E-ISSN 2581 – 7957
P-ISSN 2277 – 3916
CVR Journal of Science and Technology, Volume 23, December 2022
DOI: 10.32377/cvrjst2307

Hyper-Parameter Optimization using Metaheuristic Algorithms

D. Bhanu Prakash[1], K. Arun Kumar[2], and R. Prakash Kumar[3]
[1]Assoc. Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: pbhanududi@gmail.com
[2]Sr. Asst. Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: arun.katkoori@gmail.com
[3]Sr. Asst. Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: prakash.rachmagdu@gmail.com

Abstract: **Machine learning algorithms are widely used in various applications. To properly implement them, their hyper-parameters need to be tuned. It is often necessary to know the ins and outs of ML learning algorithms as well as the proper hyper-parameter techniques. This paper presents two metaheuristic algorithms namely, Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) that can be used to improve the performance of machine learning algorithms. In this paper, we evaluated optimized algorithms for various machine learning algorithms namely, K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF). For conducting experiments, we used four benchmark datasets namely, Breast cancer, Iris, Digits, Wine datasets from sklearn library are considered. Experimental results show that PSO is performed well for optimizing ML models based on large search space. And it is observed that Decision-Tree technique performed poorly for 'Digits' dataset.**

Index Terms: **Optimization, Machine-Learning Models, Hyper-Parameters, Genetic Algorithm, Particle Swarm Optimization.**

I. Introduction

Machine learning is a field of research that focuses on developing methods that can capture an element of interest in each data set [1]. This can be done by analyzing various components of a given data set and predicting their target values. ML algorithms are widely used in various industries such as advertising. They are typically built to perform complex and high-performance tasks [2][3].

There are two types of parameters that can be used in a ML model: the model parameter and the hyper-parameters. The model parameter can be initialized through the data learning process and can be updated through the data library [4]. Various learning methods are available. These include kernel methods, ensemble models, and biological inspired networks. One of the most common characteristics of these methods is their parameterization. Due to the nature of the search algorithms used for hyperparameters, their reproducibility is not ideal when large sets of hyperparameters are required [5]. Therefore, the idea of automated search is gaining increasing attention in various areas of machine learning. A key component of ML is choosing the appropriate complexity level for the model. Generally, if the model is complex, it should fit the data used to construct it well, but it should also not be complex.[6][7].Hyper-parameter Optimization is a process utilized for improving the efficiency of the tuning process of ML models. It is usually performed by carrying out a series of predefined steps to achieve the optimal model architecture [8]. Following are the reasons for applying Hyper-parameter Optimization to ML.

1. It allows developers to focus on their core algorithms instead of having to spend time tuning the hyper parameters.
2. It helps improve the performance of many ML models. There are many parameters that can affect the model's performance.
3. It makes the models more reproducible. Also, it helps to identify the most suitable algorithm for a particular problem.

It is important to select an optimal optimization technique that can identify optimal hyper-parameters. Usually, traditional techniques are not suitable for hyper-parameter optimization (HPO) problems. Other optimization techniques such as metaheuristics, decision-theoretic approaches, and Bayesian models are more suitable for optimizing these HPO problems [9]. They can detect continuous hyper-parameters and can also identify discrete and conditional hyper-parameters.

II. Hyper Parameters in Machine Learning

To boost the performance of ML models, we need to know what the key hyperparameters are to fit the models into specific problems. The supervised learning algorithms are usually focused on learning how to map input features of a target [10]. Some of the commonly used ones include K-Nearest Neighbors, linear models, and decision trees. Unsupervised learning methods are usually used to find unlabeled data. The importance of some of the hyper-parameters of common multi-language models are studied in Python libraries.

A. K-Nearest Neighbor (KNN)

K-Nearest Neighbor is a type of algorithm that classifies data points by their distance from one another. It does so by calculating the distance between the data points that belong to the given class.

Received on 26-09-2022, Revised on 15-10-2022, Accepted on 01-11-2022.

Assuming the training set,

$$T_r = \{ (p_1, q_1), (p_2, q_2), \dots (p_n, q_n) \} \qquad (1)$$

where Pi- Feature vector,
Qi-class of instance
i=1, 2….n.

for a test instance 'p', its class 'q' can be represented as

$$q = argmax \sum_{j=1}^{m} I(q_i = b_j), i = 1,2, \dots n\, ; j = 1,2, \dots m \qquad (2)$$

where I(p)- indicator function

I= 1, when q_i=b_j
= 0, otherwise.

