

Proceedings of the
*Workshop on the
Advances of Information Technology*
WAIT 2023

Budapest
2023

Contents

Deep learning

Bálint Szabó, Omer Saad Alkhafaf, Ameer Basheer, Ákos Szlávecz, Béla Paláncz, Geoff Chase and Balázs Benyó Comparison of Neural Network Models for Quantile Regression Based Insulin Sensitivity Prediction	1
Gergő Bendegúz Békesi Examining the hyperparameter optimization efficiency of the Tree-structured Parzen Estimator as a function of the number of initial random evaluations for the MNIST dataset.....	7
Bálint Tóth, Szabolcs Torma, Ákos Nagy and Luca Szegletes A novel deep learning-based blood pressure estimation system using ECG and PPG signals	15
Osama A. Salman and Gábor Hosszú Optimised feature dimension reduction method and its impact on the search for optimal trees	23

Visual informatics

András Fridvalszky and László Szirmay-Kalos Specular Ambient Lighting Model.....	29
Péter Salvi and Patrik Gál Tessellation of Zheng-Ball patches	35
Dóra Varnyú and László Szirmay-Kalos Comparison of Kinetic Macro-parameter Identification Methods for PET	40
Viktória Burkus, Attila Kárpáti and László Szécsi Generalization of Screen-Space Acceleration Methods for Isosurface Reconstruction on Secondary Rays	51

Robotics and autonomous Vehicles

Sándor Csurgai and Dániel Szabó	
Modelling, and control of a 2-DOF robot manipulator.....	55

Wafa Mezzi and Bálint Kiss

A comparative robustness study of nonlinear feedback laws for satellite attitude control	60
--	----

Mátyás Lévy, Barnabás Finta, Dániel Szabó, Anna Barbara Ádám, Emese Szádeczky-Kardoss

Effect of sensor parameters in the closed-loop control of a mechatronic system	66
--	----

Path planning in autonomous systems

Mátyás Szántó, Carlos Eduardo Hidalgo Vieira and Leonardo Dario Gonzalez Alarcon

Low Latency Map Updates for Automated Vehicle Maneuvering.....	71
--	----

Sára Szénási and István Harmati

Path Planning for Data Collection Multiagents System in a Sensing Field with Priority Nodes.....	79
--	----

Zoltán Gyenes, Emese Szádeczky-Kardoss and Ladislau Bölöni

Perception of obstacles at mobile robot's motion planning algorithm using Kalman filter and Particle filter	86
---	----

Mátyás Szántó and Márton Szemenyei

Reinforcement Learning-based Occlusion Avoidance using RGB-D Images	92
---	----

András Fridvalszky, László Szécsi, and László Szirmay-Kalos

Inverse Kinematic Simulation of Surgical Interventions	102
--	-----

Large software systems

Balázs Simon and Imre Szeberényi

Comparison of Unified Interface Description Language and Amazon Smithy.....	109
---	-----

Zoltán István Karsa and Imre Szeberényi

Using Infrastructure as Code tools in educational and research environments.....	116
--	-----

Preface

This book contains the papers of the *Eighth Workshop on the Advances of Information Technology* (WAIT 2023), organized as an annual conference by the *Department of Control Engineering and Information Technology (IIT)* of the *Budapest University of Technology and Economics (BME)*. The presented papers of the conference cover the fields of research of this institution, including deep learning, autonomous and assisted driving, robotics, computer graphics, computer aided geometric design, medical imaging, and software architecture.

László Szécsi prepared the proceedings for printing and designed the cover image.

Budapest, February 6, 2023.

Bálint Kiss and László Szirmay-Kalos
WAIT 2023 chairmen

Examining the hyperparameter optimization efficiency of the Tree-structured Parzen Estimator as a function of the number of initial random evaluations for the MNIST dataset

Gergő Bendegúz Békési

Faculty of Electrical Engineering and Information Technology, Budapest University of Technology and Economics

Abstract

Optimizing hyperparameters for a convolutional neural network is crucial for achieving good performance on a given task, as the choice of hyperparameters can significantly impact the model's ability to learn and generalize. Hyperparameter optimization using Tree-structured Parzen Estimator can be an effective method for finding the optimal set of hyperparameters for a machine learning model, as it can intelligently explore the search space and identify promising areas for improvement. However, optimal resource allocation for the initial iterations of TPE still carries significant potential for development. This work focuses on selecting the optimal number of TPE initialization iterations for a convolutional neural network consisting of two convolutional layers, in case of 100 runs. The best solution presented here is in the top 50 architectures according to Papers With Code, with 99.43% accuracy on the MNIST dataset, but the main contribution of this work is the examination of the initialization iteration number, which is a new approach. It can help machine learning model developments in the case of finite resources.

