Journal of Ambient Intelligence and Humanized Computing (2023) 14:13349–13359
https://doi.org/10.1007/s12652-022-03788-y

ORIGINAL RESEARCH

RL based hyper-parameters optimization algorithm (ROA) for convolutional neural network

Fatma M. Talaat[1] · **Samah A. Gamel**[2]

Received: 12 November 2021 / Accepted: 25 February 2022 / Published online: 18 March 2022

Abstract

Many real-world applications necessitate optimization in dynamic situations, where the difficulty is to locate and follow the optima of a time-dependent objective function. To solve dynamic optimization problems (DOPs), many evolutionary techniques have been created. However, more efficient solutions are still required. Recently, a new intriguing trend in dealing with optimization in dynamic environments has developed, with new reinforcement learning (RL) algorithms predicted to breathe fresh life into the DOPs community. In this paper, a new Q-learning RL-based optimization algorithm (ROA) for CNN hyperparameter optimization is proposed. Two datasets were used to test the proposed RL model (MNIST dataset, and CIFAR-10 dataset). Due to the use of RL for hyperparameter optimization, very competitive results and good performance were produced. From the experimental results, it is observed that the CNN optimized by ROA has higher accuracy than CNN without optimization. When using the MNIST dataset, it is shown that the accuracy of the CNN optimized by ROA when learning 5 epoch is 98.97%, which is greater than the 97.62% of the CNN without optimization. When using the CIFAR-10 dataset, it is shown that the accuracy of the CNN optimized by ROA when learning 10 epoch is 73.40 percent, which is greater than 71.73% of the CNN without optimization.

Keywords Optimization algorithm · Deep learning · Deep convolutional neural networks · Reinforcement learning

1 Introduction

People can recognize items in their environment and objects in less than a second. Humans have been taught to recognize things since they were children. Similarly, if computers can detect or categorize objects and environments by scanning for low-level characteristics like edges and curves, they may use a sequence of convolutional layers to develop more abstract conceptions of what they see. Convolutional Neural Networks are used in neural networks to recognize and classify images (CNN). Image recognition is used in a variety of applications (Sehgal et al. 2019). Deep Neural Networks are responsible for most of the success in this area. While deep networks have enabled numerous intriguing and valuable applications, there are still several challenges to solve (Tian et al. 2020; Calabrese et al. (2020). One impediment is the lack of an analytical method for determining the appropriate design for a deep network for tackling various issues. For many articles, writers design multiple distinct network topologies before deciding on the optimal one to utilize. This causes a knowledge and effort load in determining the appropriate architecture. Manually selecting and executing these architectures might be time intensive (Duong et al. 2022).

The implementation of CNN necessitates a set of settings that are independent of the data and that the machine learning researcher must manually modify. Hyperparameters are variables that affect the network structure and CNN's trained network (Tian et al. 2020). The hyperparameter optimization challenge also includes finding a collection of hyperparameters that produces an accurate model in an acceptable amount of time. The task of identifying a suitable model of hyperparameter or the problem of optimizing a loss function across a graph-structured configuration space is known as hyperparameter optimization. It can be computationally costly to test every potential set of hyperparameter models.

✉ Fatma M. Talaat
fatma.nada@ai.kfs.edu.eg

✉ Samah A. Gamel
s.adel.gamel@gmail.com

[1] Faculty of Artificial Intelligence, Kafrelsheikh University, Kafrelsheikh, Egypt

[2] Department of Computer Engineering and Systems, Faculty of Engineering, Mansoura University, Mansoura, Egypt

As a result, the demand for an automated and organized search method is growing, and hyperparameter space, in general, is expanding.

Hyperparameter optimization is used to increase the accuracy of neural networks. which has a lot of real-world applications such as signature verification and handwriting analysis, healthcare, traveling salesman problem, image compression, stock exchange prediction, computer vision, speech recognition, and natural language processing(Abiodun et al. 2018, Thanga et al. 2021).

