TECHNICAL REPORT

Hyperparameter Optimization for improving ML algorithms in Industrial Applications

This is a technical report that draws insights from recent work at RISE and explains the concepts for a wider audience.

Kateryna Mishchenko, Senior Researcher, Mathematical Optimization and Applied AI.

kateryna.mishchenko@ri.se

Machine learning (ML) algorithms are widely used in many applications domains, including advertising, recommendation systems, computer vision, natural language processing, and user behaviour analytics. Industrial applications, as industrial automation is one of the domains where ML and AI in general are taking an increasing role. Algorithms for data collection, preprocessing and analysis, data-based modelling, model prediction, maintenance, image recognition, anomaly detection as well as many other tasks relay on ML, optimization and control. These algorithms build and process extremely complex highly parametrized models and systems operating in some cases in real time. This leads to the necessity of implementation of efficient and robust algorithms applicable in wide range of real-life industrial applications.

In general, building an effective machine learning model is a complex and time-consuming process that involves detecting the appropriate algorithm and obtaining an optimal model architecture by tuning its hyperparameters, see (Li Yang, 2020).

Hyperparameters refer to parameters that cannot be updated during the algorithm execution, thus they needed to be defined once and used unchanged during training.

To build an optimal ML model, a range of possible hyperparameters must be explored. The process of designing the ideal model architecture with an optimal hyperparameter configuration is named hyperparameter tuning. Tuning hyperparameters is considered as key component of building an effective ML model, especially for tree-based ML models and deep neural (DL) networks, which have many hyper-parameters. Manual testing is a traditional way to tune hyper-parameters, although it requires a deep understanding of the used ML algorithms and their hyper-parameter value settings, see (Li Yang, 2020). However, manual tuning is ineffective for many problems due to certain factors, including multiple hyper-parameters, complex models, time-consuming model evaluations, and non-linear hyper-parameter interactions. These factors have inspired increased research in techniques for automatic optimization of hyper-parameters; so-called hyper-parameter optimization (HPO).

In short, the main purposes of HPO are:

- lower threshold of Research and Development.
- improve accuracy and efficiency of neural network training.
- make choice of hyper-parameters automated and convincing.
- and training results to be reproducible.

The problem of HPO has a long history, see (Hutter, 2019), and it was also established early that different hyperparameter configurations tend to work best for different datasets (Kohavi, 1995) (Ekman, 2013, v 11). In contrast, it is a rather new insight that HPO can be used to adapt general-purpose pipelines to specific application domains (Escalante, 2009). Example below illustrates one specific application HPO to speech recognition problem.

Speech Emotion Recognition Problem

To illustrate how HPO can boost performance of the algorithm for speech emotion recognition (SER). This is a typical example of classification ML problems; here human emotions should be classified. According to (Ekman, 2013, v 11), (David McCandless, 2022) most existing SER systems are based upon discrete emotional models and main emotions are categorized into six categories: sadness, happiness, fear, anger, disgust, and surprise.

Thus, the classification problem is stated as to categorize audio recorded speech into these six categories. Input data are recorded audio signals which include the target speaker's speech, background noise, and non-target speakers' voices.

Generally, SER systems consist of three components: speech signal acquisition, feature extraction, and classifiers to identify emotions. Traditional ML approaches for SER are applied after extracting the desired features from the speech signals. A variety of ML algorithms are applicable here, but deep learning (DL) classifiers are considered most suitable for emotion recognition because of its flexibility, scalability, and ease of learning.

Hyperparameter Optimization for Speech Emotion Recognition

In recent years, ML applications have exhibited a dramatic increase in privacy breaches, with personal data stored across an array of devices and analyzed by advanced technologies (David McCandless, 2022) .Federated Learning (FL) has been proposed recently and emerged as a technological solution to alleviate the privacy issue (McMahan, 2017).

FL is widely used in a variety of applications, particularly when dealing with privately identifiable information. Using the FL technique, the user can train ML models on their own devices without sharing raw data.