Categories and Subject Descriptors (according to ACM CCS): I.2.8 [Artificial Intelligence]: Hyperparameter Optimization

1. Introduction

Hyperparameter optimization (HPO) is the process of finding the best set of hyperparameters for a machine learning (ML) model. Hyperparameters are parameters that are not learned from data during model training, but rather are set before training. Furthermore, the loss function of the given ML model is not differentiable with respect to them in most cases, so gradient-based optimization methods can not be applied to find their optimal configuration. The importance of HPO lies in the fact that it can significantly affect the performance of a ML model. Finding the optimal set of hyperparameters can lead to better generalization performance on unseen data, as well as faster convergence during training. There are several techniques that can be used to optimize hyperparameters, which can be classified into four categories.

The two simpler methods of them are Grid Search (GS) and Random Search (RS). GS is the brute-force method that consists of a grid specification for hyperparameter values, and then model training and evaluation for each point of the

grid. It is computationally expensive and may not scale well to large datasets or complex models. RS is a slightly smarter solution, which involves sampling random combinations of hyperparameter values and training and evaluating a model for each combination. It is less computationally expensive than grid search, but it may still be inefficient for bigger datasets or complex architectures.

Of the four categories, the two that are competitive are Evolutionary Algorithms (EA) and Bayesian Optimization (BO) methods. EAs treat different hyperparameter combinations as a population of individuals and use operators such as selection, crossover, and mutation to evolve the population over time. While in BO methods, a probability distribution is placed over the space of possible hyperparameters, and this distribution is updated as the model is trained and evaluated. This allows the optimization algorithm to consider not only the performance of the model on the current set of hyperparameters, but also the uncertainty in the model's performance on other sets of hyperparameters.

Both types have many uses in Computer Vision (CV) and beyond. For example, Zhu et al.¹ used the EA-based particle swarm optimization technique to improve a convolutional neural network (CNN) for fault diagnosis. Also an EA was applied for the hyperparameter tuning of a multivariate time-series method by Silva et al.², while CV applications are well summarized by Nakane et al.³. BO methods can also be applied to these problems. Tang et al.⁴ carried out the same CNN tuning to improve the performance of fault diagnosis. Time series forecasting can be also supported with BO⁵, and its impressiveness in CV is proved by Bergstra et al.⁶. One of the key components of BO is the surrogate model, which is used to approximate the objective function and guide the optimization process⁷. The two most widely used categories of surrogate models are the Gaussian Processes (GP) and Tree-structured Parzen Estimators (TPE). The previously mentioned papers of Bergstra et al.^{6,7} also highlight that TPE, which is the scope of this work, outperforms GP in many cases.

This paper views TPE from a special angle. As it was mentioned a probability distribution is placed over the space of possible hyperparameters, but to be able to place this distribution random initial evaluations must be run first. The amount of these random iterations compared to the whole computational budget is not well examined yet and chosen without special justifications in most cases. In some cases, the initialization budget (IB) is less than 10%^{6,8}, but can be as high as 40%⁹ or over 60%¹⁰. So far, no general rule of thumb has been formulated for the TPE algorithm, as in the case of few evolutionary algorithms. For this, the optimal IB of the TPE must be examined in the case of benchmark datasets. This work takes one of the first steps towards this goal by examining the IB of the TPE in the case of the Modified National Institute of Standards and Technology (MNIST) dataset¹¹.

The rest of the paper is organized as follows. Section 2 introduces the methods and data used for the work. Results are presented and discussed in Section 3, and finally, conclusions are drawn in Section 4.

2. Methods and Data

In this section, I first describe the benchmark dataset, then present the multi-class classification model with its training methods, and finally highlight the TPE algorithm with its IB examination.