The typical method of accomplishing hyperparameter optimization has been grid search (Chicco et al. 2017) or parameter sweep, which is an exhaustive search of a manually chosen subset of a learning algorithm's hyperparameter space. A grid search algorithm must be directed by a performance metric, which is commonly assessed by cross-validation on the training set or assessment on a held-out validation set. Figure 1 illustrates that Grid search using two hyperparameters with varying values. Each hyperparameter is assessed and compared with ten distinct values, for a total of 100 possible combinations. Blue outlines represent places with strong outcomes, while red contours represent regions with low results.

Random Search (Freitas et al. 2016) is also a hyperparameter optimization approach, it is substituting the exhaustive enumeration of all combinations with a random selection of them. This applies to the discrete environment mentioned above, but it also applies to continuous and mixed areas. It can outperform Grid search, especially when just a limited number of hyperparameters impact the machine learning algorithm's ultimate performance. The optimization problem is said to have a low inherent dimensionality in this situation. Random Search is also embarrassingly parallel, and it allows past information to be included by selecting the distribution from which to sample. Figure 2 illustrates that For two hyperparameters, do a random search across possible combinations of values. In this example, 100 distinct random options are considered. When compared to a grid search, the green bars indicate that more individual values for each hyperparameter are examined.

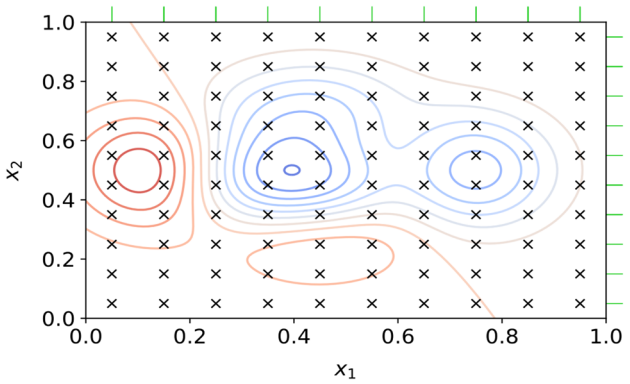

Fig. 1 Grid search optimization

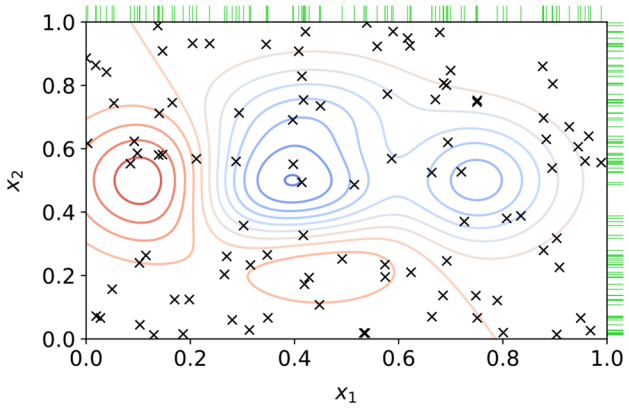

Fig. 2 Random search optimization

On the other hand, Bayesian optimization (Thornton et al. 2013) is a global optimization strategy for noisy black-box functions. Bayesian optimization, when used to hyperparameter optimization, creates a probabilistic model of the function mapping from hyperparameter values to the objective as assessed on a validation set. Bayesian optimization seeks to gather observations exposing as much information about this function and, in particular, the position of the optimum by repeatedly assessing a potential hyperparameter configuration based on the existing model and then updating it. Figure 3 illustrates that methods like Bayesian optimization intelligently examine the universe of alternative hyperparameter options by determining which combination to investigate next based on past discoveries.

