Parameter Optimization of Deep Learning Models by Evolutionary Algorithms

Levente Pető
Department of Mechatronics, Optics and
4-6 Bertalan Lajos Street, 1111 Budapest, Hungary
e-mail: peto.levente6@gmail.com

János Botzheim
Department of Mechatronics, Optics and
4-6 Bertalan Lajos Street, 1111 Budapest, Hungary
e-mail: dr.janos.botzheim@ieee.org

***Abstract*—Deep learning is a very popular gradient based search technique nowadays. In this field of machine learning we usually apply neural networks with various structure. The algorithms of the deep learning techniques and the structure of the applied networks have several parameters that have a huge impact on the performance of the search technique. These parameters are called hyperparameters. The aim of our current research is to optimize these hyperparameters using evolutionary and swarm based optimization algorithms.**

***Index Terms*—evolutionary algorithms; swarm optimization; deep learning; hyperparameters**

I. Introduction

Nowadays, researchers try to trace back a lot of problems to a maximization or a minimization problem, because there are many existing techniques to approach these problems. Besides analytic solutions there is an increasing interest in applying search space mapping methods such as evolutionary algorithms and swarm optimization based techniques. In this research we investigate several techniques such as Genetic Algorithm (GA) [1], Bacterial Evolutionary Algorithm (BEA) [2], Differential Evolution (DE) [3], Invasive Weed Optimization (IWO) [4], Particle Swarm Optimization (PSO) [5] and Simplified Swarm Optimization (SSO) [6]. In our previous works we had some experiences with many of these algorithms, thus their adjustments and finding their parameter settings are not a difficult task.

In recent years the gradient based methods are applied in several fields of minimization, and they are a part of the techniques applied in deep learning as well [7]. The huge nets used by deep learning contain many parameters. These parameters belong to the structure of the net and to the various algorithms in it. The most important things for a right solution is the optimal choice of the parameters of the deep learning techniques, called hyperparameters. In most of the cases the adjustment of these hyperparameters is done by human experts based on mathematical considerations and experiments. One of the most widely used optimization technique in this field is the Bayesian optimization [8], which is a probabilistic model based approach for optimization.

In our work we try to optimize these hyperparameters by evolutionary and swarm based optimization algorithms. First we optimize a simple Multilayer Perceptron (MLP) [9] on the MNIST [10] dataset. Then we increase the difficulty of the task and we optimize a net with a structure similar to the VGGNets structure [11]. In this case the task is the classification of the Fashion-MNIST dataset [12].

The structure of this paper is as follows. In Section II the related works and applied methodologies are introduced. The hyperparameter optimization and experimental results are presented in Section III. In Section IV the proposed method is compared with Bayesian optimization. In Section V a real world application is discussed. Section VI draws conclusions and shows the future directions of this research.

II. Related Works

In this Section the related works, applied methodologies, and the investigated problems are presented.

A. Evolutionary Algorithms

There are several optimization methods inspired by processes in the nature. The advantage of these algorithms is their ability to solve and quasi-optimize problems with non-linear, high-dimensional, multi-modal, and discontinuous character. It has been shown that evolutionary algorithms are efficient tools for solving non-linear, multi-objective and constrained optimizations. These algorithms have the ability to explore large admissible spaces, without demanding the use of derivatives of the objective functions, such as the gradient-based training methods. Their principles are based on the search for a population of solutions, which tuning is done using mechanisms similar to biological recombination. In evolutionary algorithms the individuals are evaluated and ranked using the fitness function. Genetic algorithm [1] is one of the most well known evolutionary algorithms. When generating new individuals it applies the crossover and the mutation operators. Bacterial evolutionary algorithm [2] on the other hand, applies the bacterial mutation and the gene transfer operators. Usually, evolutionary operators can either add new members to the population (e.g. crossover in GA) or modify existing members (e.g bacterial mutation in BEA). One or both are the main components of these algorithms in different implementations. In Differential Evolution [3] the mutation operator is based on

978-1-7281-0968-8/19/$31.00 ©2019 IEEE

the difference of multiple randomly selected individuals of the population. The invasive weed optimization [4] simulates weed colonizing behavior. Every seed produces seeds depending on its fitness and the produced seeds are being randomly dispread over the search area.