M_k(p)- filed invoking the k-nearest neighbors of 'p'.

The number of nearest neighbors, k, is the most critical hyper-parameter in KNN [11]. It is often used to determine the model's fit.

B. Support Vector Machine (SVM)

Support vector machines (SVM) [12][13] are supervised learning algorithms that can be used for various tasks such as classification and regression. They map data points to high-dimensional space and partition them into segments [44].

Assuming there are 'm'data points, the objective function of SVM is:

$$argmin_z \{\frac{1}{m} \sum_{i=1}^{m} \quad max\ (0, 1 - q_i f(p_i) + \ Bz^T z) \qquad (3)$$

where the z-normalization vector.
B- Penalty parameter of the error.

B is an important Hyper parameter of SVM.

The function f(p) is a type of kernel function that measures the similarity between two parallel data points. It can be tuned through different types of kernel models.

The different kernel functions can be denoted as follows:

i) Linear kernel:

$$f(p) = p_i^T p_j \qquad (4)$$

ii) Polynomial kernel:

$$f(p) = (\gamma p_i^T p_j + \gamma)^e \qquad (5)$$

iii) RBF Kernel

$$f(p) = exp^{(-\gamma ||p - p'||^2)} \qquad (6)$$

iv) Sigmoid kernel

$$f(p) = tanh(\gamma p_i^T p_j + \gamma) \qquad (7)$$

A kernel type can also have a conditional hyper-parameter called 'gamma'. This is the conditional hyper-parameter that is set when the type is specified as a polynomial or sigmoid kernel. The kernel type hyper-parameters are tuned after a kernel is chosen.

C. Decision Trees

A decision tree is a commonly used classification method that shows a set of classification rules that are computed from the data [14]. It has three main components: a tree's root node, which represents the whole data, multiple decision nodes, and leaf nodes that represent the result classes [15].

Other important factors that can be tuned to make decision tree models perform well include the quality of splits, their random selection method, and the number of features that can be considered to generate the best split. The number of features that can be considered for generating the best split, or 'max features', can be tuned as part of the feature selection process.

D. Random Forest

Random forest is a ML algorithm that combines multiple classifiers to solve a variety of problems. It can be commonly used for both classification and regression problems [16].

A random forest is a type of classifier that takes multiple decision trees and outputs a final output with the most accurate predictions. Instead of relying on one tree, it uses most votes from the trees to predict the final output.

Table I shows various hyper-parameters, type, and search space for different ML classifiers

TABLE I.
SUMMARY OF HYPER-PARAMETERS IN DIFFERENT ML MODELS

S. No	Machine Learning Classifier	Hyper-Parameter	Type	Search Space
1	Random Forest Classifier	n_estimators	Discrete	[10 ,100]
		Max_Depth	Discrete	[5,50]
		Min_samples split	Discrete	[2,11]
		Min_samples leaf	Discrete	[1,11]
		Criterion	Categorical	['gini','entropy']
2	Support Vector Machine	C	continuous	[0.1,50]
		Kernel	Categorical	['linear','poly''rbf','sigmoid']
3	KNN Classifier	N_ eighbours	Discrete	[1,20]
4	Decision Tree classifier	Cp	Discrete	[0,1]
		Max depth	Discrete	[1,30]
		Min split	Discrete	[1,60]

III. HYPER PARAMETERS OPTIMIZATION TECHNIQUES

Population-based optimization is a type of metaheuristic algorithm that starts by creating a population as each generation. It then updates the population as each generation is evaluated. The main differences between various Population-based methods are that they use different methods to generate and select populations. Population-based algorithms are easily parallelized since they can be evaluated on a swarm of 'N individuals.

A. GA-Genetic Algorithm

It is a widely used metaheuristic model that studies how individuals adapt to the environment and how likely they are to survive in the future [17]. The next generation will also have the same parents' characteristics. This means that they will either have better or worse individuals. The former will become more adaptable and resilient, while the latter will gradually disappear. Each chromosome has a hyper-parameter, which is the actual input value of that hyper-parameter in an evaluation. The population consists of all possible values within the predefined parameters.

