City, Mexico
jesus-garcia@cecav.unam.mx, ruyrdiaz@comunidad.unam.mx,
jolveres@cecav.unam.mx, boris@cecav.unam.mx

Abstract. Hyperparameter tuning is a time-consuming task for deep learning models. Meta-learning offers a promising solution to reduce the time required for this task. In this work, we propose a meta-learning approach to simulate a set of experiments and select a hyperparameter configuration (HC) that achieves high accuracy using a deep model. Our formulation involves conducting a grid search over hyperparameters to train a convolutional neural network and get an overview of their space. Then, a meta-regressor was trained using the experiment data to predict accuracy as a function of hyperparameter sets. Subsequently, the trained meta-regressor was employed to simulate diverse HCs and assess the corresponding deep model performance. Our approach was tested across two different domains: COVID-19 detection using X-ray images, and lung detection in computer tomography volumes. Furthermore, we evaluated the proposed approach with two different architectures. Our results show that the proposed method can simulate a set of experiments using the meta-regressor, saving time and computing resources during hyperparameter tuning.

Keywords: Deep learning · Meta-Learning · Simulation · Hyperparameter tuning

1 Introduction

Long training times in deep models are an important limitation especially in the absence of specialized hardware. Ceron and Castro [3], mention an example of the computational cost of training the Rainbow agent [9] with a powerful Graphic Processing Unit (NVIDIA Tesla P100). Each Atari game takes 5 days to train with the mentioned hardware and it is necessary to report the performance within confidence bounds. Consequently, an average of five independent runs is reported. They provide empirical evidence that Rainbow requires approximately 1,425 days without taking into account hyperparameter tuning.

Meta-learning is the process of distilling the experience of multiple learning episodes, often covering a distribution of related tasks, and using this experience to improve future learning performance [10]. Meta-learning has emerged as a promising approach to expedite time-consuming tasks such as hyperparameter tuning by using meta-data [14,8]. This technique leverages the knowledge gained from previous experiences to learn more efficiently. By using a reduced number of performance examples, a meta-regressor can accurately predict the performance of new configurations without having to train them all. This enables the simulation of new performances in a more efficient manner.

Based on the above, and different from those approaches mentioned in literature [1,7,14], our main contribution constitutes a meta-learning formulation that predicts the accuracy of a deep model based on a regressor trained with a limited number of hyperparameter sets. To demonstrate the effectiveness of our proposed approach, we conducted experiments using chest medical images that include X-ray and computer tomography. First, we trained deep models using a grid search strategy, resulting in a collection of training examples. We then employed a meta-regressor to learn from these examples and predict the performance of new hyperparameter configurations, thus enabling the identification of high-performing hyperparameter settings for deep model training.

The proposed method was evaluated with an interpretable convolutional neural network using Class-Specific Gate (CSG) [12]. CSG considers a matrix G that is used to train specific convolutional kernels for each class in the penultimate layer of a deep model. In the ideal case, each kernel belongs to a single class, however, in order to reduce the complexity of the implementation, a kernel might share at most two classes. During training, two paths are followed. In the first one, a standard training is carried out in which all convolutional kernels are considered for the backpropagation step. In the second path, some epochs considering the matrix G are performed. Authors argue that the obtained models are more interpretable for humans than the optimized with standard training.

Our experiments show that the proposed method accurately predicts how well different hyperparameter configurations will perform. By using this method, we can simulate new experiments and reduce the time needed for hyperparameter tuning when training deep models. We also tested our method on two different types of chest images to show that it works well in different scenarios.

This paper is organized as follows: in section 2 we introduce background of meta-learning: the proposed formulation for hyperparameter tuning is presented in section 3; the experimental results are introduced in section 4; finally, in section 5 we show the conclusions and future work.

2 Meta-Learning

Meta-learning is the process of distilling the experience of multiple learning experience - often covering a distribution of related tasks - and using this experience to improve future learning performance[10]. The framework is further developed by introducing a mechanism which allows the model to incorporate knowledge

from multiple related tasks. This is achieved by using a combination of meta-learning techniques, such as model architecture search [6] and hyperparameter optimization [14]. At the end of the paper, we discuss how this framework can be applied to reduce the time spent on hyperparameter tuning for deep learning models.

