Hyperparameters optimization for neural network training using Fractal Decomposition-based Algorithm

Léo Souquet
DSTI Labs
Data ScienceTech Institute
Paris, France
leo@dsti.co

Nadiya Shvai
National University of
Kyiv-Mohyla Academy
cyclope.ai
Paris, France
nadiya.shvai@cyclope.ai

Arcadi Llanza
cyclope.ai
Paris, France
arcadi.llanza@cyclope.ai

Amir Nakib
University Paris Est, Laboratoire LISSI
cyclope.ai
Paris, France
nakib@u-pec.fr

***Abstract*—This paper introduces the application of the fractal decomposition-based algorithm (FDA) to the optimization of the hyperparameters of deep neural network architecture. FDA is a metaheuristic that was recently proposed to solve high dimensional continuous optimization problems. In this work, we apply FDA to the optimization of well-known architectures such as VGG-16, NasNet, MobileNetV2 and ResNetV2-50. The hyperparameters of those architectures were fine-tuned using FDA on the CIFAR-10 benchmark dataset. The experiments demonstrate the superiority of proposed method over the state-of-art values. Considered approach shows promising results as every architecture was improved with hyperparameters found by using FDA. The experiments were conducted using low computational power with only 3 NVIDIA V100 GPUs, with 16GB of RAM.**

***Index Terms*—hyperparameters, optimization, deep learning, fractal decomposition**

I. Introduction

Deep Learning methods [1] have been drawn a lot of attention recently due to their efficiency in multiple fields, such as computer vision, speech recognition, natural language processing, bioinformatics, finance etc. They have been successfully applied to several difficult and important tasks, from healthcare [2] to computer vision for pedestrian tracking [3] or cybersecurity [4].

This success can be attributed to the capacity of deep learning algorithms to automatically extract features from data format such as audio, image or text (known commonly as unstructured data). These technics allow shifting from manual feature engineering where engineers spend time manipulating data sets and building meaningful new features to spending time on building deep neural network architectures and optimizing their hyperparameters. However the increased complexity of recent architectures such as AlexNet [5], VGG-16 [6] or GoogleNet [7] has motivated scientists and engineers to automate the optmimization of hyperparameters. Indeed, the training and fine tuning of architectures with hundreds of layers and millions of parameters is in practice very computational expansive and time consuming. Many different approaches have been studied to automate this task such as Grid Search or Random Search. The latter has shown to be very efficient when the number of parameters is not too large. Recently more complex hyperparameter optimization algorithms have been used such as Genetic Algorithms, Particle Swarm Optimization, Bayesian Optimization and Reinforcement learning.

In this paper we apply the metaheuristic ”Fractal Decomposition Algorithm” (FDA) [8] to the optimization of the hyperparameters of well known neural network architectures such as VGG-16 [9], Resnetv2-50 [10], NasNet [11] and MobileNetV2 [12]. Those architectures were fine-tuned on a given problem, CIFAR-10 [13] in this case, aiming to find the best validation accuracy possible. Results were compared with to parameters taken from the literature and found in the well known-knwon and wildely used deep learning framework Keras [14].

FDA is a decomposition based algorithm aiming to find the best solution possible. FDA uses hyperspheres as geometrical form to decompose the search space as this geometrical form scale well as the dimension of the problem increases. In the literature, hypercubes are more common but do not scale well due to the number of vertices increases exponentially. Besides, the fractal aspect of FDA is a reference to the fact that the search domain is decomposed using the same pattern at each level until the maximum fractal depth *k*. FDA has been designed to be easy to implement, deterministic and capable of solving black box optimization problems, i.e. problems where the algorithm does not assume any analytical form.

This paper is organized as follows. In section II we give the overview of the problematics. Section III shortly describes FDA algorithm. In section IV we go in detail over FDA application to the hyperparameters optimization problem for Neural Networks (NN). Section V contains presents the results of experiments. Finally, VI concludes the paper and discusses future work.