2.1. The MNIST dataset

The MNIST dataset¹¹ is a widely used dataset for training and testing image recognition models. It consists of a training set of 60,000 28x28 grayscale images of handwritten digits, along with a test set of 10,000 images. The images are labeled with the corresponding digit they represent, from 0 to 9.

This dataset is often used as a benchmark for ML algorithms, in the field of CV. It is considered a good starting point for developing and testing image recognition models due to its simplicity and relatively small size. So it is a valuable resource for researchers and developers working in the field of ML and CV, and it has played a key role in the development of many successful image recognition models^{12, 13, 14}.

2.2. The Multi-Class Classification Modell

In this subsection I use the c_1 , c_2 , d_1 , d_2 , lin , lr , g , $step$ expressions for the model hyperparameters which of optimization is highlighted in the next subsection.

CNNs have been very successful in many tasks, including multi-class classification^{12, 13, 14}, so I used a simple CNN with two convolutional layers and two fully-connected layers. The convolutional layers use filters to extract features from the input images. The first convolutional layer has 1 input channel and c_1 output channels, with a kernel size of 3x3. The second convolutional layer has c_1 input channels and c_2 output channels, also with a kernel size of 3x3. The output of each convolutional layer is passed through a ReLU¹⁶ activation function to introduce non-linearity.

The model also includes two dropout layers which randomly set d_1 and d_2 fraction of the input units to zero during training, in order to prevent overfitting. The first dropout layer is applied after the max pooling operation, which reduces the spatial dimensions of the output of the second convolutional layer by taking the maximum value in a 2x2 window.

The fully-connected layers are used to classify the input images based on the features extracted by the convolutional layers. The first fully-connected layer has $c_2 \times 12 \times 12$ input units and lin output units, and the second fully-connected layer has lin input units and 10 output units, corresponding to the 10 classes in the MNIST dataset. The second dropout with the fraction d_2 and another ReLU activation is applied between them. The output of the second fully-connected layer is passed through a log-softmax activation function, which gives the class probabilities for each image.

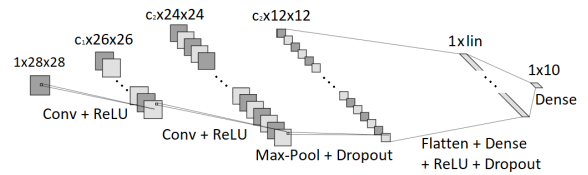


Figure 1: Applied CNN architecture.

This model was trained using the widely used Adam algorithm¹⁷ and the common negative log-likelihood loss

function¹⁸. The value of the learning rate was lr , the exponential decay was g and the step decay was $step$.

2.3. Tree-structured Parzen Estimator Algorithm

This iterative algorithm was developed 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 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. 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:

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

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. Every evaluation was run for 15 epochs with a batch size of 512. The search space where the optimal solution is sought is given in Table 1.

During the optimization I used the the first decile of loss values as threshold value, and 10-90, 20-80, 30-70, 40-60, 50-50, 60-40, 70-30, 80-20, 90-10, or 100-0 random initial evaluation-TPE evaluation ratio to examine the IB dependency. When TPE uses too few initial random iterations, it

Table 1: Ranges of the hyperparameters.

Hyperparameter	Minimum value	Maximum value
c_1	24	48
c_2	64	96
d_1	0.1	0.5
d_2	0.25	0.7
lin	128	192
lr	10^{-4}	10^{-2}
g	0.1	0.9
$step$	1	5

may not have enough data to accurately model the function being optimized, which can lead to suboptimal results. On the other hand, using too many initial random iterations may take up a part too big of the computational budget and not leave enough resources for the BO. So, it is important to strike a balance between random initial evaluations and TPE runs. That is why this examination is important.

3. Results

In this paper I use Accuracy, Precision, Recall, F1 Score, Receiver Operating Characteristic (ROC) and the area under the ROC curve (AUROC) for performance evaluation. Accuracy, Precision, Recall, and F1 Score are evaluation metrics used to measure the performance of a classifier on a classification task. They are calculated based on the number of true positive, false positive, true negative and false negative predictions made by the classifier. The ROC curve is a graphical representation of the trade-off between true positive rate (TPR) and false positive rate (FPR). It is often used to evaluate the performance of a binary classifier. AUROC is a metric that quantifies the overall performance of a classifier based on the ROC curve. It is calculated by integrating the curve between 0 and 1 with different threshold values. In multi-class classification, these metrics can be calculated for each class separately, or they can be averaged across all classes, as it is done in this paper, to get the overall evaluation of the classifier's performance. Further detailed description of these metrics is well summarized by Fawcett¹⁹.