In application to the SER problem, by sending local parameters instead of raw video or image data, FL can protect users' privacy when dealing with fatigue detection tasks. In addition, the relatively smaller size of the local parameters compared to raw data can reduce communication costs and save network bandwidth.

As mentioned above, the goal of HPO is to develop an automated choice of the optimal hyperparameters improving performance of the model under consideration. In this study, the accuracy and CPU time of the SER model implemented in FL framework are optimized.

In particular, the first problem is to maximize the accuracy of the FL model. As optimization variables, number of global and local epochs, learning rate, batch size, division factor. This problem is constrained by the limits on the variables. This is nonconvex nonlinear mixed integer programming problem (MINLP) formulated as below:

$$\max_{s.t} Accuracy(LR, LE, GE, BS, CD)$$

$$LR = \{0.1, 0.01, 0.001\}$$
$$LE = \{1,3,5\}$$
$$GE = \{50,100,150,200\}$$
$$BS = \{8,16,32,64\}$$
$$CD = \{0.1,0.5,0.75,1\}$$

The second HPO is the problem of minimization of the CPU time for running the FL problem, its' statement is very similar to the problem above.

There are multiple ways of solving this kind of problems depending on the properties of the objective function and constraint set. In this case the brute force is used, where the objective function is evaluated on all points of the multidimensional grid based on all possible combinations of variables.

Table 1 below shows optimal results obtained by solving two HPO problems along with the standard setup used for training the FL model before the HPO was applied.

Table 1. Results from HPO

	Accuracy	CPU Time	Global Epochs	Local Epochs	Learning Rate	Batch Size	Client Division
Initial Setup	0.751	351.66	200	3	0.001	32	1.0
Optimal setup: Max Accuracy	**0.770**	96.98	50	5	0.01	16	0.50
Optimal Setup: Min CPU	0.729	**25.55**	50	1	0.001	16	0.10

Line 1 of Table 1 shows the vales of all parameters in the standard setup, Line 2 show the optimal values of parameters which are attained when solving the problem of maximizing accuracy. Line 3 shows the results solving the problem of minimizing the CPU time calculations.

Columns 2 and 3 show the values of objective functions, the rest of columns are due to the values of parameters of the FL NN problem which were used as optimization variables.

As row 2 says, the optimal accuracy is 0.770 against 0.751 used in the standard setup. Additionally, even the CPU time is shorter than in the standard setup.

Minimal CPU time, see row 3 of Table 1, is 25.55 sec against the 351.66 for the standard setup.

Except the optimal values of objective functions, it is important to find out and analyze model parameters that influence the performance of the model at most. In this case, the learning rate, LR parameter is the most significant parameter influencing the accuracy. This parameter defines the length of the step at each step of the DL algorithm.

Figure 1 below shows values of accuracy depending on the learning rate.

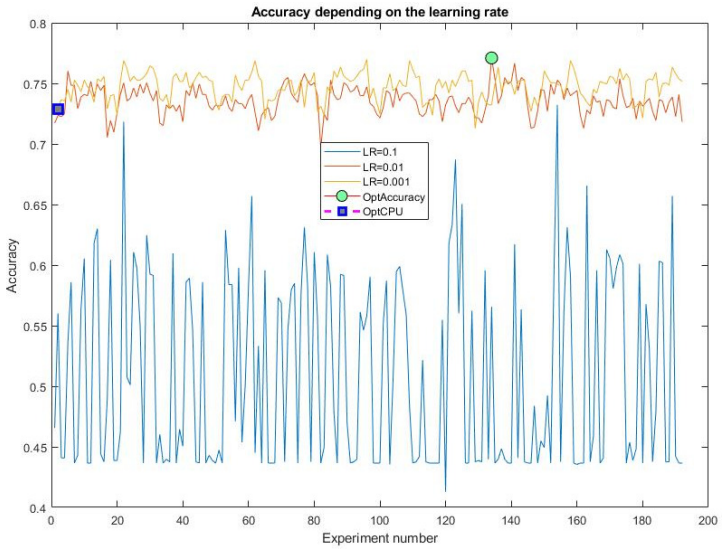

Figure 1.Accuracy of the model

Three learning rates define distinct accuracy curves calculated for all possible combinations of parameters. It is clear from this figure that LR, learning rate = 0.1 delivers the worst accuracy for the whole model irrespectively to other model parameters. Thus, by choosing LR equal to 0.01 or 0.001 the accuracy of the model is improved a lot.