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

II. Related work

The selection of hyperparameters for a neural network architecture can be seen as an optimizaton problem where the objective is to find the set of parameters that minimizes a target function $\mathcal{L}(\mathcal{M}|\mathcal{X}^{(va)})$ given a model $\mathcal{M}$ on a validation set $\mathcal{X}^{(va)}$ trained on a training set $\mathcal{X}^{(tr)}$. The learning algorithm $\mathcal{A}$ can also be parametrized by a set of hyperparameters λ, $\mathcal{M} = \mathcal{A}\left(\mathcal{X}^{(tr)}|\lambda\right)$.

The hyperparameters search also referred as fine-tuning, consists of finding the set of hyperparameters λ^* such that the neural network architecture $\mathcal{M}$ minimizes the target function $\mathcal{L}(\mathcal{M}|\mathcal{X}^{(va)})$.

$$\lambda^* = \arg\min_{\lambda} \mathcal{L}(\mathcal{M}|\mathcal{X}^{(va)}) = \arg\min_{\lambda} f(\lambda; \mathcal{A}, \mathcal{X}^{(tr)}, \mathcal{X}^{(va)}, \mathcal{L}) \tag{1}$$

where f is the objective function and λ the hyperparameters [15]. Target function $\mathcal{L}$ is commonly the loss function, accuracy, or any other goodness-of-fit metric of the model. The learning algorithm $\mathcal{A}$, the target function $\mathcal{L}$, as well as the $\mathcal{X}^{(tr)}$ and $\mathcal{X}^{(va)}$ are known. Fine-tuning of hyperparameters of learning algorithms is indeed hard as the gradients are usually not available [16]. Different methods can be found in the literature for hyperparameters optimization. Grid Search (GS) is among the well-known methods for fine-tuning. It consists of first selecting a range of values for every hyperparameter to explore is selected. Then the learning algorithm will be trained for every combination of the given range of values. While this method is simple, its complexity is high as the number of combination grows exponentially with the number of hyperparameters and their different possible values. This can be however addressed by parallelizing the different training but its efficiency remains limited as GS. In [17] authors used a Random Search(RS) and show both empirically and theoretically that randomly selecting values for the different hyperparameters is more efficient than GS. Besides, Random Search is as easy to implement as Grid Search and as easily parallelizable. However, similarly to GS, RS suffers from being non-adaptive, meaning that sets of hyperparameters to be evaluated are not selected using existing results. It is important to note that in the context of hyperparameter optimization for neural network architecture, the cost of evaluating the cost function $\mathcal{L}$ is high. This is why in [18], [19] authors have applied Bayesian Optimization to the fine-tuning of learning algorithms. In their work, the neural network architectures' generalization performance is modelled as a sample from a Gaussian process (GP). Resulting models outperformed architectures fine-tuned by domain experts.

In addition to Bayesian approaches, Metaheuristics have applied fine-tuning of hyperparameters. This is because this family of algorithms has shown to be performant in solving black-box optimization problems. Genetic Algorithms are among the most popular algorithms for optimization as they allow to direct the search in the hyperparameter space [20]. Evolutionary Algorithms have also been widely used in architecture search [21]–[23]. Other metaheuristics have been successfully applied to hyperparameter optimization such as Particle Swarm Optimization can be an effective tool for challenging datasets [24].

III. The Fractal Decomposition Algorithm

The *Fractal Decomposition Algorithm* [8] (FDA) a "Divide-and-conquer" based metaheuristic that has been designed to solve large-scale continuous optimization problems. The main principle of the approach consists of dividing the feasible search space into subregions with the same geometrical pattern. When designing FDA, the use of hyperspheres as an elementary geometric form was considered. This choice was motivated by their low complexity and flexibility to cover the search space. Indeed, other decomposing methods can be found in the literature such as FRACTOP [25] or the well-known DIRECT [26]. Those algorithms use hypercubes as the geometrical form to decompose the search space. In both cases, the distance from the center of the hyper-interval to the vertices of the hypercube is computed. However, the number of vertices increases exponentially as the problem dimension increases. Consequently, the performances of the algorithms decrease drastically, in terms of computation time and quality of the final solution.