3.1. Comparison of Different IB Choices

During the HPO TPE used the Accuracy, which is the most used metric of the MNIST dataset, to find the optimal CNN architecture. It is shown in Figure 2 for all the budget scenarios. It is clear that the best accuracies are achieved with a random initial evaluation-TPE evaluation ratio of 20-80 and 30-70. The best hyperparameter configuration at the 20-80 ratio is given in Table 2.

This 20-80 set also outperforms its competitors in terms of Precision, Recall, and F1 Score metrics, although the 30-70 set performs the best in terms of AUROC value. This

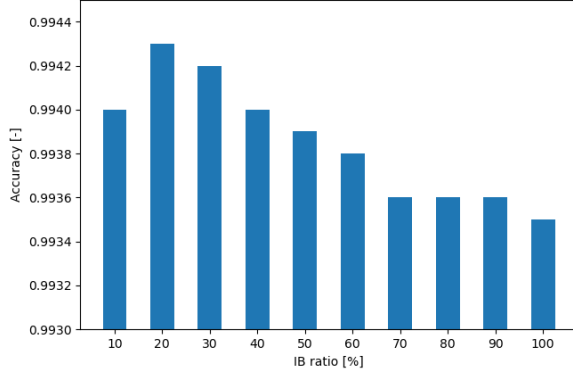


Figure 2: TPE-optimized CNN accuracies at different IBs

Table 2: Best hyperparameter configuration.

Hyperparameter	Value
c_1	33
c_2	80
d_1	0.42674887062658584
d_2	0.3989921607853992
lin	139
lr	0.0024478035602169294
g	0.5659849851416996
$step$	5

can be seen in Figure 3, where, similar to Figure 2's Accuracy value, Precision, Recall, and F1 Score are optimal at the 20-80 ratio, while the best AUROC result is observed at the 30-70 ratio. The effectiveness with which the various

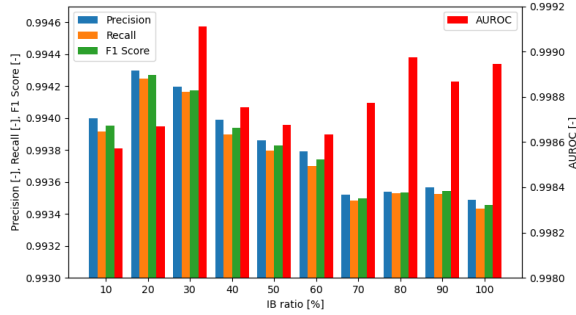


Figure 3: Precision, Recall, F1 Score and AUROC values at different IBs

TPE-optimized classifiers with different IBs classify the different classes, as well as the types of errors they commit, can be interpreted using their confusion matrices. The confusion matrices of the classifiers with ratios of 10-90, 20-80, 30-70, 40-60, 50-50, 60-40, 70-30, 80-20, 90-10, and 100-0 are shown in Figures 4 through 13, respectively.