Rather than manually tuning these hyperparameters, a novel technique based on reinforcement learning (Minaee et al. 2021) is presented to tune hyperparameters for a convolutional neural network. Instead of requiring a researcher to manually modify hyperparameter knobs in order to gradually

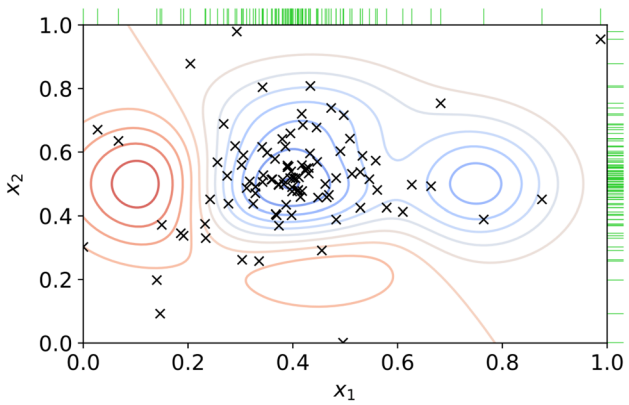

Fig. 3 Bayesian optimization

converge to an ideal solution, this work automates the process and allows an asynchronous reinforcement learning algorithm to automatically alter hyperparameters and discover an optimal configuration. When the algorithm has finished executing, the architecture is ready for usage. With so many presents and future image recognition applications, being able to quickly find appropriate network designs is crucial. In this paper, we will introduce a new Q-learning RL-based Optimization Algorithm (ROA) for CNN hyperparameters optimization. RL overcomes the limitations of the traditional evolutionary techniques. From the experimental results, it is observed that the CNN optimized by ROA has higher accuracy than CNN without optimization.

The following is how the rest of the paper is structured: the recent related works in CNN hyperparameter optimization techniques are detailed in Sect. 2 in Sect. 2. The problem definition is described in Sect. 3. The findings and discussion are presented in Sect. 4 of the report. In Sect. 5, we talk about our conclusion.

2 Background

This section discusses the background knowledge of deep learning, advantages of deep learning, Convolutional Neural Networks, and Reinforcement Learning.

2.1 Deep learning

Deep learning is a type of machine learning technique that employs numerous layers to extract higher-level characteristics from raw input In image processing, for example, lower layers may recognize boundaries, while higher layers may identify concepts meaningful to humans, such as digits, characters, or faces (Tahir et al. 2021, Yuan et al. 2020). The majority of modern deep learning models are based on artificial neural networks, especially convolutional neural networks (CNNs), though they can contain probabilistic formulas or latent variables structured layer-wise in deep generative models like Bayesian networks and deep Boltzmann devices. Each level of deep learning learns to turn the data it receives into a little more complex and composite representation; The second layer may create and encode edge configurations; the third layer may encode a nose and eyes, and the fourth layer may identify the presence of a face in the image. Furthermore, a deep learning process can figure out which traits belong at which level by itself. This does not eliminate the necessity for hand-tuning; for example, adjusting the number of levels and the size of the layers can yield variable degrees of abstraction (Minaee et al. 2021; Luo et al. 2017a, b).

2.2 Advantages of deep learning (DL)

Because of its powerful automatic representation capabilities, deep learning has achieved significant discoveries in a variety of domains (Ren et al. 2021). The importance of neural architecture design in data feature representation and final performance has been demonstrated. The neuronal architecture, on the other hand, is strongly reliant on the researchers' existing knowledge and experience. People find it challenging to break out from their original thinking paradigm and develop an optimal model due to the constraints of human intrinsic knowledge. As a result, it seems logical to limit human interaction as much as possible and let the algorithm develop the neural architecture on its own. The Neural Architecture Search (NAS) algorithm is a game-changing technique, and the research surrounding it is complex and extensive. As a result, a thorough and systematic survey of the NAS is required. Like in (Yan et al. 2021), authors turn to neural architecture search (NAS) and aim to integrate NAS approaches into the ZSL world for the first time.