While searching for the best solution possible, FDA builds a search tree of promising optimum areas of a depth k (called fractal depth), by dividing the search space recursively using geometrical hyperspheres. Three main phases (see Fig. 1) compose the algorithm. 1. Initialization detailed in Sub-Section III-A; 2. Exploration phase (in Sub-Section III-B); 3. Exploitation Phase (in Sub-Section III-C).

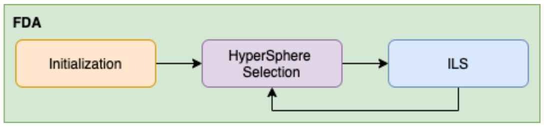

Fig. 1. FDA is composed by 3 main phases. First, initialization of the biggest hyperSphere. Then, exploration phase which aims to asses the quality and select the best hyperspheres sub-region. Finally, exploitation procedure using the local search ILS.

A. Initialization procedure

FDA starts by initializing, at level $l = 0$, the biggest hypersphere possible at the center of the search space within its limits as shown on Fig. 2. The center of the first hypersphere $\vec{C}^{(1)}$ and radius r are computed using the expressions (2) and (3), respectively.

$$\vec{C}_j^{(1)} = L + (U - L)/2, \text{ for } j = 1, 2, \ldots, D \tag{2}$$

$$r = (U - L)/2 \tag{3}$$

where $\vec{C}^{(1)}$ are the coordinates first hypersphere within the search space, D the dimension of the search space and r the hypersphere's radius. U is the upper bound, L is the lower bound of the whole search space.

Once the first hypersphere is created, it is partitioned into $2 \times D$ child-hyperspheres using the expression (4) and as shown on Fig. 2.

$$\vec{C}_k^{(i)} = \vec{C}_k^{(i)} + (-1)^i \times ((r - r') \times \vec{e}_k) \quad (4)$$

where $\vec{C}^{(i)}$ represents the center of the i_{th} child-hypersphere with $i = 1, \ldots, 2 \times D$, $r' = r/(1+\sqrt{2})$ and $\vec{e}_k$ the unit vector at the dimension k. Once the initialization is done, FDA moves on to the exploration phase.

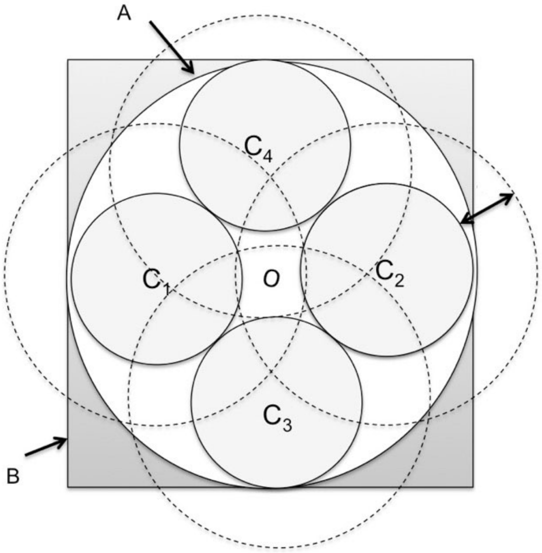

Fig. 2. Illustration of the decomposition procedure in the case of a 2D search space, where A is the biggest hypersphere inside the search space (B), C_1, C_2, C_3, and C_4 are centers of the child-hyperspheres created at the first level

B. Exploration procedure

The main objective of the exploration phase is to assess the quality of each sub-region aiming to detect the most promising hyperspheres to be further decomposed. Because child-hypersphere cannot cover te entier search space, child-hypersphere are inflated using a factor α. This increase produces overlaps between hyperspheres which allow covering the entire search space to be covered. The analytical and detailed explication on the value of α can be found in the original paper [8]. This procedure is called relaxation and is also illustrated in Fig. 2.