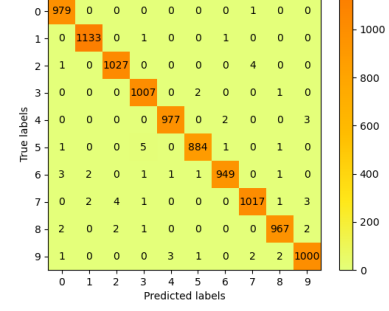


Figure 4: Confusion matrix of the 10-90 ratio

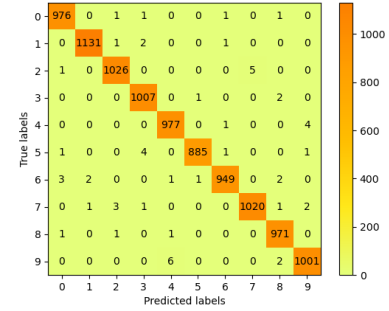


Figure 5: Confusion matrix of the 20-80 ratio

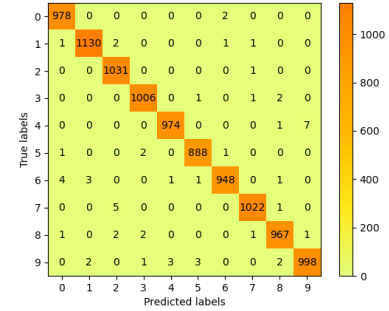


Figure 6: Confusion matrix of the 30-70 ratio

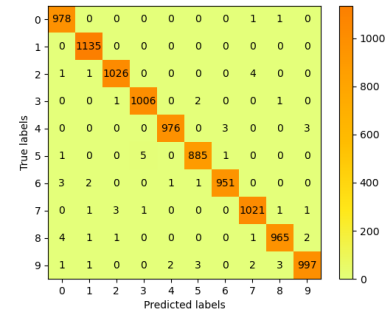


Figure 7: Confusion matrix of the 40-60 ratio

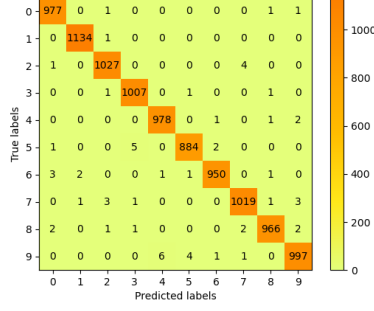


Figure 8: Confusion matrix of the 50-50 ratio

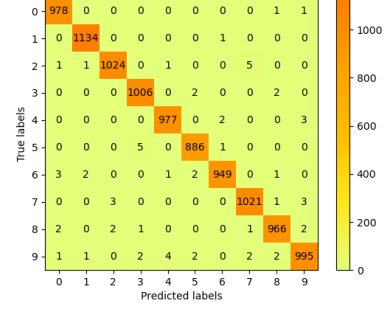


Figure 12: Confusion matrix of the 90-10 ratio

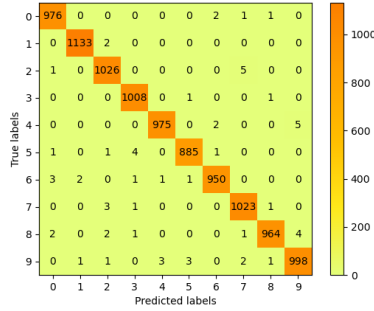


Figure 9: Confusion matrix of the 60-40 ratio

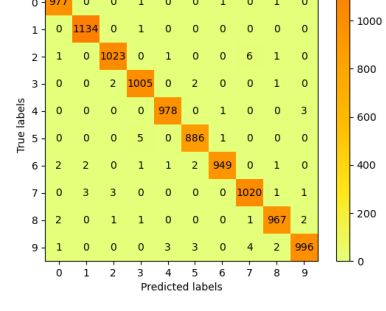


Figure 13: Confusion matrix of the 100-0 ratio

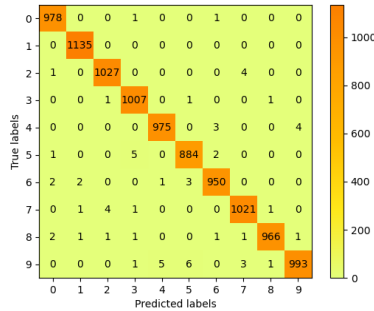


Figure 10: Confusion matrix of the 70-30 ratio

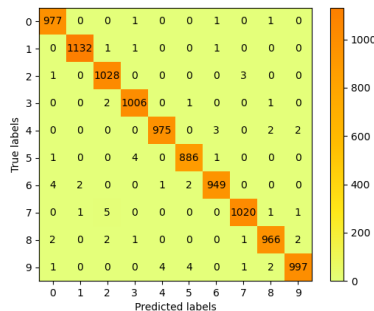


Figure 11: Confusion matrix of the 80-20 ratio

The shown classifiers mostly confuse similar numbers, which is the main source of their few errors. Therefore, it is worth observing the number pairs 4-9, 2-7, 3-5, 6-0, 8-0, 8-9, 7-9 and 5-9 in Figures 4 through 13. These 16 types of error source account for 60.3% of all errors for all classifiers. Therefore, 371 of the total 615 errors, which can be classified into 90 sources, occur in these typical pairs. The best classifier with an IB of 20% primarily has error sources of type 2-7, 3-5, and 4-9 among these types of error sources. Similarly, 2-7 and 4-9 error sources are characteristics of the second-best classifier with 30% IB, but 6-0 error source is more significant than 3-5.