Since the random-generated parameter values do not contain the optimal values, several operations involving selection, crossover, mutation, and selection operations are performed on the healthy chromosomes to identify the optimal. Chromosome selection aims to increase the population size by selecting the best possible chromosomes. Chromosome selection is a process that selects good characteristics for later generations. This process involves swapping a proportion of genes in each of the chromosomes [18].

Steps of GA are as follows:

1. Intricate representations of the entire search space are provided. Randomly initialize the population, genes, and chromosomes.
2. The fitness function is a calculation that shows the objective of a model.
3. Initiate selection, crossover, or mutation operations on the lysed chromosomes to produce a new generation of hyper-parameter configurations.
4. Repeat steps 2 and 3 until the 'termination' condition was met.
5. Terminate and output the 'optimal hyper-parameter' confirmation.

Population initialization is an important step in the optimization process and can significantly improve the performance of parameter optimization algorithms. A good initial population should have individuals with global optimums who can cover promising regions. Random initialization commonly used in GA is a good alternative to good initialization. It allows the configuration of hyper-parameter candidates without missing the global optimum. The GA algorithm can be useful when the data analyst has little experience in defining the appropriate search space for the various hyper-parameters. The time complexity of GA is O(n^2). This algorithm can be inefficient when used with low convergence speed.

B. PSO-Particle Swarm Optimization

It is a type of evolutionary algorithm that is inspired by biological phenomena. They use the same principles to solve problems involving large-scale networks. Unlike GA, PSO doesn't require crossover or mutation [2].

Instead, all members of the population share information with each other, which enables them to move toward the optimal region. PSO has computational complexity that consists of O (nlogn). In most cases, its convergence speed is faster than that of GA.

PSO is only capable of reaching a local level if it has the proper population initialization. This means that developers should have experience in implementing population initialization techniques and implementing global optimums. Many population initialization techniques are proposed to improve the efficiency of evolutionary algorithms. However, these methods require many resources and time to perform their intended function.

Metaheuristic algorithms such as PSO and GA are more complex than other high-pressure algorithms, but they usually perform well in complex optimization problems.[19] PSO is good for large-scale parallelization and is usually preferred over GA for complex HPO problems. It is also faster than GA due to its sequential execution. To properly identify a local, population initialization is very important for PSO. It can slow down or only identify a local instead of a global optimum.

IV. Experimental Results

Datasets:

For experimental setup, we considered four standard benchmark datasets from sklearn library. Namely, Breast cancer dataset, Iris dataset, Digits dataset, and wine dataset. The summary of the dataset is shown in the table below.

TABLE II.
Summary of Datasets Used

Dataset	Classes	Samples per class	Total samples	Dimensionality
Breast cancer	2	212-Malignant 357-Benign	569	30
Iris	3	50-setosa 50-versicolor 50-verginica	150	4
Digits	10	0 to 9	1797	64
Wine	3	59-class 0 71- class 1 48-class 2	178	13

All experiments are conducted using Googlecolab. Colab is a product from Google Research that allows anyone to write and execute Python code through the browser. It is a great tool for machine learning, data analysis, and education.

The first step is training and evaluating the model with its default hyperparameter confirmation. The second step is implementing the two algorithms i.e., GA and PSO to evaluate and compare the model's performance.

TABLE III.
PERFORMANCE EVOLUTION OF APPLYING HPO METHODS TO THE DIFFERENT CLASSIFIERS ON BREAST CANCER DATASET

SI No	Classifiers	Optimization Method	Accuracy (%)	Evaluation time (msec)
1	Random Forest Classifier	Default Hyper-parameters	92.6	1.13
		Genetic Algorithm	96.66	58.2
		PSO	97.66	45.25
2	Support Vector Machine	Default Hyper-parameters	95.25	22.15
		Genetic Algorithm	95.6	598.49
		PSO	96.6	359.4
3	KNN Classifier	Default Hyper-parameters	92.4	0.06
		Genetic Algorithm	93.32	2.15
		PSO	94.33	2.92
4	Decision Tree classifier	Default Hyper-parameters	91.91	0.04
		Genetic Algorithm	92.79	2.73
		PSO	93.79	3.31