The quality of the child-hyperspheres are computed as per [8] and only the best one is selected to be further decomposed (using expression 4). FDA then moves down one level in the search tree ($l = l + 1$). This repetitive pattern at each level composes the fractal dimension of FDA.

Hyperspheres that have not been decomposed are sorted according to their quality and stored in a stack for further decomposition. If all hyperspheres at a level l have been explored, FDA selects the next one in the stack at level $l = l - 1$ to be partitioned. This would have the effect of creating a new branch in the search tree.

C. Exploitation procedure

One the maximum depth k is reached, FDA enters in the exploitation phase. The main objective is to find the best solution possible from the most promising identified sub-regions. To do so, FDA will trigger a local search aiming to exploit all generated child-hyperspheres at the k_{th} level. At this step, different learning-based optimization methods can be used. However, to maintain a low complexity of the approach a simple algorithm, called Intensive Local Search (ILS) was implemented.

For each child-hypersphere ILS starts at the center $\vec{C}$ with $\vec{x}^s$ and moves along each dimension with a step ω for which two solutions $\vec{x}^{s_1}$ and $\vec{x}^{s_2}$ are evaluated. Both solutions are expressed in (5) and (6), respectively.

$$\vec{x}^{s_1} = \vec{x}^s + \omega \times \vec{e}_i \quad (5)$$

$$\vec{x}^{s_2} = \vec{x}^s - \omega \times \vec{e}_i \quad (6)$$

where $\vec{e}_i$ is the unit vector where the i^{th} element is set to 1, and other elements to 0. ω is the step size in which $\vec{e}_i$ changes. At each step, the best solution among $\vec{x}^s$, $\vec{x}^{s_1}$ and $\vec{x}^{s_2}$ is chosen to be the next current solution $\vec{x}^s$, then ILS moves to the next dimension. If no improvement has been made for $\vec{x}^s$, then ω is reduced by factor $1/\varphi$. This local search stops when either the stopping criterion or threshold ω_{min} is reached. In this case, ω_{min} represents the tolerance or the desired precision of the problem being solved.

Once ILS stops for the current ILS, the best solution found locally,$\vec{x}^s$, is returned and the global solution is updated if an improvement has been made.

Once all hyperspheres at the $k - th$ level have been exploited, if the stopping criterion has not been yet met, then, FDA backtracks in the search tree, and the next hypersphere to be decomposed is selected at the level $l = l - 1$. This procedure allows the algorithm to explore other regions of the search space.

IV. Methodology

In this section, we describe in details the application of *FDA* algorithm to the problem of CNN training hyperparameters optimization (see Fig. 3).

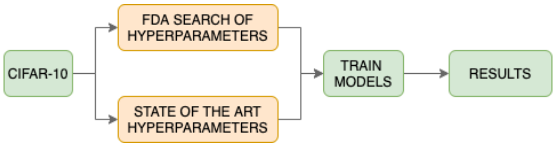
CIFAR-10
FDA SEARCH OF HYPERPARAMETERS
STATE OF THE ART HYPERPARAMETERS
TRAIN MODELS
RESULTS

Fig. 3. Comparison of multiple modules taken into account to benchmark FDA optimization method. In Green color are represented the modules that are common for both approaches. In Orange are highlighted the modules that are being compared.

A. CNN Architecture

For the experiments we consider four popular architectures: VGG16 [27], NASNetMobile [11], MobileNetV2 [28], ResNetv2-50 [29]. They were selected for their omni-availability and proven performance on different computer vision problems.

The convolutional part of these networks is used as a base, and a dense layer is added on top, followed by the predictions layer. The dense layer is surrounded by two dropout layers as shown in Fig. 4.

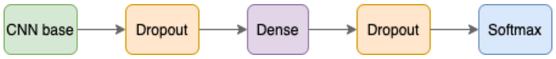

Fig. 4. Extra layers appended to the CNN base architecture.

The size of this added dense layer and the dropout rates are included in the list of optimized parameters (see subsection IV-B for more details).