2.3 Convolutional neural networks (CNN)

The term "convolutional neural network" refers to the network's use of a mathematical procedure known as convolution. Convolutional networks are a subset of neural networks that employ convolution instead of generic matrix multiplication in at least one layer (Goodfellow et al. 2016; Luo et al. 2017a, b). As illustrated in Fig. 4, the convolutional neural networks consist of three layers. An input layer, hidden layers, and an output layer make up a convolutional neural network. Any intermediary layers in a feed-forward neural network are referred to be hidden since the activation function and final convolution hide their inputs and outputs. The convolution layer computes neurons' output related to particular regions in the input volume, with each computing a dot product between their weights and a tiny region in the input volume to which they are connected (Chang et al. 2015). By executing a down-sampling process along the spatial dimensions, pooling reduces the number of retrieved features. The dense layers are responsible for calculating either the hidden convolutions or the class scores.

2.4 Reinforcement learning (RL)

Reinforcement learning tries to educate an agent on how to complete a task by allowing the agent to explore and experience the environment while maximizing a reward signal. It differs from supervised machine learning in that the algorithm learns from a set of samples labeled with the right responses. One advantage of reinforcement learning over

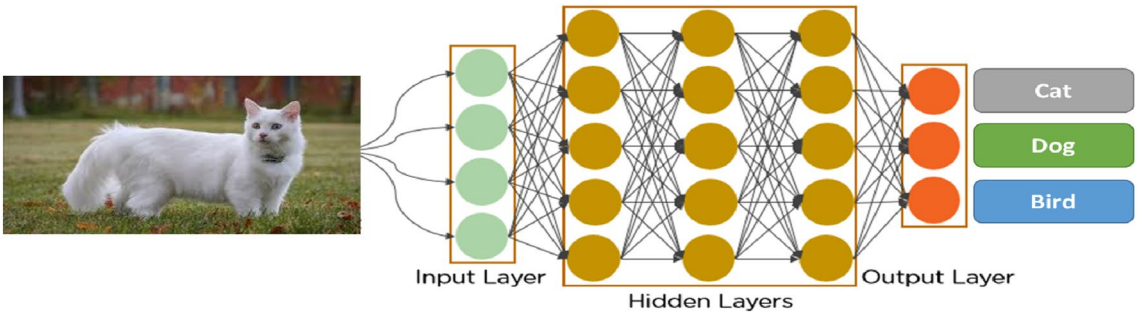

Fig. 4 Conventional neural network

supervised machine learning is that the reward signal may be generated without prior knowledge of the proper path of action, which is especially beneficial if such a dataset does not exist or is impractical to gather. While reinforcement learning may appear to be comparable to unsupervised machine learning at first look, they are not. Unsupervised machine learning seeks to discover some (hidden) organization inside a dataset, whereas reinforcement learning does not seek to discover structure in data (Yamauchi et al. 2020). Reinforcement learning, on the other hand, seeks to educate an agent on how to accomplish a task through incentives and experiments. There are two types of algorithms in reinforcement learning: value-based algorithms and policy-based algorithms. Value-based algorithms attempt to approximate or uncover the value function that provides a reward value to state-action pairings. These reward values can then be included in a policy. The raw input in an image recognition application could be a matrix of pixels; the first layer may extract the pixels and detect edges (Zhu et al. 2021).

3 Related work

The most current works for CNN hyperparameter optimization, such as grid search, Bayesian optimization, Genetic Algorithm (GA), and random search, are discussed in this section.

For each hyperparameter setting on a defined range of values, the grid search method is a trial-and-error method. The use of a grid search has the advantage of being easily parallelized (Bergstra et al. 2012). The boundaries and steps between values of hyperparameters will be specified by researchers and practitioners, resulting in a grid of configurations (Li et al. 2020). However, if one task fails, the others will follow suit. In most circumstances, a machine learner will start with a small grid and subsequently expand it, making it more efficient to set the best grid while searching for a new one (Li et al. 2020). Due to dimensionality constraints, four hyperparameters will become unworkable as the number of functions to assess grows with each additional parameter.