B. Swarm Optimization

In swarm optimization methods we never add new members to the population, we only modify them. In these methods the individuals try to find better and better places by exploring their environment led by their own experiences and the experiences of the whole community. One of the most well known swarm optimization methods is the particle swarm optimization [5]. The particles move in the search space. They have a velocity vector, and they remember their personal best point (cognitive component) and the global best point of the whole swarm (social component) in the search space. A new velocity vector is composed based on these three components. In the case of simplified swarm optimization [6] immediately the new position vector is formed based on the different components.

C. Multilayer Perceptron

A multilayer perceptron is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation function distinguish MLP from a Perceptron. It can learn data that is not linearly separable [9].

D. VGGNet

VGGNet is invented by VGG (Visual Geometry Group) at University of Oxford [11]. Though VGGNet is the first runner-up, not the winner of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2014 in the classification task. This net is very well structured and separable, therefore we chose it for optimization.

E. MNIST, Fashion-MNIST

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems [10]. The black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. The MNIST database contains 60,000 training images and 10,000 testing images.

The Fashion-MNIST dataset was created by the e-commerce company Zalando and it contains fashion images instead of handwritten digits [12]. It shares the same image size and structure of training and testing splits. The Fashion-MNIST dataset is illustrated in Figure 1.

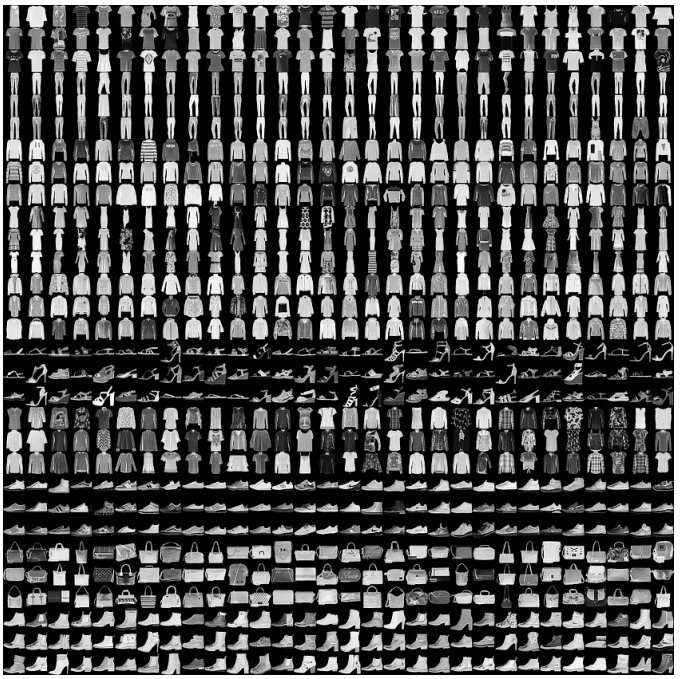

Fig. 1: Fashion-MNIST dataset

III. Hyperparameter Optimization

We implemented the evolutionary and swarm algorithms in a framework using Python language. One advantage of the evolutionary algorithms over Bayesian optimization is that they can be easily parallelized. We used this feature and applied four Nvidia GeForce GTX 1080 Ti for parallelly training the networks during the search processes. The values of the hyperparameters are in many cases integer numbers, however, in most of the evolutionary and swarm optimization algorithms the so-called genes are real numbers. Thus, before evaluating a candidate solution when creating a phenotype of the individual rounding down is performed.