In conventional machine learning, a training dataset contains pairs of input (x_i) and output (y_i) values, represented as $D = (x_1, y_1), \ldots, (x_n, y_n)$. The goal of training a neural network is to learn a function that can accurately predict output values from inputs. This function is typically represented as it is shown in Equation 1, where θ is a set of parameters, such as weights in a neural network. The parameters are learned through an optimization process that minimizes the difference between the predicted outputs and the predicted outputs in the training dataset.

$$\hat{y} = f_\theta(x) \tag{1}$$

Meta-learning is a learning technique that aims to obtain knowledge that can be used to learn the parameters θ of a model [10]. This is typically achieved by solving an optimization problem in the form of Equation 2, where $\mathcal{L}$ is a loss function that measures the error between the true labels and the predicted ones, and ω are assumptions about "how to learn", such as the hyperparameters of a neural network. In this scheme, the optimization problem is solved across a set of related learning tasks, such as classification or regression problems, in order to learn a set of parameters that are better suited for these tasks. The resulting meta-learned parameters can then be used to improve the performance of a model on new, unseen tasks.

$$\theta^* = \arg\min_\theta \mathcal{L}(D; \theta, \omega) \tag{2}$$

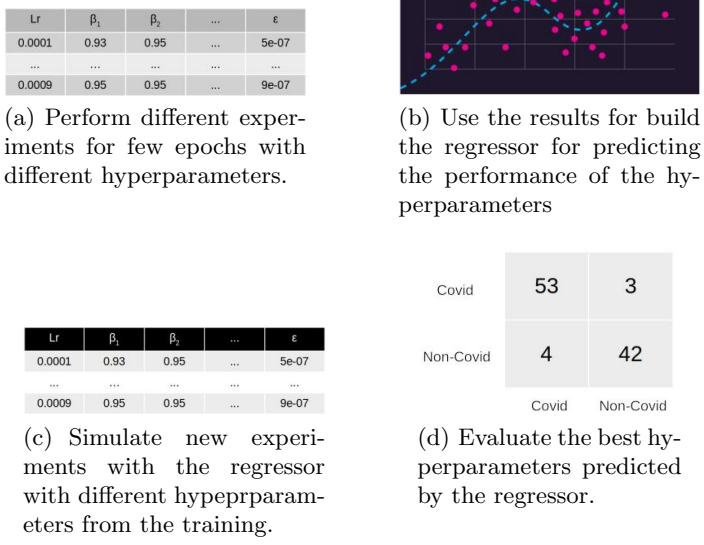
This study demonstrates the effectiveness of using a meta-regressor to approximate the accuracy of different hyperparameter configurations for training deep models. The meta-regressor, which plays the role of the ω in meta-learning, was trained using multiple episodes collected from training the model using different hyperparameter configurations (ω) following a grid search strategy. By learning from the collected episodes, the meta-regressor can accurately predict the performance of new hyperparameter configurations without having to perform an exhaustive grid search. The experimental results presented in this study show that the proposed method can achieve high accuracy in predicting the performance of different hyperparameter configurations.

3 Meta-Learning for hyperparameter tuning

This section outlines the proposed method for hyperparameter tuning using meta-learning. Specifically, we introduce the use of a meta-regressor to predict the performance of a hyperparameter configuration when used to train a deep model. The meta-regressor takes important hyperparameters for convolutional

neural network training as input and outputs the predicted accuracy. This approach aims to improve the efficiency of the hyperparameter tuning task by reducing the number of configurations that need to be trained and evaluated.

The proposed formulation can be seen graphically in Figure 1. First, we need some examples of the training performance. In order to obtain those samples, we ran experiments following a grid search strategy [13]. To reduce the training time we ran the experiments only for few epochs, hypothesizing that if the performance increases in few epochs it will increase even more in subsequent epochs [20].



(a) Perform different experiments for few epochs with different hyperparameters.

(b) Use the results for build the regressor for predicting the performance of the hyperparameters

(c) Simulate new experiments with the regressor with different hypeprparameters from the training.

(d) Evaluate the best hyperparameters predicted by the regressor.