The random search method "randomly" samples the hyperparameter space. According to Bergstra et al. (2012) random search provides greater advantages than grid search in terms of applications that can continue to run even if the computer cluster fails. It allows practitioners to adjust the "resolution" on the fly, as well as add additional trials to the set or even skip the fail test entirely. Simultaneously, the random search process can be stopped at any time, forming a complete experiment that can be run in parallel (Bergstra et al. 2011). Furthermore, if more computers become available, a new trial can be added to the experiment without endangering the results (Bergstra et al. 2013).

Bayesian optimization is another recent advancement in hyperparameter tuning. It employs the Gaussian Process, which is a distribution over functions. To train with the Gaussian Process, it is necessary to fit it to the given data, as it will generate a function that closely resembles the data. The Gaussian process will optimize the predicted improvement and surrogate the model which is the likelihood of the new trial and will enhance the current best observation in Bayesian optimization. The next step will be to determine the largest expected improvement, which can be done at any point in the search space. Spearmint, which uses the Gaussian process, is a widely used example of Bayesian optimization (Snoek et al. 2015). Bergstra et al. (2012) contend that the Bayesian optimization method is constrained because it works with high-dimensional hyperparameters and is computationally expensive. As a result, it performs poorly. Bayesian optimization (BO) works by fitting a probabilistic model to the data and then utilizing that model as a cheap proxy to select the next most promising place to assess. The Gaussian Process Regressor, Bayesian Neural Network (Springenberg et al. 2016), and Random Forest Regressor are some of the proxy models that have been proposed.

Genetic algorithms use a binary representation of individuals (each individual is a string of bits), making mutation and crossover easier to implement. Such processes generate

candidate values that fall outside of the searchable area. On the other hand, evolutionary algorithms rely on specific data structures and require carefully crafted mutation and crossover, which is strongly reliant on the situation at hand (Chiong et al. 2007). When there is no knowledge of the gradient function at assessed sites, genetic algorithms can be applied according to the author (Rojas et al. 1996). When there are multiple local minima or maxima, it can produce good results. Unlike any other search method, the function is determined in multiple places concurrently rather than in a single location. They can be done on many processors because the function calculations on all points of a population are independent of one another (Muhlenbein et al. 1991). They can also be easily parallelized, allowing numerous approaches to the optimal to be processed in parallel.

(Xiao et al. 2020) employed a variable-length genetic algorithm in 2020 to systematically tweak the hyperparameters of CNN to increase performance. This work includes a detailed comparison of several optimization methods such as random search, large-scale evolution, and traditional genetic algorithms. (Liashchynskyi et al. 2019) conducted a rigorous evaluation of optimization strategies such as random search, grid search, and genetic algorithm. These algorithms were utilized to create the Conventional Neural Network by the authors. The dataset for their research is the CIFAR-10 dataset with augmentation and pre-processing procedures. Grid search, according to the authors' experience, is not ideal for huge search spaces. When there is a huge search space and too many parameters to optimize, the authors recommend using the genetic algorithm. Adrian Catalin et al. suggested a new form of a random search for hyperparameter optimization for machine learning algorithms in 2020 (Andonie et al. 2020). This improved random search version creates new values with a likelihood of change for each hyperparameter. The proposed variant of random search outperforms the traditional random search approach. The authors used this for optimizing the CNN hyperparameters. This can be used for any optimization problem in the discrete domain.

Authors in (Li et al. 2018a, b) present a flexible embedding approach for a rank-constrained SC. To recover the block-diagonal affinity matrix of an ideal graph, an adaptive probabilistic neighborhood learning approach is used. The suggested algorithm's performance is demonstrated by experimental findings on both synthetic and real-world data sets. In Sebe et al. (2018), the authors suggested a new method for assigning affinity weights to data points on a per-data-pair basis. The proposed method is effective in learning the affinity network while also fusing characteristics, resulting in better clustering results. A unique Event-Adaptive Concept Integration algorithm is developed, which uses different weights to measure the efficiency of semantically related concepts (Xu et al. 2019). Authors in Yu et al. (2018) employ semantic regression to increase the nearby link between data with similar semantics. To address the data ambiguity problem, the authors of Zhang et al. (2019) loosen the usual ranking loss and propose a unique deep multi-modal network with a top-k ranking loss.

