

Comparison of the Bayesian Optimization-based Tree-structured Parzen Estimator Algorithm with the Bandit-Based Successive Halving Algorithm in Fully Convolutional Neural Network Finetuning for Semantic Segmentation

Gergő Bendegúz Békési Péter Ekler

*Department of Automation and Applied Informatics
Budapest University of Technology and Economics*

bekesi.gergo@aut.bme.hu

Abstract. This work contributes to a measurement automation project by improving its vision system, whose task is to create high-quality images and process them to aid robot navigation. One of the image processing challenges for the vision system is semantic segmentation, which is carried out by fully convolutional neural networks. The focus of this work is the hyperparameter fine-tuning of these fully convolutional neural networks. First, the basic concept of the Bayesian optimization-based Tree-structured Parzen Estimator Algorithm is presented. Next, the main characteristics of the bandit-based Successive Halving algorithm are highlighted. The goal of this work is to compare the performance of these methods under the same runtime conditions. The comparison is made using internationally accepted metrics: Intersection over Union, Mean Class Accuracy, and Pixel Accuracy.

Keywords: Hyperparameter Optimization; Deep Learning; Semantic Segmentation, Successive Halving Algorithm, Tree-structured Parzen Estimator Algorithm

1 Introduction

Due to global trends such as electrification, proliferation of smart devices, increasing electrical energy consumption, the expansion of electronics and automotive electronics, and the growing popularity of electric cars, special emphasis is being placed on the electronics industry, including the direct bonded copper (DBC) substrate industry. These electronic devices must deal with more and more heat power, because of the increasing electrical energy consumption, although their surfaces are getting smaller and smaller by the time. This trend is a serious technical challenge, since too high temperature is the leading cause of

failure. Thus, thermal examination and identification is extremely important, which can be performed by the T3Ster equipment [1]. This work is connected to a project at Robert Bosch Ltd that aimed to automate and accelerate T3Ster measurements. Figure 1 illustrates a power module using DBC technology.

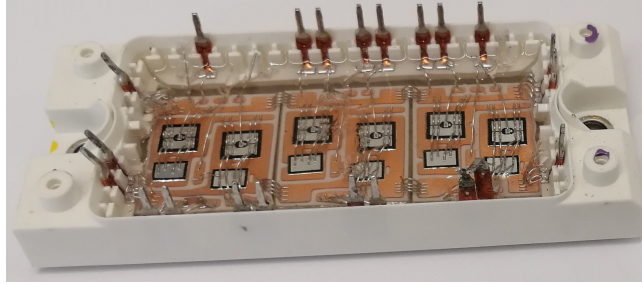


Figure 1: Illustration of a power module using DBC technology [2]

A vision system was developed as a complex electronic device for a robot, controlled in a LabVIEW environment. As a result, the system can help the robot to detect the DBCs. After the detection, the robot can automatically obtain the required measurements. The hardware and software development for the system was also carried out. A camera with appropriate sensor and optics was selected, and a server platform capable of serving several computers that would control or monitor the measurement at the same time with the help of a Raspberry Pi 4 Model B was implemented. The tool that supports mechanical integration was designed in Autodesk Inventor Professional 2022, and then a prototype with a 3D printer was created. The measurement automation system is shown in Figure 2.

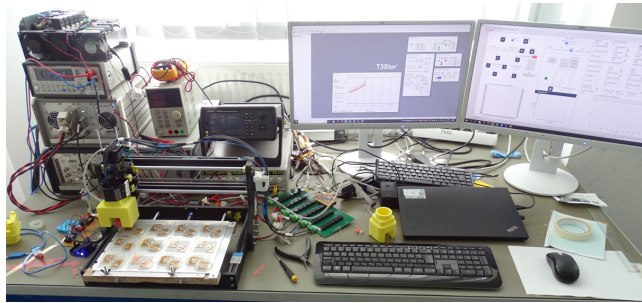


Figure 2: The measurement automation system

In the most emphatic part of this work semantic segmentation and keypoint detection was performed for robot navigation using artificial intelligence, fully convolutional neural networks (FCN), and computer vision algorithms. FCNs were implemented and the dataset necessary for the teaching was created. That dataset was made with the camera and optics that are used during real measurements and annotation happened manually. Neural networks were studied

comprehensively. Using automated runs, networks were taught and evaluated, modifying depth, number of channels, step size, learning rate, learning rate reduction ratio, optimizer algorithm, regularization and dilation. It was done in the open-source Python 3 programming language primarily using the OpenCV library [3], the Pytorch library [4], the Optuna library [5], and the Torchvision library [6]. The two algorithms compared in this work served this hyperparameter tuning.