Similarly, CPU time depends a lot on the client division factor, so that lowest value corresponds to the shortest CPU time, see Figure 2.

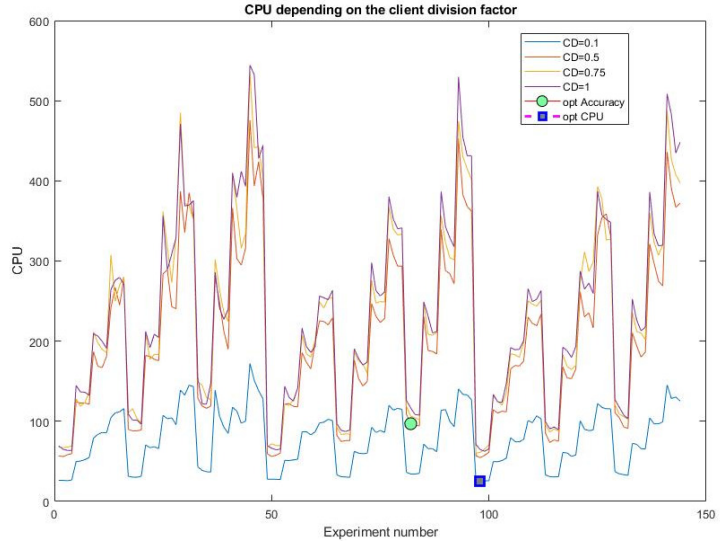

Figure 2.CPU time of the model

These results show clearly advantages of tuning hyperparameters by solving optimization problems, since optimal parameters setup allows to improve the performance of the FL model both in terms of accuracy and computational time. HPO problem is solved just once and defines optimal values of parameters included in the model. Thus, model design and network training could be enhanced a lot by using proper model parameters while running FL problem.

This simple example illustrates the impact of introducing HPO, which is the preprocessing step, taken before the learning process and model desing begin. Here, only five hyperparameters are optimized, while the number of them could be much higher which means that manual tuning will be even more complex and time consuming. Usage of HPO for optimal tuning of multiple model parameters helps to automate the procedure of creating of optimal models as well as for testing and investiagtion of models parameters. This will save both time and computationa resoucres and introduce standartization, so that researchers working with the same model will use teh same automated procedure of tuning complex DL models.

This approach could be extended further by taking into account more parameters. Another direction is to introduce mutiojective optimization to optimize several critaria at once , not only only accuracy or CPU time.

References

David McCandless, T. E. (2022). *World's Biggest Data Breaches & Hacks.* Retrieved from https://informationisbeautiful.net/visualizations/worlds-biggest-data-breaches-hacks/.

Ekman, P. a. (2013, v 11). *Emotion in the human face: Guidelines for research and an integration of findings.* Elsevier.

Escalante, H. M. (2009). Particle Swarm Model Selection. *Journal of Machine Learning Research 10*, 405-440.

Hutter, M. F. (2019). Hyperparameter Optimization. In L. K. Frank Hutter, *Automated Machine Learning. Methods, Systems, Challenges* (pp. 3-34). Springer.

Kohavi, R. J. (1995). Automatic Parameter Selection by Minimizing Estimated Error. In: Prieditis, A., Russell, S. (eds.) Proceedings. *Proceedings of the Twelfth International Conference on Machine Learning,* (pp. 304–312). Morgan Kaufmann Publishers.

Li Yang, A. S. (2020). On hyperparameter optimization of machine learning algorithms:Theory and practice. *Neurocomputing, 415*, 295–316.

McMahan, B. a. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistic* (pp. 1273-1282). PMLR.

This is a technical report that draws insights from recent work at RISE and explains the concepts for a wider audience.

RISE – Research Institutes of Sweden
ri.se / info@ri.se /