TABLE IV.
PERFORMANCE EVOLUTION OF APPLYING HPO METHODS TO THE DIFFERENT CLASSIFIERS ON IRIS DATASET

S I No	Classifiers	Optimization Method	Accuracy (%)	Evaluation time(m sec)
1	Random Forest Classifier	Default Hyper-parameters	96	0.92
		Genetic Algorithm	96.66	40.64
		PSO	96.6	28.8
2	Support Vector Machine	Default Hyper-parameters	97.33	0.01
		Genetic Algorithm	98.00	0.67
		PSO	98.66	0.03
3	KNN Classifier	Default Hyper-parameters	97.0	0.02
		Genetic Algorithm	98.00	0.69
		PSO	98.0	1.01
4	Decision Tree classifier	Default Hyper-parameters	96.66	0.01
		Genetic Algorithm	96.66	0.72
		PSO	97.33	0.69

Tables III to VI, we can see that using default hyper-parameter configurations, do not yield best model performance in our experiments, which emphasizes the importance of utilizing optimization methods. From the above tables, we can conclude that PSO is better than GA.

TABLE V.
PERFORMANCE EVOLUTION OF APPLYING HPO METHODS TO THE DIFFERENT CLASSIFIERS ON DIGITS DATASET

SI No	Classifiers	Optimization Method	Accuracy (%)	Evaluation time(m sec)
1	Random Forest Classifier	Default Hyper-parameters	93.15	1.78
		Genetic Algorithm	93.6	97.12
		PSO	93.32	80.51
2	Support Vector Machine	Default Hyper-parameters	94.765	0.17
		Genetic Algorithm	97.44	13.68
		PSO	97.38	7.39
3	KNN Classifier	Default Hyper-parameters	95.93	0.2
		Genetic Algorithm	96.66	6.61
		PSO	96.6	10.22
4	Decision Tree classifier	Default Hyper-parameters	77.8	0.09
		Genetic Algorithm	78.57	5.34
		PSO	79.02	6.39

TABLE VI.
PERFORMANCE EVOLUTION OF APPLYING HPO METHODS TO THE DIFFERENT CLASSIFIERS ON WINE DATASET

SI No	Classifiers	Optimization Method	Accuracy (%)	Evaluation time (m sec)
1	Random Forest Classifier	Default Hyper-parameters	97.7	78.6
		Genetic Algorithm	98.31	46.4
		PSO	98.33	45.21
2	Support Vector Machine	Default Hyper-parameters	95.11	0.9
		Genetic Algorithm	96.66	5.94
		PSO	96.11	9.87
3	KNN Classifier	Default Hyper-parameters	69.14	0.01
		Genetic Algorithm	72.53	1.15
		PSO	72.53	1.27
4	Decision Tree classifier	Default Hyper-parameters	89.31	0.01
		Genetic Algorithm	91.04	0.08
		PSO	90.48	0.89

TABLE VII.
OPTIMIZED HYPER-PARAMETERS OF TESTED CLASSIFIERS USING GA

SI No	Machine Learning Classifier	Hyper-Parameter	Optimized Values for Breast cancer dataset	Optimized Values for Iris dataset	Optimized Values for Digits dataset	Optimized Values for Wine dataset
1	Random Forest Classifier	n_estimators	96	64	98	44
		Max_Depth	9	9	9	7
		Min_samples split	2	2	2	10
		Min_samples leaf	1	4	1	1
		Criterion	Entropy	Entropy	Entropy	Entropy
2	Support Vector Machine	C	49	17	7	2
		Kernel	Linear	Rbf	Rbf	Linear
3	KNN Classifier	N_ neighbours	13	7	3	50
4	Decision Tree classifier	Cp	0	0	0	0
		Max depth	2	9	21	4
		Min split	57	18	8	38

TABLE VIII.
OPTIMIZED HYPER-PARAMETERS OF TESTED CLASSIFIERS USING PSO

SL No	Machine Learning Classifier	Hyper-Parameter	Optimized Values for Breast cancer dataset	Optimized Values for Iris dataset	Optimized Values for Digits dataset	Optimized Values for Wine dataset
1	Random Forest Classifier	n_estimators	46	60	54	85
		Max_Depth	5	8	9	6
		Min_samples split	6	8	10	2
		Min_samples leaf	2	6	3	1
		Criterion	Gini	Gini	Entropy	Gini
2	Support Vector Machine	C	49	6	17	49
		Kernel	Linear	Rbf	Rbf	Linear
3	KNN Classifier	N_ Neighbours	12	11	3	50
4	Decision Tree classifier	Cp	0	0	0	0
		Max depth	2	3	27	27
		Min split	5	6	5	13

Below figures 1 to 4 shows the accuracies of different classifiers with respect to default hyper parameters (without optimization) and optimized hyper parameters (using GA and PSO). It is observed that PSO is performing well in most of the cases.