A. MLP on MNIST

The first task is to optimize an MLP where the search algorithm can adjust the following hyperparameters:

- Number of hidden layers (NHL): 0 – 4
- Size of hidden layers (SOL): 5 – 1495
- Dropout (DO): 0 – 0.9
- Learning rate (LR): 0.0001 – 1

The parameters of the applied algorithms:

- DE: $F = 0.8$, $CR = 0.6$
- IWO: $iter_{max} = 100$, $\sigma_{init} = 0.18$, $\sigma_{fin} = 0.05$, $N_{min} = 1$, $N_{max} = 6$, $e = 2$
- PSO: $\omega = 1$, $\phi_p = 2$, $\phi_g = 4$
- SSO: $C_w = 0.2$, $C_p = 0.4$, $C_g = 0.8$

The population of the IWO algorithm contains 8 chromosomes (individuals), the other algorithms have 12 individuals. The neural network is trained by Adam optimizer [13]. In front of each layer we put a dropout layer, thus one chromosome contains 11 genes. Not all the elements of the chromosomes

were always used. In the case of the learning rate, the power of base 10 was changed between 0 and -4. Cross-entropy is used as the loss function. The training data were normalized between 0 and 1. There were no other preprocessing applied on the data. The training process was finished when the train loss did not improve during 14 epochs. The learning rate was decreased to its one fifth if there was no improvement during 5 epochs. The terminal criterion of the search algorithms is when the global best individual of the population does not improve during 10 iterations.

The fitness function is designed in that way that it punishes those neural networks which have more trainable parameters. Hence, the fitness function described in Equation (1) has two parts, the accuracy part and the parameters part.

$$accuracy\_part = 100 - (validation\_accuracy * 100)$$

$$parameters\_part = log_{10}(number\_of\_parameters)$$

$$fitness\_value = accuracy\_part + (parameters\_part/5) \quad (1)$$

The function can balance the two parts. It allows the addition of approximately one layer with 1% improvement.

The following four algorithms were applied for solving this task: PSO, SSO, IWO, DE. Table I presents the optimized parameters. Table II shows the result of the optimization: validation accuracy (ACC), number of parameters (NOP), number of evaluations (NOE), and fitness value (FV).

TABLE I: Optimized parameters of MLP

	NHL	SOL	DO	LR
DE	2	1297, 894	0.22, 0.38, 0.09	6.7e-4
IWO	1	661	0.31, 0.09	1.2e-3
PSO	1	1210	0.41, 0.03	1.4e-3
SSO	1	554	0.32, 0.02	1.5e-3

TABLE II: Optimization results of MLP

	ACC	NOP	NOE	FV
DE	99.01%	2 187 507	264	2.26
IWO	98.91%	525 505	321	2.23
PSO	98.87%	961 960	204	2.33
SSO	98.97%	440 440	492	2.16

These accuracies on test sets are near the results from the literature using similar networks without preprocessing data [10]. By using the proposed fitness function the best network was found by SSO, whose accuracy is a bit worse than that of the network found by DE, however its size is five times less. SSO had the biggest number of fitness calls. The structure of the best MLP is illustrated in Figure 2.

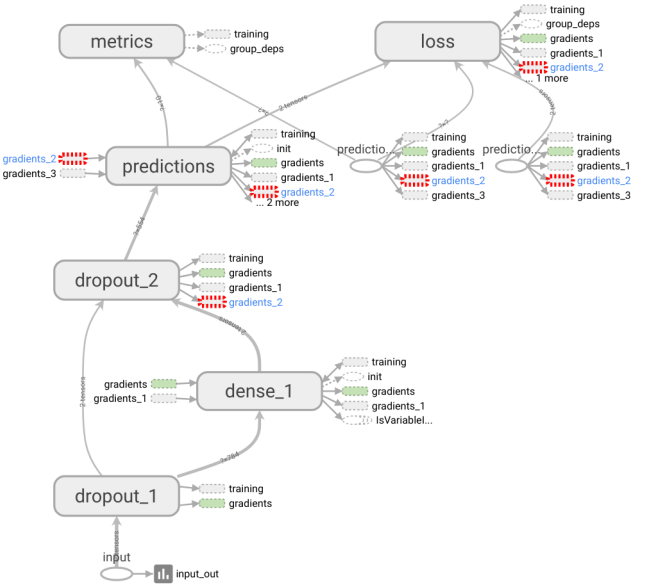

Fig. 2: Structure of optimized MLP

B. VGG on Fashion-MNIST

Now we investigate the main goal of the paper, the optimization of the structure of VGG16-like networks on the Fashion-MNIST dataset. The terminal conditions are similar as in the previous section with some changes. Here, the evolutionary and swarm algorithms global best individual has 5 iterations time to improve, and the patience during the neural network's training was initially 8 epochs followed by a refined search with 12 epochs.