Fig. 1. Proposed method for hyperparameter tuning with meta-learning: (a)) Experimental results of few epochs using a grid search; (b) The obtained results are used to build a regressor for predicting the performance of the hyperparameters; (c) Simulation of new experiments with a regression model with new samples; (d) Evaluation of the best hyperparameters according to the results obtained in (c).

The results obtained from training deep models with different hyperparameter configurations were used to train the meta-regressor, as described in the background section. Specifically, some important the hyperparameters of the Adam optimizer [11] (learning rate, β_1, $beta_2$, ϵ) were used as features to train the meta-regressor for standard training. For CSG training, relevant hyperparameters related to interpretability were selected as features (see Table 1). The meta-regressor was trained using a set of episodes, each consisting of a combina-

tion of hyperparameters and their corresponding validation accuracy after a few epochs of training. Formally, the meta-regressor was trained to learn the mapping $\omega \rightarrow \mathcal{L}_{val}$, where ω represents the used hyperparameters and $\mathcal{L}_{val}$ represents the validation loss after a certain number of epochs.

To further evaluate the effectiveness of the proposed method, we conducted simulations of new experiments using the trained meta-regressor. We simulated experiments to explore if there was any hyperparameter configuration that outperformed the best result obtained in the previous grid search. Afterwards, we evaluated the performance of the selected hyperparameters configuration, identified by the meta-regressor, by training a deep model using the corresponding HC.

To clarify the objectives of the simulation, our aim is to observe whether there exists a hyperparameter configuration that performs better than the ones obtained through the grid search. By using the meta-regressor, we are able to simulate the performance of different hyperparameters without running the entire grid search, thereby reducing the time required for hyperparameter tuning. The method also allows us to obtain a hyperparameter configuration with good results in a shorter time compared to training the entire grid search. Furthermore, the proposed method can be easily adapted to different deep learning architectures and datasets, as long as a representative set of hyperparameter configurations is available for training the meta-regressor. Overall, the proposed method offers a practical and time-saving approach to hyperparameter tuning, which can lead to significant reductions in computational costs when conducting deep learning experiments.

4 Experimental Results

The aim of the experiments was to show that with a regression model, the experiments of hyperparameter tuning could be reduced simulating different HCs. In this section, we first describe the dataset. Then, we introduce the results obtained fitting the regression model. Finally, the results of the validation stage of the meta-regressor are presented.

4.1 Datasets description

We used the updated kaggle X-ray dataset for COVID-19 detection. The number of images in the dataset are 21,165 images comprising four different classes: COVID-19, normal, pneumonia and lung opacity [4]. Additionally, we used 199 training volumes from the 2020 COVID-19 Lung CT Lesion Segmentation Challenge [18]. The volumes belong to chest computer tomography from patients with positive RT-PCR for SARS-CoV-2. In these experiments, we used a subset of 4,891 slices. Classes were balanced, so they were randomly sampled among images with and without lungs. In the present study, a random stratified selection approach was employed to partition both datasets into training (80%) and testing (20%) subsets, with 20% of the training subset reserved for validation purposes.

4.2 Results of the meta-regressor

As discussed in section 3, in order to train the meta-regressor, it is necessary to gather examples of the system performance along with their corresponding sets of hyperparameters. To collect this data, we employed a grid search technique that systematically explores a range of values for the most significant hyperparameters of the learning method, examining its response to different hyperparameter configurations. This allowed us to assemble a comprehensive dataset encompassing a diverse set of hyperparameter combinations and their corresponding system performance measures, which was subsequently used to train the meta-regressor.

For the X-ray images, we followed a standard training procedure using the Adam optimizer [11]. The hyperparameters used to apply the grid search included the learning rate, β_1, β_2, and ϵ, this is because these hyperparameters are important during the optimization of the neural network weights. In this experiments, we used the VGG16 architecture [19], the agents was trained for five epochs to reduce the overall training time.

In the CSG training, we employed the hyperparameters listed in Table 1 for the grid search. These experiments used a computer tomography dataset to demonstrate the robustness of the proposed method. Since the tomography dataset contains more images than the X-ray dataset, we tested our method with a lighter architecture, specifically the VGG11 architecture. This also allowed us to validate our method with different neural network architectures. The agent was trained for 30 epochs.

Table 1. Hyperparameters used for the grid search in CSG training.