2 Methods

In this chapter, the main characteristics of the FCN architecture to be optimized and the optimization process are highlighted.

2.1 FCN architecture

As a result of a detailed literature search, FCNs were found to be the most suitable for semantic segmentation because conventional machine learning algorithms, like K-means clustering, did not provide an accurate solution [7]. The effectiveness of FCNs is confirmed by [8], [9], and [10]. The developed network, which needs to be fine-tuned, consists of several downscalings and upscalings with convolutions and transposed convolutions. Each scaling convolution is preceded by a non-scaling convolution, and after each convolution, batch normalization is also performed. Downscaling is done by a halving factor; however, setting the channel numbers is the task of hyperparameter optimization. Convolution kernels are always 3x3 kernels with a padding of 1 and a dilation of 0 or 1. This can be set by the hyperparameter optimizer, which is also responsible for setting the appropriate step size, learning rate, learning rate reduction ratio, optimizer algorithm, and regularization. Shortcut connections are also applied between same-sized layers to perform identity mapping by skipping layers. The general architecture is shown in Figure 3.

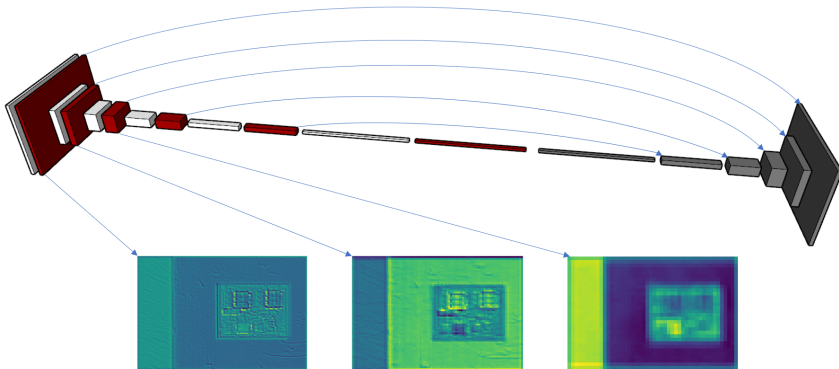


Figure 3: The general FCN architecture [11]

The parameter optimization algorithm is selected from the state-of-the-art Adam [12] and the original SGD methods. The ranges of the numerical hyperparameters are given in Table 1.

Table 1: Ranges of the numerical hyperparameters.

Hyperparameter	Minimum value	Maximum value
Depth	1	7
Channel numbers	6	16
Learning rate	10^{-4}	10^{-2}
Learning rate reduction ratio	0.1	0.5
Step size	10	30
Weight of L2 regularization	10^{-5}	10^{-3}
Dilation	0	1

In this 8-dimensional hyperparameter space, the optimization happened with a maximum epoch number of 100 and a batch size of 64.

2.2 Hyperparameter Optimization

Machine learning models have parameters and hyperparameters as well. The parameters are set by the parameter optimization algorithms, and the hyperparameters are set by the hyperparameter optimization algorithms. However, hyperparameters are not optimized during training, but before. There are several options to solve this task. One of them is manual hyperparameter optimization, which is also the least efficient. The grid search method, which consists of several nested loops in the hyperparameter space, is slightly better than manual search. Its main advantage is that it is automated, just like the next method, the random search in the hyperparameter space. Based on practical experience, random search is better than grid search, but modern hyperparameter optimization algorithms perform even better. According to Figure 4, these algorithms can be divided into Black-Box based and Multi-Fidelity methods. The methods presented in the work belong to the categories highlighted in green.