The search algorithm can adjust the following hyperparameters:

- L1/L2 regularization weight decay (WD): 0.00001 – 0.1
- Dropout (DO): 0 – 1
- Learning rate (LR): 0.0001 – 1
- Number of convolution blocks (NB): 1 – 3
- Number of convolution layers in one block (NC): 1 – 6
- Number of fully connected layers (ND): 1 – 6
- Size of fully connected layer (SD): 1 – 501
- Size of convolution kernel (KS): 1 – 11
- Number of filters (F): 1 – 51
- Activation function (A): relu, sigmoid, tanh (th)
- Optimizer (O): adam, nadam, sgd, rmsprop

The best test results on Fashion-MNIST are around 95-96% in the literature, however they mostly applied preprocessing [12]. The human (non-expert) accuracy for this dataset is 83.5%. More precisely, the result in the literature of a VGG16 26M network is 93.5%. The only thing we changed on the data again is to normalizing it between 0 and 1.

The parameters of the applied evolutionary and swarm algorithms:

- GA: $p_{mut} = 0.7$
- BEA: $N_{clones} = 2$

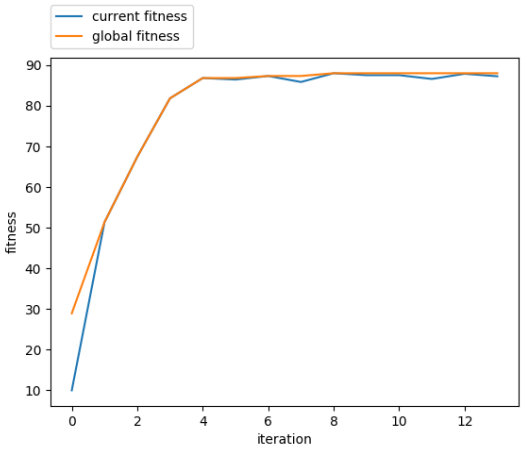

Fig. 3: Fitness value by SSO

- DE: $CR = 0.3$
- PSO: $\omega = 1.5$, $\phi_p = 2$, $\phi_g = 3$
- SSO: $C_w = 0.25$, $C_p = 0.5$, $C_g = 0.9$
- IWO: $iter_{max} = 10$, $\sigma_{init} = 32$, $\sigma_{fin} = 8$, $N_{min} = 1$, $N_{max} = 6$, $e = 2$

In this task we did not punish the bigger neural networks (i.e. the network with more trainable parameters) because here the original network is also big. Hence, the fitness function is the accuracy of the network calculated on the test set.

In the case of those algorithms where the expected number of fitness function calls is less, we increased the number of individuals (chromosomes) in order to achieve better results. Table III shows the obtained results: validation accuracy (ACC), number of chromosomes (NOC), number of iterations (NOI), number of evaluations (NOE).

TABLE III: Optimization results of VGG

	ACC	NOC	NOI	NOE
GA	90.76%	4	32	200
BEA	89.27%	4	11	503
DE	19.33%	6	11	78
PSO	62.83%	8	11	104
SSO	88.03%	8	14	128
IWO	90.74%	5	10	130

The above 90% accuracy obtained by GA and IWO are already acceptable result. Considering these results and the number of fitness function calls, the swarm based methods and GA seem promising methods to obtain an accurate result within reasonable time. For example, the simulation running by the SSO method is depicted in Figure 3. In the figure better fitness means that the value is closer to 100. The global best fitness is represented by orange color. This means the best solution found so far during the optimization process. On the other hand, blue color represents the best result of the current population. In the case of elitist algorithms these two functions are the same.

The obtained parameters by the best runs are presented in Tables IV and V.