3.1 Problem definition

This section discusses the selected hyperparameters and the used methodology. The proposed algorithm relies on using the Reinforcement Learning (RL) algorithm to optimize the CNN hyperparameters. Firstly, we explain the used CNN hyperparameters in Sect. 4.1. Secondly, we explain the proposed Q-learning RL-based Optimization Algorithm (ROA) in Sect. 4.2 in detail.

3.2 Selected CNN hyperparameters

It can be difficult to define model architectures because there are so many design options. The author does not know what the best model architecture for a given model is right now. As a result, the purpose of this article is to investigate a variety of options. An actual machine learner will instruct the machine to conduct this investigation and automatically create the best model architecture. The hyperparameters are variables in the configuration that are external to the model and whose value cannot be predicted from the data. There are two different types of hyperparameters: (i) Hyperparameter that determines the network structure such as (a) Kernel Size—the size of the filter. (b) Kernel Type-values of the actual filter (e.g., edge detection, sharpen). (c) Stride-the rate at which the kernel passes over the input image. (d) Padding-add layers of 0 s to make sure the kernel passes over the edge of the image. (e) Hidden layer layers between input and output layers. (f) Activation functions-allow the model to learn nonlinear prediction boundaries. (ii) Hyperparameter that determines the network trained such as (a) Learning rate-regulates on the update of the weight at the end of each batch. (b) Momentum–regulates the value to let the previous update influence the current weight update. (c) Some epochs–the iterations of the entire training dataset to the network during training. (d) Batch size–the number of patterns shown to the network before the weights are updated. The hyperparameters investigated in this study are listed in Table 1. Each hyperparameter is given a more concise and understandable name (abbreviation). In addition, ranges are denoted using square brackets. The network-trained hyperparameters are listed in Table 2.

Many real-world applications necessitate optimization in dynamic situations, where the difficulty is to locate and follow the optima of a time-dependent objective function. To solve Dynamic Optimization Problems (DOPs), many evolutionary techniques have been created. However, more efficient solutions are still required. Recently, a new intriguing

Table1 Network structure hyperparameters (NSH)

Hyperparameter	Abbreviation	Range
Number of Filters	Filters_1	[16, 32, 64, 96]
Kernel Size	Ksize_1	[3, 4, 5]
Number of Filters	Filters_2	[48, 64, 96, 128]
Kernel Size	Ksize_2	[3, 4, 5]
Number of Filters	Filter_3	[64, 96, 128]
Kernel Size	Ksize_3	[3, 4, 5]
Hidden Layer	full_hidden1	[60, 100, 125]
Hidden Layer	full_hidden2	[60, 100, 125]
Activation	activation	[„relu", „lrelu", „elu"]

trend in dealing with optimization in dynamic environments has developed, with new Reinforcement Learning (RL) algorithms predicted to breathe fresh life into the DOPs community. In this paper, a new Q-learning RL-based Optimization Algorithm (ROA) for CNN hyperparameter optimization is proposed.

3.3 RL based optimization algorithm (ROA)

RL is an AI technique in which an agent performs a task in order to be rewarded. The agent receives the current status of the environment and takes an appropriate response. The action made causes a change in the environment, and the agent is notified of the change through a reward. The ROA learns to select the optimal values for the selected parameters. The ROA learns to select the optimal values for the selected parameters. ROA is used to optimize the hyperparameters for CNN. The overall steps of ROA are shown in Algorithm 1. The ROA contains an agent called Optimization agent (OA) as shown in Fig. 5.