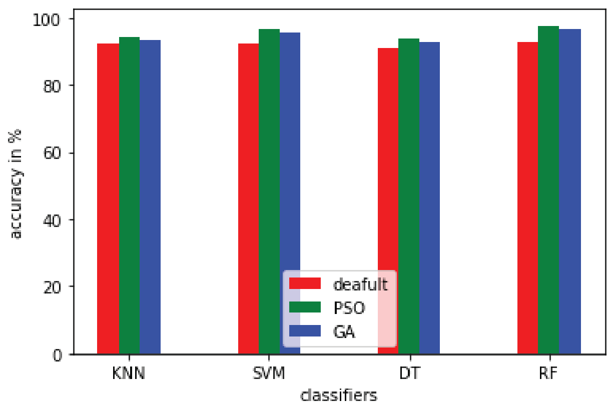

Figure 1. Breast Cancer Dataset

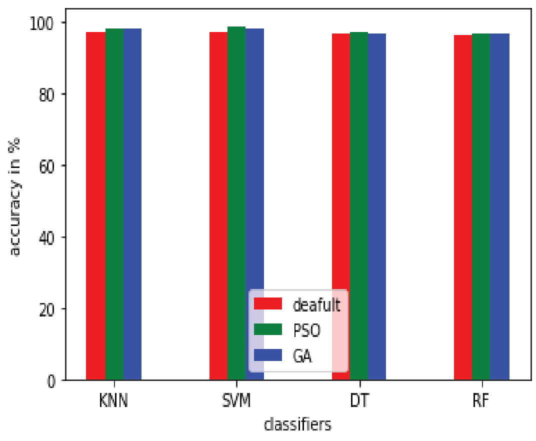

Figure 2. Iris dataset

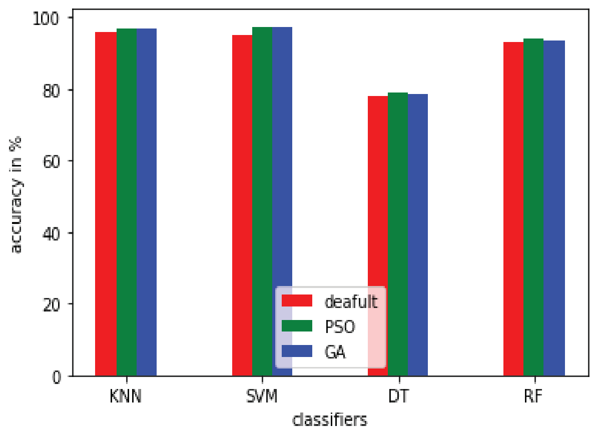

Figure 3. Digits dataset

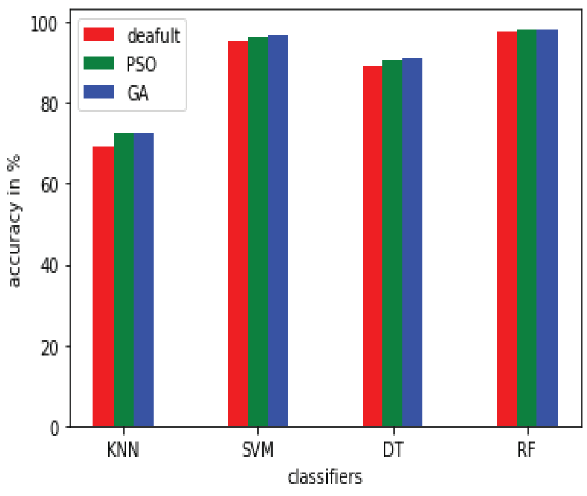

Figure 4. Wine dataset

V. Conclusions

Machine learning has become the main strategy for solving data-related problems. Its goal is to identify the most suitable hyper-parameters for the given problem. In this work, we comprehensively discussed different optimization algorithms as well as how to apply them to different ML models. To summarize PSO is performing better for optimizing ML models based on large search space. From our experiments, it is observed Decision-Tree performed poorly for Digits dataset.