Hyperparameter	Desciption
Learning rate (LR)	Learning rate for the optimizer
Mask Period (MP)	Number of epochs the period is alternated
Mask epoch min (MEM)	Epochs applying CSG training
Lr reg (L_REG)	Learning rate of the loss regularization path
λ reg (λ_REG)	Regularization coefficient

A computer with a Titan RTX graphic card, 128 GB of memory and a AMD thread tripper 3970 processor with 64 cores was used for experimentation. In order to show an empirical result of the training time during the hyperparameter tuning task, we ran the experiments of the grid search for the X-ray dataset, which took approximately 5 hours (each epoch takes 30 seconds). In contrast, training with the computer tomography dataset took almost 10 days (each epoch takes near to 1 minute).

We selected Random Forests (RF) [2] and Support Vector Regressor (SVR) [5] to build the regressor model due to their robustness to overfitting, and generalization capabilities. In the Random Forest (RF) model, we used a forest consisting of 100 decision, this empirically chosen value yielded the best performance across various practical applications [15]. In the case of the Support Vector Regression

(SVR) model, multiple experiments were conducted using multiple kernels, including the radial basis function, linear, and polynomial kernels. The remaining parameters were set according to the default values recommended by the scikit-learn library [16].

During the grid search, the data collected was partitioned into a training set comprising 80% of the samples and a validation set comprising the remaining 20%. To ensure robustness and avoid potential bias from a single data split, we followed the methodology outlined in Raschka and Mirjalili work [17]. Specifically, we repeated the data splitting process one hundred times, each time using a different random seed. The results of this iterative selection process are summarized in Table 2, which presents the mean and standard deviation values obtained from these repetitions. Notably, the RF regressor exhibited superior performance in terms of mean squared error compared to the SVR. Consequently, we selected the RF model as the most promising candidate for validating new hyperparameter configurations. Finally, using the entire dataset, we constructed the meta-regressor.

Table 2. Results of the meta-regressor training. We show the results for both datasets, X-ray and Computer Tomography (CT). Random Forest regressor obtained the best performance with the lowest mean squared error (MSE).

Algorithm	X-ray	CT
RF	0.0122 ± 0.003	$3.96^{-5} \pm 3.34^{-5}$
SVR-RBF	0.0160 ± 0.002	0.0010 ± 0.0002
SVR-Poly	0.0159 ± 0.002	0.0010 ± 0.0003
SVR-Linear	0.0266 ± 0.004	0.0010 ± 0.0003

4.3 Validation with new data

In this section, we present the outcomes of simulated experiments using the regression model. To validate the model's performance, we assessed various hyperparameters in conjunction with the regression model chosen in the preceding subsection (a Random Forest algorithm with 100 decision trees).

The simulations with the meta-learner were tested on a computer with less resources than the training computer described in the previous subsection. The computer has a core i7-6560 processor with 8 GB of memory. The experiments were run in a single core of the computer. The objective of employing a computer system with limited resources was to demonstrate the diminished duration required for hyperparameter tuning, as compared to a more robust computational platform.

We simulated 250,500 HCs for the X-ray dataset. Considering that each experiment with 5 epochs takes about 30 seconds in the computer with the Titan GPU for the X-ray dataset, if we trained the entire grid search it would take approximately 70.31 days, while the simulation of the experiments in the computer

described in the above paragraph took 23.98 minutes. Although the simulation results for the top HC scores showed similar predicted accuracy, it is important to note that the validation of the HC space was performed using a range of percentiles rather than solely relying on the Top-HC. Table 3 presents some examples of the HC scores at different percentiles of the space.

Table 3. Best hyperparameter configurationS for the X-ray dataset with corresponding predicted accuracies of the regression model.

LR	β_1	β_2	ϵ	Predicted Accuracy	Real Accuracy
0.0001	0.82	0.70	$5x10^{-7}$	0.9580	0.9527
0.0001	0.82	0.89	$4x10^{-7}$	0.9509	0.9527
0.0001	0.62	0.90	$5x10^{-7}$	0.9450	0.9312
0.0001	0.59	0.52	$9x10^{-7}$	0.9400	0.9398

The next experiment aims to show the effects of increasing the number of epochs on the training of the agent. The learning curve of the best HC is depicted on Figure 2. It can be seen that in the fifth epoch the predicted performance is similar to the real one (0.0053%). Then, we trained the model for 20 epochs and obtained 96.57% accuracy, which is an acceptable performance.