TABLE IV: Optimized parameters of VGG (I)

	WD	DO	LR	NB	NC
GA	0.0095	0.656	0.0014	1	4
SSO	0.021	0.43	0.205	1	1
IWO	0.084	0.36	0.0001	3	2

TABLE V: Optimized parameters of VGG (II)

	ND	SD	KS	F	A	O
GA	4	186	9	45	relu	nadam
SSO	1	199	2	43	relu	rmsprop
IWO	1	175	10	11	th	nadam

Many parameter settings can be successful, because the numbers in the table have a big range.

Five algorithms are further investigated by running additional simulations. In the case of IWO the maximal number of seeds (individuals) is increased to 7. The ω parameter of PSO is decreased to 1, ϕ_g is increased to 6, the normalizer factor remained 5. The parameters of SSO, DE and GA were not changed. The obtained results are presented in Table VI, and the corresponding parameters are shown in Tables VII and VIII.

TABLE VI: Refined optimization results of VGG

	ACC	NOC	NOI	NOE	NOP
IWO	93.31%	6	22	419	490 671
SSO	93.57%	10	32	340	605 480
PSO	83.13%	10	13	150	657 342
GA	91.72%	8	6	94	1 757 546
DE	93.39%	6	19	114	1 791 818

TABLE VII: Refined optimization parameters of VGG (I)

	WD	DO	LR	NB	NC
IWO	0.0062	0.55	0.0768	3	2
SSO	0.0048	0.28	0.0383	2	3
PSO	0.011	0.46	0.621	2	1
GA	0.0753	0.27	0.0209	2	3
DE	2.41e-5	0.4	9e-4	3	3

PSO could improve, however, its final result is still not acceptable. The result of GA is obtained in the first generation, there was no improvement in the next 5 generations, so the search stopped. The result is not bad, but the corresponding network is too complex, it contains too many trainable parameters. The result of IWO looks like the original structure, with surprisingly few parameters. Its accuracy approaches the test accuracy of the VGGNet. The 93.57% accuracy of SSO is better than the test accuracy of the VGGNet, it uses a bit more parameters than the IWO and DE, however, its number of fitness function calls is less by almost 80. Besides, SSO and DE chose another path than IWO. They organized fewer

TABLE VIII: Refined optimization parameters of VGG (II)

	ND	SD	KS	F	A	O
IWO	1	103	3	39	relu	sgd
SSO	4	166	3	49	th	sgd
PSO	2	317	8	37	sigmoid	sgd
GA	2	466	8	36	th	sgd
DE	4	374	3	48	relu	adam