B. Encoding of the problem

For optimization we have selected a set of training parameters: starting learning rate, momentum, dropout rates, period T for cosine annealing rate scheduler [30] (see Fig. 5) and additionally the size of the added dense layer. The range of the hyperparameters being optimized is given in Table I.

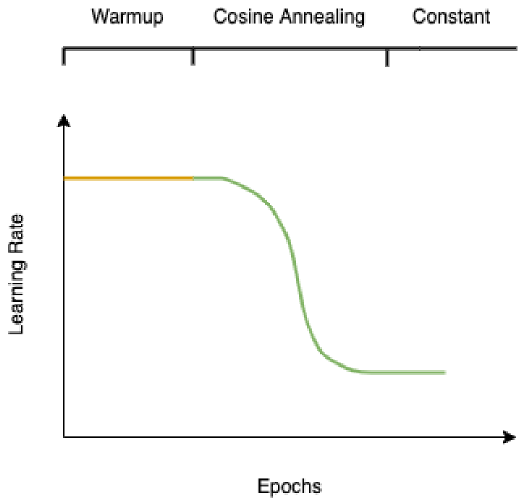

Fig. 5. Three main phases have been followed for the learning rate scheduler. First, a warm up period in order to easy the network the learning of some stable values. Second, cosine annealing learning rate to decrease quickly the learning rate when the network is in a plateau. Third, when the minimum amount of epochs is reached in the previous stage, the learning rate is kept as a constant.

V. Experiments and Discussion

Here we present the results of the CNN hyperparameters optimization by the means of FDA. The experiments were conducted on the CIFAR-10 dataset.

The implementation was done using Python as a programming language and the framework Keras [14] with Tensorflow [31] as backend. Experiments were done using only three NVIDIA V100 GPUs with 16GB of RAM.

TABLE I
Values range of the CNN hyperparameters that were optimized.

Parameters	Value Range
Learning Rate	$[0, 1]$
Momentum	$[0, 1]$
Batch Size	$[32, 512]$
Period T for cosine annealing LR scheduler	$[0, 400]$
Dropout rate 1	$[0, 1]$
Dropout rate 2	$[0, 1]$
Number of units in the dense layer	[0,4000]

A. CIFAR-10

This benchmark is composed of 60000 32×32 colour images divided into ten different classes. The training set is composed of 50000 images and the test set of 10000 images. The classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. The test set contains 5000 images of each class and the test set, 1000 images of each class. Classes are mutually exclusive no overlap exists between trucks and automobiles. Automobile only includes cars and assimilated. Truck includes only big trucks. Neither includes pickup trucks.

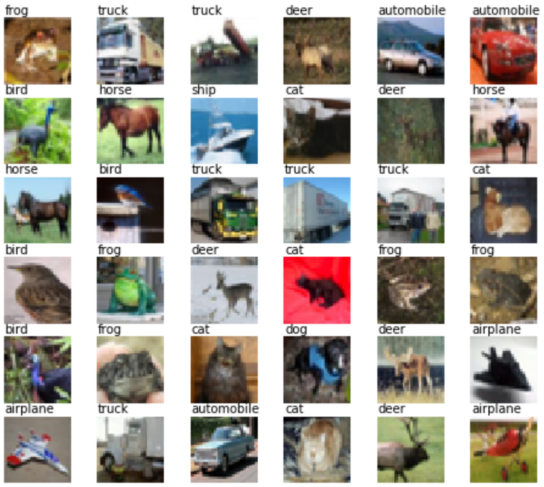

Fig. 6. Examples of images in CIFAR-10 with their class labels.

B. Training details

During the FDA phase, at each function evaluation (in our case, accuracy estimation) we train a CNN with given hyperparameters. The training is done using 80% of the initial training set for training, and 20% for validation. For the data augmentation (see Fig. 7), we apply color channel shift, random shift and horizontal flip. We also use mixup [32] with a fixed parameter $\alpha = 0.4$. Illustrations of aforementioned data augmentations techniques are given in Fig. 7.