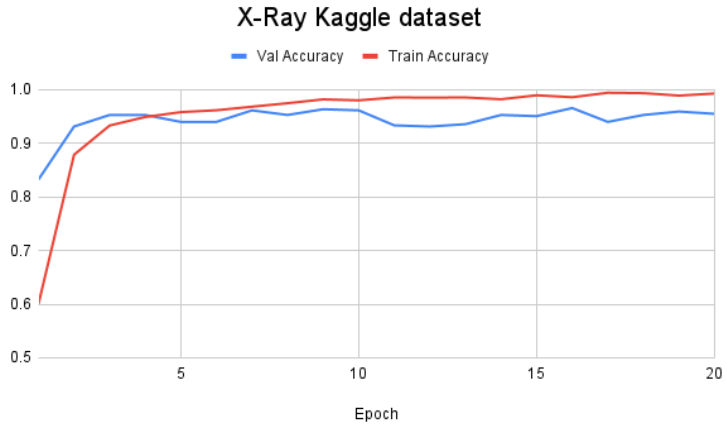


Fig. 2. Learning curve for the best HC. The accuracy at the fifth epoch is 95.27% and the highest accuracy was 96.57% training with 20 epochs.

Table 4 presents some of the HCs obtained for the computer tomography dataset. We initially used the first parameter configuration, expecting it to yield the best performance. However, the first and second HCs achieved similar per-

formance. We also followed a similar strategy of the X-ray dataset to illustrate some parts of the hyperparameter space. Subsequently, we trained the agent with those hyperparameters, and obtained an accuracy of 98.87%, which was close to the predicted performance of 99.17%. Figure 3 shows the learning curve for this training. Despite the second HC yielding better performance, our approach accurately predicted the performance of the meta-regressor in general terms.

We simulated 108,864 HCs for this dataset, each experiment took approximately 30 minutes. Training the entire grid search would take about 54,432 days. However, with the regression model it took 15.66 minutes in the computer with limited resources, which is considerable less time than training all the HCs.

Table 4. Best hyperparameter configurations for the computer tomography dataset with corresponding predicted accuracies of the regression model.

LR	MP	MEM	L_REG	λ_REG	Predicted Accuracy	Real Accuracy
0.0006	5	5	0.007	0.004	0.9919	0.9887
0.0003	6	6	0.007	0.005	0.9896	0.9928
0.0003	4	2	0.003	0.005	0.9799	0.9856
0.0001	5	1	0.004	0.005	0.9699	0.9754

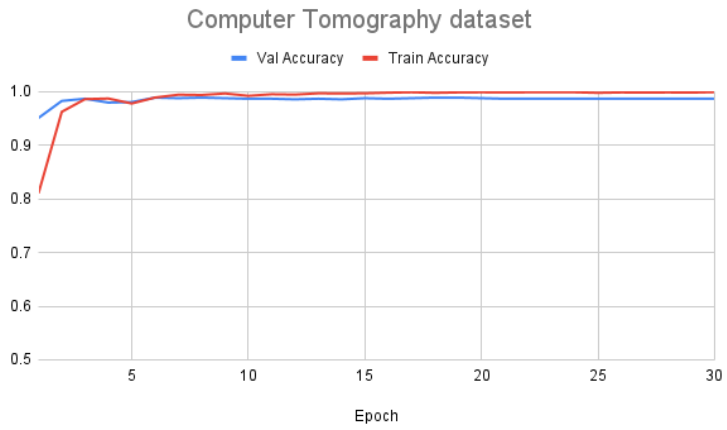


Fig. 3. Learning curve for the best HC. The best accuracy obtained after 30 epochs was 99.08%.

5 Conclusions and Future Work

Based on the clear findings presented in this paper, our proposed meta-learning approach exhibits promising potential in addressing the time-consuming nature of hyperparameter tuning tasks within deep learning models. We have demonstrated the robustness of our approach by involving X-ray images and a computer tomography dataset. Also, leveraging an interpretable convolutional neural network (CNN) and conducting experiments with CSG training and various architectural configurations.