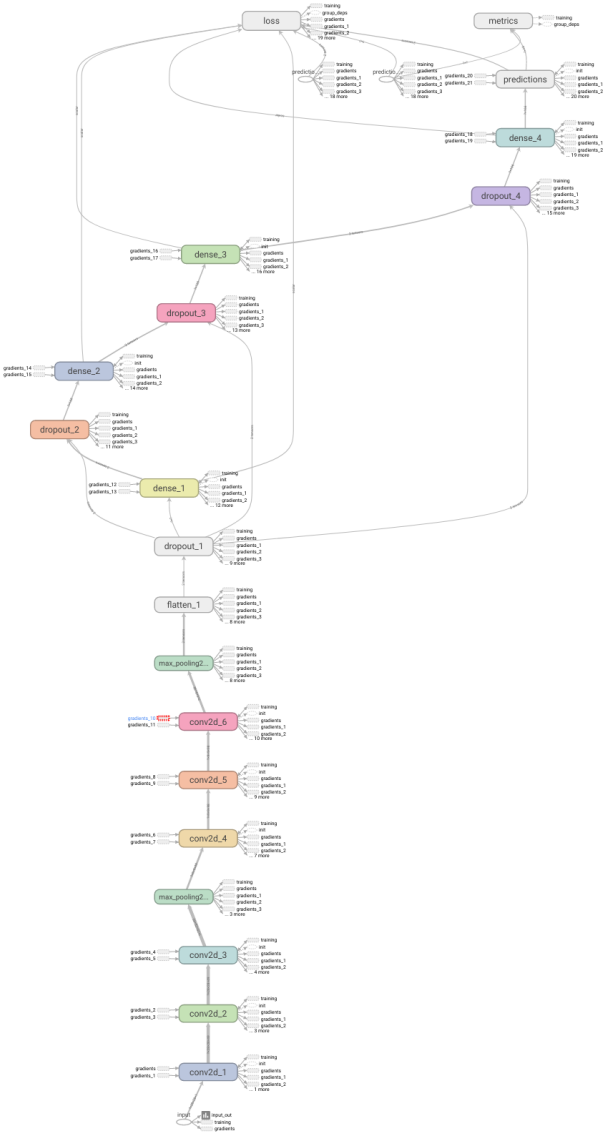

Fig. 4: Structure of optimized VGG

blocks with more convolutional layers and they put a much deeper fully connected network at the end of the structure. It is interesting that all solutions chose the "weakest" sgd optimizer, except DE. The structure of the best optimized VGG is illustrated in Figure 4.

The simulation results of the refined optimizations are illustrated in Figs. 5–7.

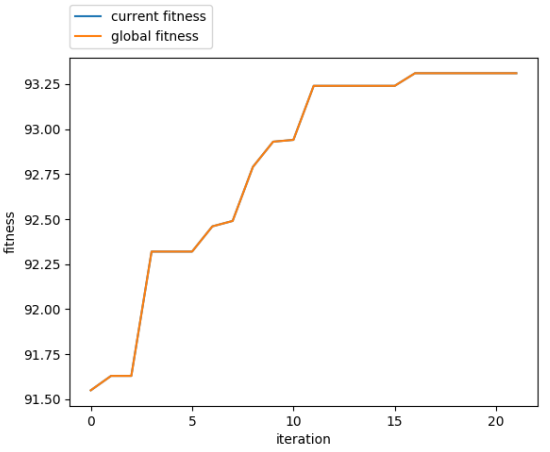

Fig. 5: Fitness value by IWO (refined optimization)

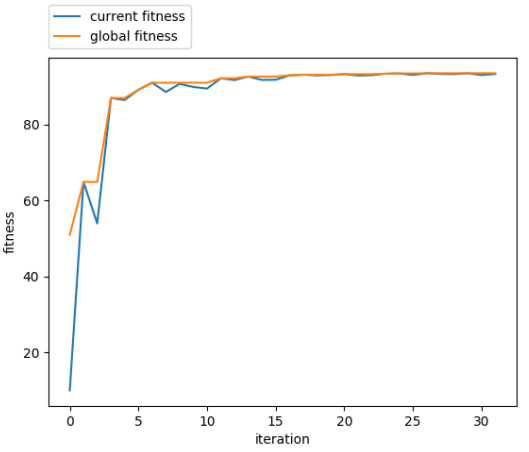

Fig. 6: Fitness value by SSO (refined optimization)

IV. Comparing with Bayesian Optimization

On the previous two problems Bayesian Optimization with Gaussian Processes was also executed. This method first evaluates initialization points and then it determines the next point to be evaluated in the search space by a Gaussian process. In both cases the algorithm was set to create 16 initialization points and in the usual way it stops when after 16 trials there is no improvement in the best fitness evaluation.

The optimization results of the best runs are shown in Table IX, Table X, Table XI, and Table XII.

TABLE IX: Bayesian optimization results

	ACC	NOE	NOP
MLP	98.79%	62	804 550
VGG	93.03%	40	1 030 923

The results are similar to that of the evolutionary algorithms. This method needed fewer evaluations, however, since it cannot be parallelized even in the case of more GPUs, thus in our case the running time was similar or little longer than in the case of evolutionary algorithms.