Table 2 Network trained hyperparameters (NTH)

Hyperparameter	Abbreviation	Range
Learning rate	Learning rate	Learning rate
Learning_rates	Learning_rates	Learning_rates
[0.001, 0.003, 0.01,0.03]	[0.001, 0.003, 0.01,0.03]	[0.001, 0.003, 0.01,0.03]
Batch Size	Batch Size	Batch Size

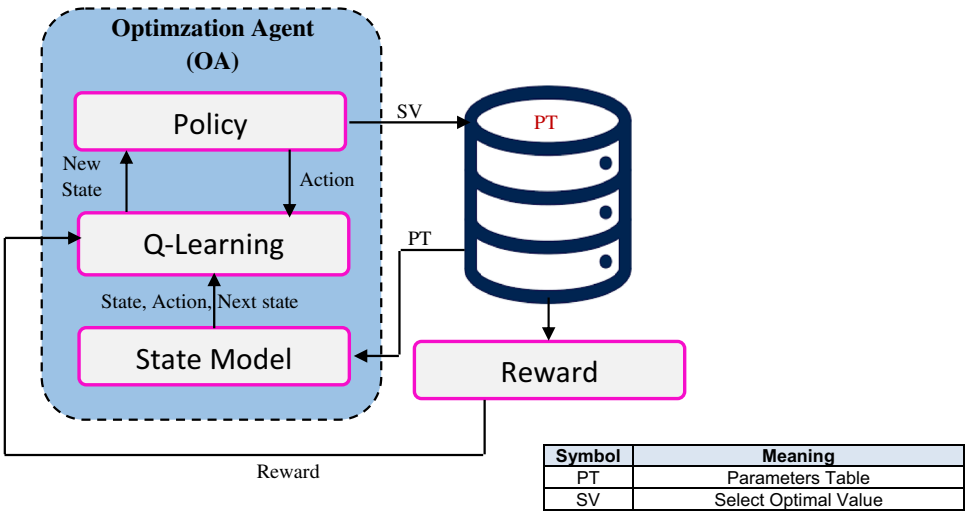

Fig. 5 RL based Optimization framework

RL based Optimization Algorithm (ROA)

- ***Input:***
  - The data in the PT
- ***Output:***
  - The optimized values for each hyperparamert in PT.

Symbol	Meaning
PT	Parameters Table
CNN	Convolutional Neural Network

- ***Steps:***

1: **For each hyperparameter do**
2: Create a Q-table containing two columns [State: data in PT, Action: Selecting the best value for each hyperparameter] initialized to zero.
3: **//Taking Action**
4: The agent interacts with the environment (training and testing the CNN) and makes updates to the state-action pairs in the Q-table Q[state, action].
5: The agent uses the Q-table as a reference and views all possible actions (all possible available values for each hyperparameter) for a given state. Then it selects the best value for each hyperparameter with the best accuracy.
8: **// Updating the Q-table**
9: Update the values in the Q-table
10: **//Update the data in PT**
11: The OA updates the data in the PT (update the values for each hyperparamert in PT)
12: **Next**

As shown from Fig. 5, The ROA is used to update the values of the hyperparameters in the parameter Table (PT) which is a table used to save the values of all hyperparameters (either Network Structure Hyperparameters (NSH) or Network Trained Hyperparameters (NTH)).

The overall steps of the RL-based Optimization Algorithm (ROA) are as follows: (i) For each hyperparameter, create a Q-table containing two columns (state and action). The state is described as the overall data in the parameter table, and the action is described as the selection of the best value for each hyperparameter. (ii) The agent interacts with the environment (train and test the CNN using the parameters in the parameter table). (iii) Makes updates to the state-action pairs in the Q-table Q[state, action]. (iv) The agent uses the Q-table as a reference and views all possible actions (all possible available values for each hyperparameter) for a given state. (v) Then it selects the best value for each hyperparameter with the best accuracy. (vi) Update the values in the Q-table. (vii) The OA updates the data in the PT (update the values for each hyperparameter in the parameter table).