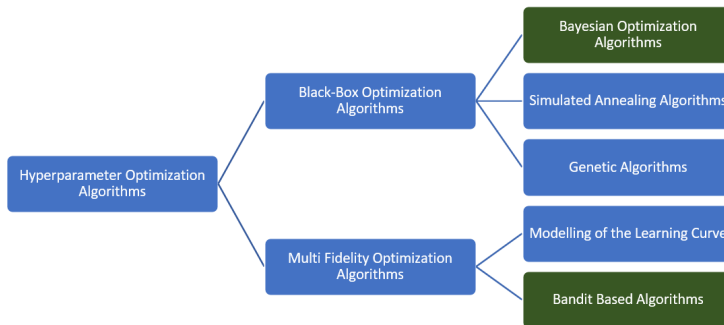


Figure 4: Hyperparameter optimization algorithms

By the way, there are many applications for genetic algorithms [13], simulated annealing algorithms [14], and learning curve modeling [15]. However, the Tree-structured Parzen Estimator (TPE) algorithm is a Bayesian optimization algorithm, and the Successive Halving (SH) algorithm is a bandit-based method. These are the focus of this case study. The purpose of this study is to compare these solutions, although there are many other methods in both categories, such as Gaussian Process [16] and Hyperband [17].

2.2.1 Tree-structured Parzen Estimator Algorithm

This iterative algorithm was developed [18] to provide the best possible latency improvement by achieving an expected accuracy target in terms of the y loss function. It uses the historical evaluations of the hyperparameter configurations (x) to create probabilistic models, which can suggest the next set of hyperparameters for evaluations. TPE simulates $p(x|y)$ and $p(y)$ with Gaussian mixtures to indirectly obtain $p(y|x)$. According to Equation 1, the I model improvement is created by choosing a new configuration:

$$I(x) = ReLU(y^* - y(x)), \quad (1)$$

where y^* is the threshold value of the loss function and $ReLU$ is the widely used Rectified Linear Unit [19]. The expected improvement is calculated by Equation 2:

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y) \frac{p(x|y) \cdot p(y)}{p(x)} dy. \quad (2)$$

$p(x|y)$ is modeled as $g(x)$ if $y \geq y^*$, and as $l(x)$ otherwise, where $g(x)$ is the probability density of the loss values more than y^* , and $l(x)$ is the probability density of the remaining values. By factoring out $p(x)$ from Equation 2:

$$EI_{y^*}(x) = \frac{1}{p(y \geq y^*) \cdot g(x) + (1 - p(y \geq y^*)) \cdot l(x)} \int_{-\infty}^{y^*} (y^* - y) \cdot l(x) \cdot p(y) dy, \quad (3)$$

which is equivalent to Equation 4:

$$EI_{y^*}(x) = \frac{y^* - p(y \geq y^*) \cdot y^* - \int_{-\infty}^{y^*} p(y) dy}{1 - p(y \geq y^*) + p(y \geq y^*) \cdot \frac{g(x)}{l(x)}}. \quad (4)$$

TPE uses this last formula. It evaluates the hyperparameters iteratively by $g(x)/l(x)$ to find the hyperparameters x^* with the maximum expected improvement. During the optimization, the first decile of loss values were used as threshold value, and the whole process was started with 10 random evaluations.

2.2.2 Successive Halving Algorithm

The SH Algorithm is based on the multi-armed bandit methodology. Its basic concept comes from the field of gambling. In this case, the optimization also starts with random searches, but they run with a smaller number of epochs. After several trainings with different hyperparameter configurations and a smaller number of epochs, the best performers are selected. These best configurations are run for a few more epochs, and then the best ones are selected again. This method is repeated for a few iterations and finally the best performing architecture can be selected. This method allows for more configuration tests than TPE for the same amount of resources. This can be explained by the fact that it uses fewer resources to evaluate weaker architectures. In this case, the strategy for selecting configurations is highlighted in Table 2.

Table 2: Strategy for selecting configurations.

Number of configurations to evaluate	Starting epoch	Finishing epoch
33	1	34
16	35	67
8	68	100

So SH uses approximately the same amount of resources as TPE, which ran for $19 \cdot 100$ epochs.

3 Results

The essence of semantic segmentation is to determine the class of each pixel in an image. Thus, the resolution of its output is the same as the resolution of the input. There are as many types of output for each pixel as classes. For this work, it must be decided for each pixel whether it is a DBC or not. Thus, two values can be obtained, 1 or 0. The widely used and accepted quality measure of this decision is the Intersection over Union (IoU) value, which quantifies the overlap ratio between the segmentation output and the actual segmentation. This is described as follows:

$$IoU = \frac{\text{output of segmentation} \cap \text{solution of segmentation}}{\text{output of segmentation} \cup \text{solution of segmentation}}. \quad (5)$$

Pixel accuracy is a simpler metric, but less robust for small objects. It is calculated by Equation 6:

$$\text{Pixel accuracy} = \frac{\text{correctly classified pixels}}{\text{all pixels}}. \quad (6)$$

The mean class accuracy of the validation set, which is the mean of the pixel accuracies of pixels with a value of 0 and pixels with a value of 1, is also calculated. Furthermore, the widely used Cross-entropy loss values [20] of the evaluations are highlighted in the following figure.