Learning rate is decreased during the training with cosine annealing learning rate scheduler [30]. Period T of the cosine annealing learning rate scheduler is one of the parameters being optimized; after T epochs learning rate will not decrease any more. In the beginning of the training we have 5 ”warm-up” epochs, during which we do not decrease the learning

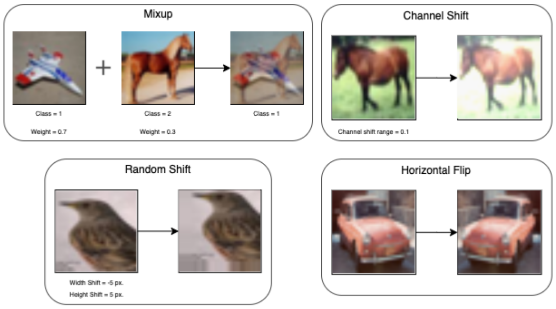

Fig. 7. Examples of the data augmentation techniques applied in CIFAR-10.

rate and do no use the mixup data augmentation. These ”warm-up” are used to quickly find a relatively good training point, and then continue training using a smaller learning rate (thus in a slower manner) and using more challenging data augmentation. Overall scheme of learning rate schedule is shown in Fig. 5.

For the final test, we have used all the 50k training data for the training, and have increased the number of epochs from 50 to 200. The period T for the cosine annealing learning rate scheduler is increased proportionally to the increase in number of epochs, i.e. 4 times. The final validation accuracies are obtained on the 10000 images composing the original test set of the benchmark, which are unseen data during the hyperparameter search done using FDA.

C. Results

In this subsection we discuss the experiment results and make the comparison to the baseline.

TABLE II
COMPARISON OF RESULTS FOR THE DIFFERENT BENCHMARKED ARCHITECTURES. TEST ERROR (%) (LESS IS BETTER).

	Benchmark	**FDA**
VGG16	8.85	7.44
NasNetMobile	20	18.5
MobileNetV2	16.77	15.34
ResnetV2-50	24.1	17.64

The hyperparameters found by FDA are given in Table III. It is interesting to observe that:

- Some hyperparameters, like learning rate, are in close range (from 0.003 to 0.006), while others change a lot from one network to another, like batch size (from 52 to 272) or period T (from 280 to 1220);
- In the best performing network, VGG16, found momentum value (0.87) is very close to the benchmark setting, 0.9;
- Period T of cosine annealing learning rate might be much larger than the number of epochs, notably it reaches 1220 for VGG16 (while number of epochs is 200). This corresponds to the very slow decrease of the learning rate over the training process;
- Found values of dense layer size (545 and 1002) are close to the ones that are commonly used: 512 and 1002.
- In contrast to the previous finding, dropout rate 1 for NASNetMobile (0.7) is considerably higher than ordinarily used values.

The change of validation accuracy during the optimization phase is illustrated on Fig. 8

Table II comparison results using the test error (%). For the benchmark we use the state-of-art values of hyperparameters found in the literature [27], [32], [33].

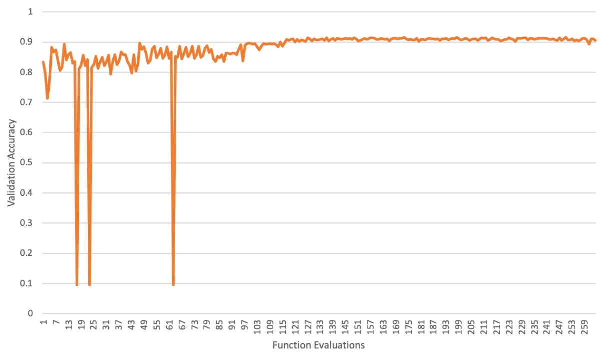

Fig. 8. Validation Accuracy of VGG-16 during the hyperparameters optimization using FDA

It can be observed that:

- FDA has shown improvement over baseline in all the experiments;
- Despite not achieving overall best result, Resnetv2-50 gets the biggest improvement with FDA;
- The rank of the networks in the table is the same for benchmark and FDA; thus, although potentially after the optimization the rank could have changed, initial benchmark run is a good indicator to eliminate poor candidates, and, consequently, to decrease the final optimization costs.