3.2. Results of the 20% IB case

In Subsection 3.1, it was seen that the TPE optimized classifier that uses a 20% budget for initialization gives the best results. The training and testing results of this classifier's 15 epoch long training are presented in this Subsection. To do this, Loss, Accuracy, Precision, Recall, F1 Score, and AUROC values are used in Figures 14 through 19 respectively.

Figure 14 shows that the negative log-likelihood loss values are similar in magnitude for training and testing, although there is minimal overfitting in the system. This can be seen from the fact that the testing values are slightly bigger than the training values. The effects of step decay are also clearly visible on the testing data set, as the loss decreases at the fifth and tenth values.

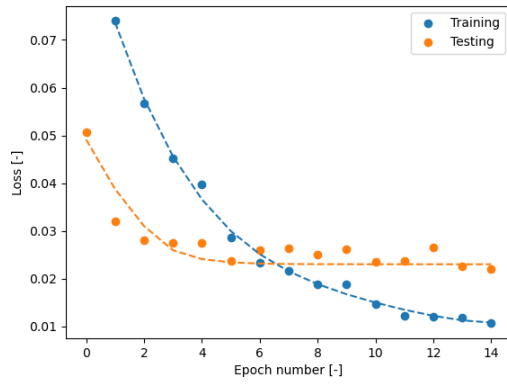


Figure 14: Loss values of the best CNN

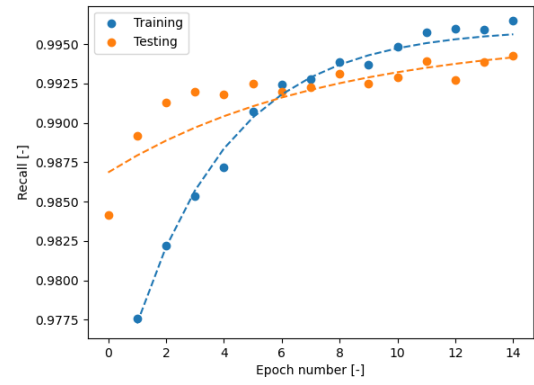


Figure 17: Recall values of the best CNN

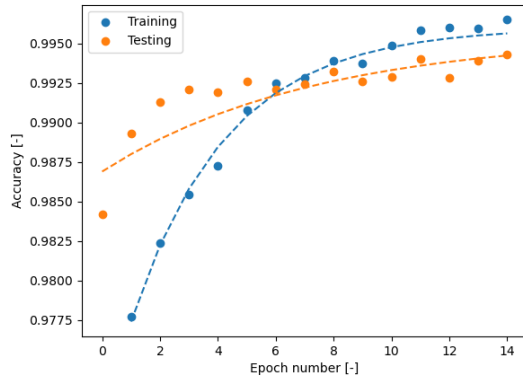


Figure 15: Accuracy values of the best CNN

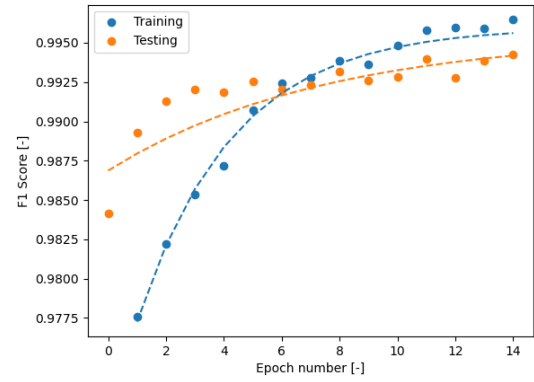


Figure 18: F1 Score values of the best CNN

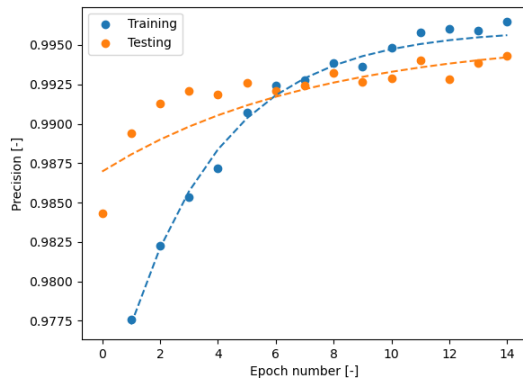


Figure 16: Precision values of the best CNN

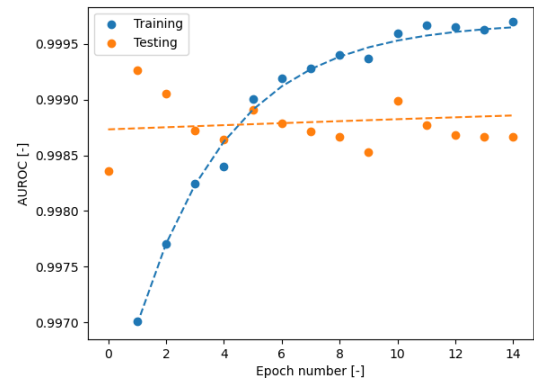


Figure 19: AUROC values of the best CNN

Accuracy, Precision, Recall, and F1 Score values are quite similar in Figures 15, 16, 17, and 18. This is not really surprising, as there is a significant correlation between these metrics. Here, the training and testing results are similar as well, but the former is slightly better.