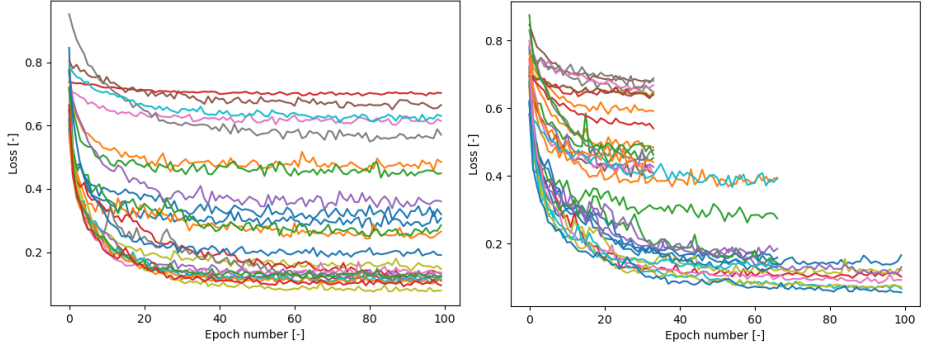


Figure 5: TPE and SH training losses

Figure 5 shows that the training loss of SH is lower than that of TPE. The best performing TPE configuration has a final loss of 0.079, which is not as good as the best performing SH architecture with a loss of 0.056. This is also true for validation losses. A value of 0.046 outperforms a value of 0.075. It is important to emphasize that the validation loss is smaller than the training loss in both cases, so the overfitting was avoided.

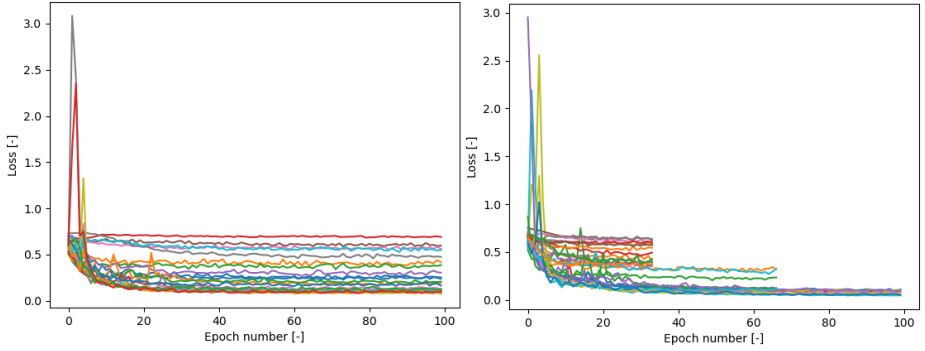


Figure 6: TPE and SH validation losses

Similar phenomena can be observed in terms of pixel accuracy. The validation values are better and the bandit-based method performs better. The best SH results are 0.9783 and 0.9823. And the best TPE results are 0.9702 and 0.9703.