VI. CONCLUSIONS

In this work, we present the solution of NN hyperparameters optimization using Fractal Decomposition Algorithm. We evaluate this method on four common CNN architectures (VGG16, NASNetMobile, MobileNetV2 ResNetv2-50) with CIFAR-10 dataset. The experiment results show improvement over the baseline hyperparameters choice.

Proposed pipeline follows a relatively computationally cheap alternative to network architecture search: problem-based hyperparameters tuning in the range of potentially good network candidates. The experiments were conducted using only 3 NVIDIA V100 GPUs for a total computation time of 15 GPU-Days, thus being potentially suitable for the wide range of users. To further study FDA could be applied to handcrafted or automatically generated architectures and/or using other benchmarks such as CIFAR-100.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–44, 05 2015.

TABLE III
HYPERPARAMETERS FOUND BY FDA OPTIMIZATION FOR EACH CNN

	FDA				**Benchmark**
	VGG16	**NasNetMobile**	**MobileNetV2**	**Resnet50**	**All**
Learning Rate	0.006	0.006	0.003	0.0034	0.01
Momentum	0.87	0.74	0.76	0.5	0.9
Dropout rate 1	0.5	0.70	0.24	0.57	0.25
Dropout rate 2	0.39	0.17	0.5	0.26	0.5
Dense layer size	1002	545	1002	1002	4096
Period T	1220	280	804	384	200
Batch Size	84	52	272	146	128

[2] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature Medicine*, vol. 25, 01 2019.

[3] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing*, vol. 300, pp. 17 – 33, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S092523121830290X

[4] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security," in *2018 10th International Conference on Cyber Conflict (CyCon)*, May 2018, pp. 371–390.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 1–9.

[8] A. Nakib, S. Ouchraa, N. Shvai, L. Souquet, and E.-G. Talbi, "Deterministic metaheuristic based on fractal decomposition for large-scale optimization," *Applied Soft Computing*, vol. 61, no. Supplement C, pp. 468 – 485, 2017.

[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," vol. 9908, 10 2016, pp. 630–645.

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," 06 2018, pp. 4510–4520.

[13] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[14] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[15] M. Claesen, J. Simm, D. Popovic, Y. Moreau, and B. De Moor, "Easy hyperparameter search using optunity," 12 2014.

[16] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Gradient-based hyperparameter optimization through reversible learning," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15. JMLR.org, 2015, p. 2113–2122.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 1, p. 281–305, Feb. 2012.

[18] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *ICML*, 2013.

[19] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 2951–2959.

[20] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, ser. MLHPC '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2834892.2834896

[21] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: Neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19. New York, NY, USA: ACM, 2019, pp. 419–427. [Online]. Available: http://doi.acm.org/10.1145/3321707.3321729

[22] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, pp. 2902–2911. [Online]. Available: http://dl.acm.org/citation.cfm?id=3305890.3305981

[23] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17. New York, NY, USA: ACM, 2017, pp. 497–504. [Online]. Available: http://doi.acm.org/10.1145/3071178.3071229

[24] P. Ribalta Lorenzo, J. Nalepa, M. Kawulok, L. Sanchez, and J. Ranilla, "Particle swarm optimization for hyper-parameter selection in deep neural networks," 07 2017, pp. 481–488.

[25] M. Demirhan, L. Özdamar, L. Helvacğlu, and c. I. Birbil, "Fractop: A geometric partitioning metaheuristic for global optimization," *J. of Global Optimization*, vol. 14, no. 4, pp. 415–436, Jun. 1999.

[26] D. R. Jones, C. D. Perttunan, and B. E. Stuckman, "Lipshitzian optimization without the Lipshitz coefficient," *Journal of Optimization Theory Applications*, vol. 79, no. 1, pp. 157–181, 1993.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[30] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[32] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[33] Y. E. Nesterov, "A method for solving the convex programming problem with convergence rate $o(1/k^2)$," 1983.