By incorporating relevant hyperparameters as input during the training phase, our method effectively predicts the accuracy of deep models. This indicates its ability to provide valuable insights and inform decision-making regarding hyperparameter selection. However, it is important to discuss the primary limitation of our method, which lies in the requirement for some examples of the deep model performance to train the meta-regressor. This reliance on a sufficient number of performance examples may present challenges in scenarios where such data is limited or unavailable.

In the future, this technique could be extended to other types of deep learning models and datasets, potentially improving the efficiency of hyperparameter tuning across a range of applications. Also, we will apply our formulation to other performance evaluation functions such as F1 measure, precision, etc. Moreover, we wish to study the interpretability of the deep learning models and verify if it can be predicted with the proposed approach as well.

Acknowledgments

This work was supported by the Universidad Nacional Autónoma de México by means of PAPIIT grants TA101121 and IV100420. Rodrigo Ramos Díaz acknowledges CONACYT for the scholarship that supports his PhD studies associated with CVU number 927245.

References

1. Balaji, Y., Sankaranarayanan, S., Chellappa, R.: Metareg: Towards domain generalization using meta-regularization. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31, pp. 1–11. Curran Associates, Inc. (2018)
2. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
3. Ceron, J.S.O., Castro, P.S.: Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In: International Conference on Machine Learning. pp. 1373–1383. PMLR (2021)
4. Chowdhury, M.E., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M.A., Mahbub, Z.B., Islam, K.R., Khan, M.S., Iqbal, A., Al Emadi, N., et al.: Can ai help in screening viral and covid-19 pneumonia? IEEE Access **8**, 132665–132676 (2020)
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine learning **20**, 273–297 (1995)

6. Ding, Y., Wu, Y., Huang, C., Tang, S., Yang, Y., Wei, L., Zhuang, Y., Tian, Q.: Learning to learn by jointly optimizing neural architecture and weights. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 129–138 (June 2022)
7. Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., Pontil, M.: Bilevel programming for hyperparameter optimization and meta-learning. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1568–1577. PMLR (10–15 Jul 2018)
8. Garouani, M., Ahmad, A., Bouneffa, M., Hamlich, M., Bourguin, G., Lewandowski, A.: Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data. Journal of Big Data **9**(1), 57 (2022)
9. Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D.: Rainbow: Combining improvements in deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence (2018)
10. Hospedales, T., Antoniou, A., Micaelli, P., Storkey, A.: Meta-learning in neural networks: A survey. IEEE transactions on pattern analysis and machine intelligence **44**(9), 5149–5169 (2021)
11. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
12. Liang, H., Ouyang, Z., Zeng, Y., Su, H., He, Z., Xia, S.T., Zhu, J., Zhang, B.: Training interpretable convolutional neural networks by differentiating class-specific filters. In: European Conference on Computer Vision. pp. 622–638. Springer (2020)
13. Liashchynskyi, P., Liashchynskyi, P.: Grid search, random search, genetic algorithm: a big comparison for nas. arXiv preprint arXiv:1912.06059 (2019)
14. Occorso, M., Sabbioni, L., Metelli, A.M., Restelli, M.: Trust region meta learning for policy optimization. In: ECMLPKDD Workshop on Meta-Knowledge Transfer. pp. 62–74. PMLR (2022)
15. Oshiro, T.M., Perez, P.S., Baranauskas, J.A.: How many trees in a random forest? In: Machine Learning and Data Mining in Pattern Recognition: 8th International Conference, MLDM 2012, Berlin, Germany, July 13-20, 2012. Proceedings 8. pp. 154–168. Springer (2012)
16. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
17. Raschka, S.: Model evaluation, model selection, and algorithm selection in machine learning. arXiv preprint arXiv:1811.12808 (2018)
18. Roth, H.R., Xu, Z., Tor-Díez, C., Jacob, R.S., Zember, J., Molto, J., Li, W., Xu, S., Turkbey, B., Turkbey, E., et al.: Rapid artificial intelligence solutions in a pandemic—the covid-19-20 lung ct lesion segmentation challenge. Medical image analysis **82**, 102605 (2022)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)
20. Xiao, X., Yan, M., Basodi, S., Ji, C., Pan, Y.: Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm. arXiv preprint arXiv:2006.12703 (2020)