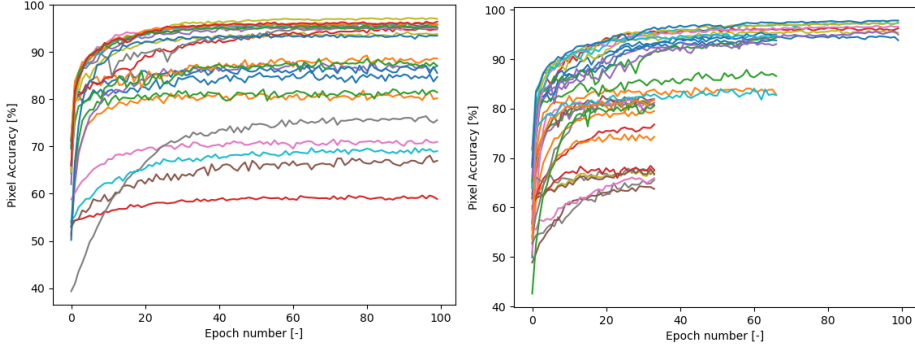


Figure 7: TPE and SH training pixel accuracies

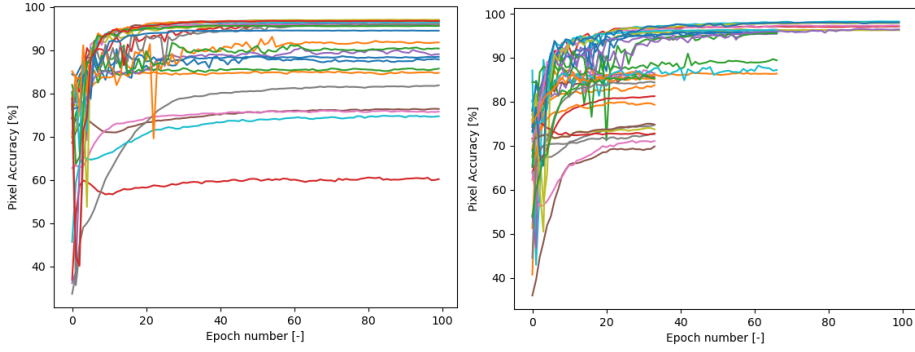


Figure 8: TPE and SH validation pixel accuracies

SH outperforms TPE in both mean class accuracy and IoU. The best mean class accuracies are 0.9784 and 0.9653. And the best IoUs are 0.9507 and 0.9249.

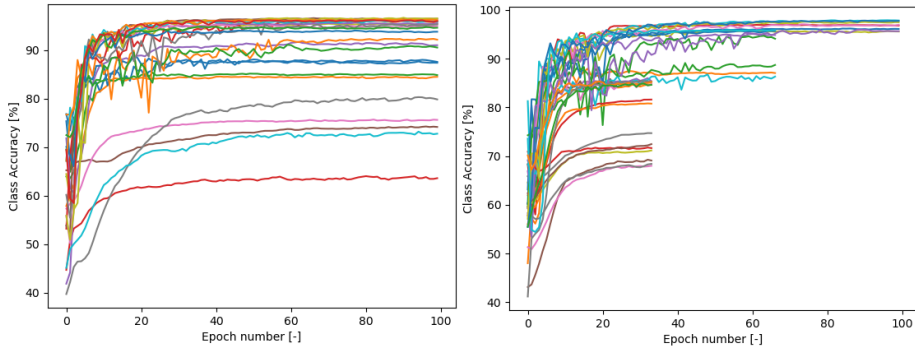


Figure 9: TPE and SH validation mean class accuracies

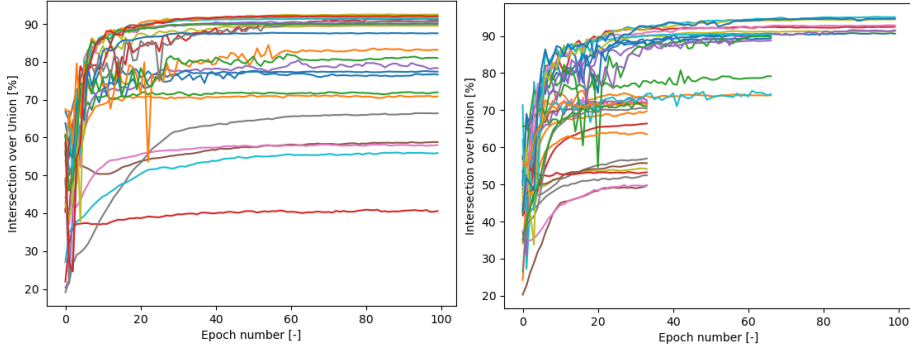


Figure 10: TPE and SH validation Intersection over Union values

4 Summary

An electronic system was created with a robust implementation of navigation to the DBCs, which are measured. $100\ \mu\text{m}$ and 0.75° accuracy was achieved, which was associated with an IoU value of 98.37% accuracy for semantic segmentation previously [11]. This was the result of over 25 days of TPE optimization, however, in a shorter run, SH outperformed TPE. Of course, the results of the SH method have not yet surpassed those of the 25-day training, but it can be seen that perhaps more resources should be allocated to this method in the future. This is because, in 5-day training, SH achieved a 95.07% IoU, which is significantly better than the 92.49% IoU of TPE. Furthermore, SH has better performance in terms of mean class accuracy and pixel accuracy as well. It outperformed TPE with 97.84% and 98.23%, compared to TPE's 96.53% and 97.03%, respectively. Another advantage of SH over TPE is that it can be operated better with parallel resources. An NVIDIA Quadro RTX 3000 GPU and an Intel Core i7-10850H CPU were used for both the referenced research and current one. Figure 11 illustrates a result of a semantic segmentation.