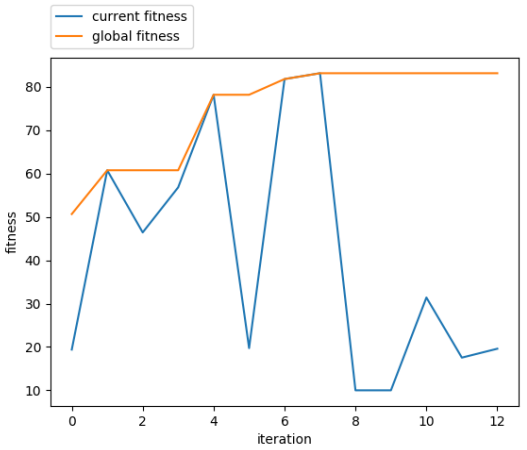

Fig. 7: Fitness value by PSO (refined optimization)

TABLE X: Bayesian optimized parameters of MLP

NHL	SOL	DO	LR
1	1012	0.259	0.005

V. Sleep Stage Classification from Single Channel EEG

Based on [14] we created a CNN-CNN model. This model applies a 1D CNN for the epoch encoding and then another 1D CNN for the sequence labeling. We optimized some parameters of this network using the best behaving SSO algorithm. The results of the original network is: F1 = 0.81, ACCURACY = 0.87. During the simulation the number of filters of convolution layers and the number of neurons of the last fully connected layer are changed. The original network has 250416 trainable parameters. The number of convolution filters is 16 in the first block, 32 in the second and third block, and 256 in the fourth block, and the last layer contains 64 neurons. The smallest network found by the simulation which has similarly good results as the original one has the following parameters: the number of filters in the blocks is 18, 29, 30, 212, respectively, and the number of fully connected neurons is 60. The sizes of convolutional filters were not changed. This network could reach a F1 = 0.805, ACCURACY = 0.855 result and it contains 178379 trainable parameters. This means that we could reach similar result by a significantly smaller network thanks to the SSO optimization.

VI. Conclusion and Future Work

VGGNet can contain depending on its settings even 144 million parameters. The obtained network by the SSO algorithm is closer to the optimal. Based on the result we recommend the SSO algorithm for hyperparameter optimization since it achieved the best result and it needs the same number of evaluations in each iteration as the population size. Similar number of evaluations was necessary in the case of Bayesian optimization. SSO can be ideally parallelized by setting the population size proportionally to the number of available GPUs, thus it can evaluate more solutions in the same time increasing the chance of finding better solutions in this way.

We have to note that each algorithm was executed only once, thus in the future the obtained results may be further improved. A future work is to execute the algorithms several times and to test them on even more complex tasks as well in order to explore their advantages and drawbacks.

TABLE XI: Bayesian optimization parameters of VGG (I)

WD	DO	LR	NB	NC
8.73e-5	0.475	5.7e-4	1	5

TABLE XII: Bayesian optimization parameters of VGG (II)

ND	SD	KS	F	A	O
2	359	4	29	th	nadam

Acknowledgments

Authors would like to thank Biot.ai lab Ltd to provide the resources needed for the simulations.

Supported by the ÚNKP-18-4 New National Excellence Program of the Ministry of Human Capacities.

References

[1] J. H. Holland, *Adaption in Natural and Artificial Systems*. Cambridge, Massachusetts: The MIT Press, 1992.

[2] N. E. Nawa and T. Furuhashi, "Fuzzy system parameters discovery by bacterial evolutionary algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 5, pp. 608–616, Oct. 1999.

[3] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[4] A. Mehrabian and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecological Informatics*, vol. 1, no. 4, pp. 355–366, 2006.

[5] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942–1948.

[6] C. Bae, W.-C. Yeh, N. Wahid, Y. Chung, and Y. Liu, "A new simplified swarm optimization (SSO) using exchange local search scheme," *Int. J. Innovative Computing, Information and Control*, vol. 8, no. 6, pp. 4391–4406, 2012.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[8] P. I. Frazier, "A tutorial on Bayesian optimization," www.arxiv.org/pdf/1807.02811.pdf, 2018.

[9] R. Hecht-Nielsen, *Neurocomputing*. Addison-Wesley, 1990.

[10] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[11] K. Simonyan and A. Zisserman, "VGGNet," www.arxiv.org/pdf/1409.1556.pdf, 2015.

[12] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST," www.arxiv.org/pdf/1708.07747.pdf, 2017.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," www.arxiv.org/pdf/1412.6980.pdf, 2014.

[14] A. Supratak, H. Dong, C. Wu, and Y. Guo, "Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, 2017.