The AUROC values of the testing data set on Figure 19 do not show the same asymptotic convergence as the previous Loss, Accuracy, Precision, Recall, and F1 Score values. The effect of step decay is also significant here, as it results in an improvement as can be seen at the fifth and tenth data points. However, a slight decrease in the AUROC values can be observed thereafter. The ROCs corresponding to the last data points can be seen on Figures 20 and 21. As can be seen, the ROCs are shown separately for each class both for training and testing. Only the range of TPR above 0.95 and FPR below 0.03 is shown for the sake of better visibility.

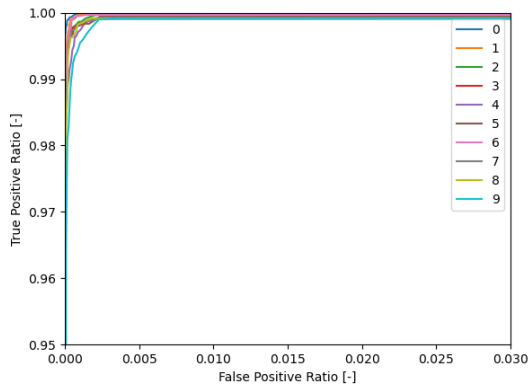


Figure 20: Training ROC values

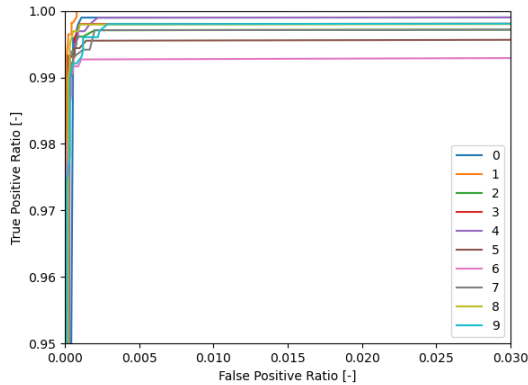


Figure 21: Testing ROC values

4. Conclusion

In this work, after reviewing the modern HPO options and CNN HPO options, including TPE HPO, the optimization of TPE for IB was examined. It is shown in this paper that if TPE has too low IB ratio, it finds a local optimum in the

hyperparameter space. However, a too high ratio of IB consumes too many resources and the probability-based optimization does not reach the optimum. So, it is crucial to find a balance between random initial evaluations and TPE runs.

According to the first table of hyperparameters, I optimized a CNN for the MNIST dataset using a batch size of 512 and 15 epochs. I varied the IB-TPE budget ratio between 10-90 and 100-0 with a step of 10. The total amount of resources was 100 runs in every case. Out of all the options, 20% IB turned out to be the best and 30% IB was the second best. The former was the best in terms of Accuracy, Precision, Recall, and F1 Score, while the latter was the best in terms of AUROC. From the confusion matrices of the 10 options, it was found that the better alternatives (20% IB, 30% IB) do not show any significant differences in the distribution of errors between classes compared to the worse alternatives. So, their performance is consistently better.