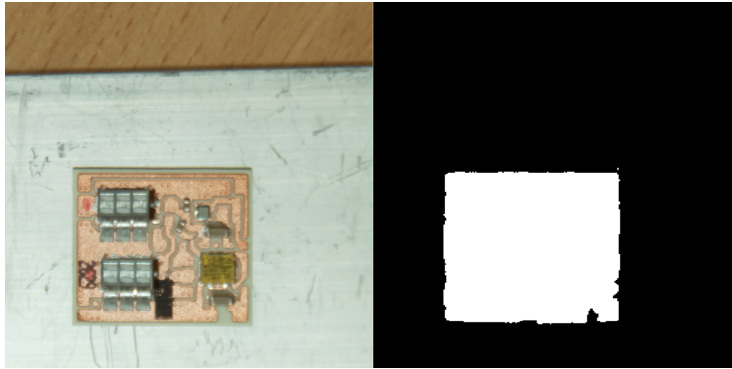


Figure 11: Visualization of semantic segmentation

References

- [1] “Simcenter T3STER: Characterize the thermal properties of components.” <https://plm.sw.siemens.com/en-US/simcenter/physical-testing/t3ster/>, 2022. Date Accessed: 04. 28. 2023.
- [2] A. Yahyae, A. S. Bahman, and F. Blaabjerg, “A modification of offset strip fin heatsink with high-performance cooling for igbt modules,” *Applied Sciences*, vol. 10, no. 3, 2020.
- [3] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [4] 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*, pp. 8024–8035, Curran Associates, Inc., 2019.
- [5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna: A next-generation hyperparameter optimization framework,” in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.
- [6] D. Falbel, *torchvision: Models, Datasets and Transformations for Images*, 2022. <https://torchvision.mlverse.org>, <https://github.com/mlverse/torchvision>, Date Accessed: 28. 04. 2023.
- [7] G. B. Békési, “Measurement automation of MOSFETs and other electronic components using artificial intelligence, neural networks and computer vision,” *Scientific Students’ Association Report, BME*, 2021.
- [8] Y. Guo, Y. Liu, and T. Georgiou, “A review of semantic segmentation using deep neural networks,” *International Journal of Multimedia Information Retrieval*, vol. 7, pp. 87–93, 2018.
- [9] F. Lateef and Y. Ruichek, “Survey on semantic segmentation using deep learning techniques,” *Neurocomputing*, vol. 338, pp. 321–348, 2019.
- [10] B. Li, Y. Shi, Z. Qi, and Z. Chen, “A survey on semantic segmentation,” in *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 1233–1240, 2018.
- [11] G. B. Békési, “Vision system development for thermal transient measurement automation of dbcs using fully convolutional deep learning neural networks, image processing algorithms and tree-structured parzen estimator,” *Scientific Students’ Association Report, BME*, 2022.
- [12] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014.

- [13] N. M. Aszemi and P. Dominic, “Hyperparameter optimization in convolutional neural network using genetic algorithms,” *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, 2019.
- [14] S. Xavier-de Souza, J. A. K. Suykens, J. Vandewalle, and D. Bolle, “Coupled simulated annealing,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 2, pp. 320–335, 2010.
- [15] T. Domhan, J. T. Springenberg, and F. Hutter, “Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves,” in *Twenty-fourth international joint conference on artificial intelligence*, 2015.
- [16] E. Schulz, M. Speekenbrink, and A. Krause, “A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions,” *Journal of Mathematical Psychology*, vol. 85, pp. 1–16, 2018.
- [17] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.
- [18] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyperparameter optimization,” *Advances in neural information processing systems*, vol. 24, 2011.
- [19] A. F. Agarap, “Deep learning using rectified linear units (relu),” *arXiv preprint arXiv:1803.08375*, 2018.
- [20] Q. Wang, Y. Ma, K. Zhao, and Y. Tian, “A comprehensive survey of loss functions in machine learning,” *Annals of Data Science*, vol. 9, no. 2, pp. 187–212, 2022.