References

[1] Dudi, B., Rajesh, V. “Optimized threshold-based convolutional neural network for plant leaf classification: a challenge towards untrained data”. *Journal of Combinatorial Optimization,Springer,* 2021.
[2] Katukuri Arun Kumar, Ravi Boda, A Multi-Objective Randomly Updated Beetle Swarm and Multi-Verse Optimization for Brain Tumor Segmentation and Classification, *The Computer Journal*, 2021.
[3] M.I. Jordan, T.M. Mitchell, “Machine learning: Trends,perspectives, and prospects”, Science 349, 255260, 2015.
[4] M. Kuhn and K. Johnson, “Applied Predictive Modeling”, Springer, ISBN: 9781461468493, 2013.
[5] F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., “Automatic MachineLearning: Methods, Systems, Challenges”, Springer ISBN:9783030053185, 2019.
[6] N. Decastro-Garca, . L. MuozCastaeda, D. Escudero Garca, and M. V.Carriegos, “Effect of the Sampling of a Dataset in the HyperparameterOptimization Phase over the Efficiency of a Machine Learning Algorithm,Complexity”, 2019.
[7] S. Abreu, “Automated Architecture Design for Deep NeuralNetworks”, arXiv preprint arXiv:1908.10714,2019.
[8] O. S. Steinholtz, “A Comparative Study of Black-box Optimization Algorithms for Tuning of Hyper-parameters in Deep Neural Networks”, M.S. thesis, Dept. Elect. Eng., Lule Univ. Technol., 2018.
[9] S. Lessmann, R. Stahlbock, S.F. Crone, “Optimizing hyperparameters ofsupport vector machines by genetic algorithms”, Proc. Int. Conf.Artif.Intell. ICAI05. 1, 2005.
[10] R. Caruana, A. Niculescu-Mizil, “An empirical comparison of supervisedlearning algorithms”, ACM Int. Conf. Proceeding Ser. 148, 161168, 2006.

[11] W. Zuo, D. Zhang, K. Wang, “On kernel difference-weighted k- nearestneighbor classification”, Pattern Anal. Appl. 11, 247257, 2008.

[12] Dudi, Bhanuprakash, and V. Rajesh. "Medicinal plant recognition based on CNN and machine learning." *International Journal of Advanced Trends in Computer Science and Engineering* 8.4 (2019): 999-1003

[13] Mahesh Kusuma K. Arun Kumar, P. Rajashekar Reddy,”Medical Image Classification Based On Normalized Coding Network with Multiscale Perception”,International Journal of Innovative Technology and Exploring Engineering,vol.8,issue 11,2019.

[14] S. Rasoul, L. David, “A Survey of Decision Tree Classifier Methodology”,IEEE Trans. Syst. Man. Cybern. 21, 660674, 1991.

[15] L. Yang, R. Muresan, A. Al-Dweik, L.J. Hadjileontiadis, “Image-Based Visibility Estimation Algorithm for Intelligent Transportation Systems”, IEEE Access. 6, 7672876740,2018.

[16] Palimkar, Prajyot, Rabindra Nath Shaw, and Ankush Ghosh. "Machine Learning Technique to Prognosis Diabetes Disease: Random Forest Classifier Approach." *Advanced Computing and Intelligent Technologies*. Springer, Singapore, 2022. 219-244.

[17] D.M. Manias, M. Jammal, H. Hawilo, A. Shami, P. Heidari,A. Larabi, R. Brunner, “Machine Learning for Performance-aware Virtual Network Function Placement”,IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc. 1217, 2019.

[18] A. Gogna, A. Tayal, Metaheuristics: Review and application, J. Exp. Theor. Artif. Intell. 25, 503526, 2013.

[19] Kumar, K.A., Boda, R, “A computer-aided brain tumor diagnosis by adaptive fuzzy active contour fusion model and deep fuzzy classifier”, Multimed Tools Appl 81, 25405–25441 (2022). https://doi.org/10.1007/s11042-022-12213-7