The best alternative was evaluated separately as well. Asymptotic convergence characterized the Loss (negative log likelihood), Accuracy, Precision, Recall, and F1 Score, but this was not the case for AUROC. Due to the minimal overfitting, the noticeable effect of step decay, and the nature of AUROC, it may be worthwhile to conduct these studies with larger epoch numbers, different batch sizes, and more resources in the future. However, the solution presented here is still included in the top 50 classifier architectures according to Papers With Code for MNIST²⁰, despite the fact that the aim of this work is to study the TPE IB optimization, rather than creating the best architecture. Therefore, this architecture can be further optimized.

Future research will focus on HPO using more epochs and resources, as well as working with different batch sizes. After that, I am going to examine the IB of TPE on different datasets as well.

References

1. Zhu, Y., Li, G., Wang, R., Tang, S., Su, H. & Cao, K. Intelligent fault diagnosis of hydraulic piston pump combining improved LeNet-5 and PSO hyperparameter optimization. *Applied Acoustics*. **183** pp. 108336 (2021)
2. Silva, P., Lucas, P., Sadaei, H. & Guimarães, F. Distributed Evolutionary Hyperparameter Optimization for Fuzzy Time Series. *IEEE Transactions On Network And Service Management*. **17**, 1309-1321 (2020)
3. Nakane, T., Bold, N., Sun, H., Lu, X., Akashi, T. & Zhang, C. Application of evolutionary and swarm optimization in computer vision: a literature survey. *IPSI Transactions On Computer Vision And Applications*. **12**, 1-34 (2020)
4. Tang, S., Zhu, Y. & Yuan, S. Intelligent fault diagnosis of hydraulic piston pump based on deep learning and Bayesian optimization. *ISA Transactions*. (2022)

5. Yang, T., Li, B. & Xun, Q. LSTM-Attention-Embedding Model-Based Day-Ahead Prediction of Photovoltaic Power Output Using Bayesian Optimization. *IEEE Access*. **7** pp. 171471-171484 (2019)
6. Bergstra, J., Yamins, D. & Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. *International Conference On Machine Learning*. pp. 115-123 (2013)
7. Bergstra, J., Bardenet, R., Bengio, Y. & Kégl, B. Algorithms for hyper-parameter optimization. *Advances In Neural Information Processing Systems*. **24** (2011)
8. Watanabe, S., Awad, N., Onishi, M. & Hutter, F. Multi-objective Tree-structured Parzen Estimator Meets Meta-learning. *ArXiv Preprint ArXiv:2212.06751*. (2022)
9. Ozaki, Y., Tanigaki, Y., Watanabe, S. & Onishi, M. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. *Proceedings Of The 2020 Genetic And Evolutionary Computation Conference*. pp. 533-541 (2020)
10. Nguyen, H., Liu, J. & Zio, E. A long-term prediction approach based on long short-term memory neural networks with automatic parameter optimization by Tree-structured Parzen Estimator and applied to time-series data of NPP steam generators. *Applied Soft Computing*. **89** pp. 106116 (2020)
11. Deng, L. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]. *IEEE Signal Processing Magazine*. **29**, 141-142 (2012)
12. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proceedings Of The IEEE*. **86**, 2278-2324 (1998)
13. Krizhevsky, A., Sutskever, I. & Hinton, G. Imagenet classification with deep convolutional neural networks. *Communications Of The ACM*. **60**, 84-90 (2017)
14. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*. (2014)
15. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 770-778 (2016)
16. Agarap, A. Deep learning using rectified linear units (relu). *ArXiv Preprint ArXiv:1803.08375*. (2018)
17. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*. (2014)
18. Bishop, C. & Nasrabadi, N. Pattern recognition and machine learning. (Springer, 2006)
19. Fawcett, T. An introduction to ROC analysis. *Pattern Recognition Letters*. **27**, 861-874 (2006)
20. Image Classification on MNIST, <https://paperswithcode.com/sota/image-classification-on-mnist>, Accessed: 2022-12-31