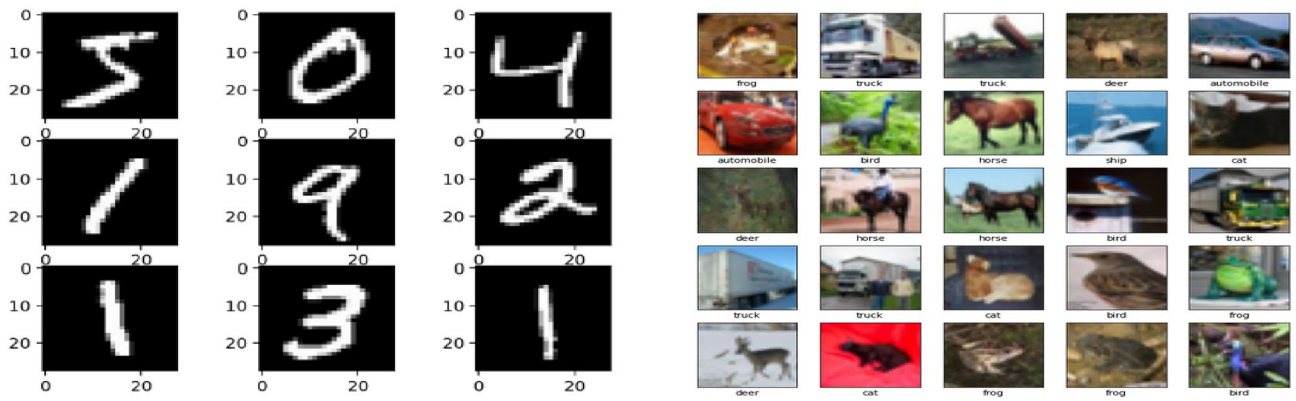

Fig. 6 Example of some images; **A** MNIST dataset, **B** CIFAR-10 dataset Figs. 1, 5

Table3 Accuracy and Loss of CNN without optimization and CNN optimized by ROA using MNIST dataset

Epoch	CNN without optimization		CNN optimized by ROA	
	Accuracy	Loss	Accuracy	Loss
1	91.36	30.22	92.96	29.70
2	95.74	14.79	97.48	14.43
3	96.72	10.8	98.07	10.79
4	97.23	9.01	98.52	9.00
5	97.62	7.65	98.97	7.05

4 Experiment and results

4.1 Used datasets

We experiment with two datasets in this study. The MNIST dataset is used in the first experiment, and the CIFAR-10 dataset is used in the second. We also calculate the training time and testing time in case of using each dataset. Firstly, in case of using MINIST dataset, the training time = 21.315921783447266 s and the testing time = 0.5247492790222168 s. Secondly, in case of using CIFAR-10 dataset, the training time = 803.1955409049988 s and the testing time = 4.01332426071167 s. A sample of each dataset is shown in Fig. 6.

4.2 Experiment using MNIST dataset

MNIST is a handwritten number image dataset that contains 60,000 images for learning and 10,000 for testing. From 0 to 9 proper labels are assigned to each image. In the experiment, optimization was carried out with ROA every 5 epochs, and learning was carried out using the parameters gained. Each experiment is repeated 30 times to obtain an average value (Table 3).

Figure 7 illustrates the accuracy of the CNN without optimization vs. the accuracy of the CNN optimized by ROA using MNIST dataset. Figure 8 illustrates the loss of the CNN without optimization vs. the loss of the CNN optimized by ROA using MNIST dataset.

From Table 3 and Fig. 7, it is shown that the accuracy of the CNN optimized by ROA when learning 5 epoch is 98.97 percent, which is greater than the 97.62 percent of the CNN without optimization. When examining any epoch, it is discovered that the CNN optimized by ROA has higher accuracy than the CNN without optimization.

4.3 Experiment using CIFAR-10 dataset

The CIFAR-10 dataset consists of 50,000 learning images and 10,000 testing images. It has a 10-class label and is frequently used as a standard for object recognition. In the experiment, optimization is done with ROA every 10 epochs, and learning is done with the parameters that are enhanced. Each experiment is repeated 30 times to obtain an average value. Table 4 illustrate the accuracy and Loss.

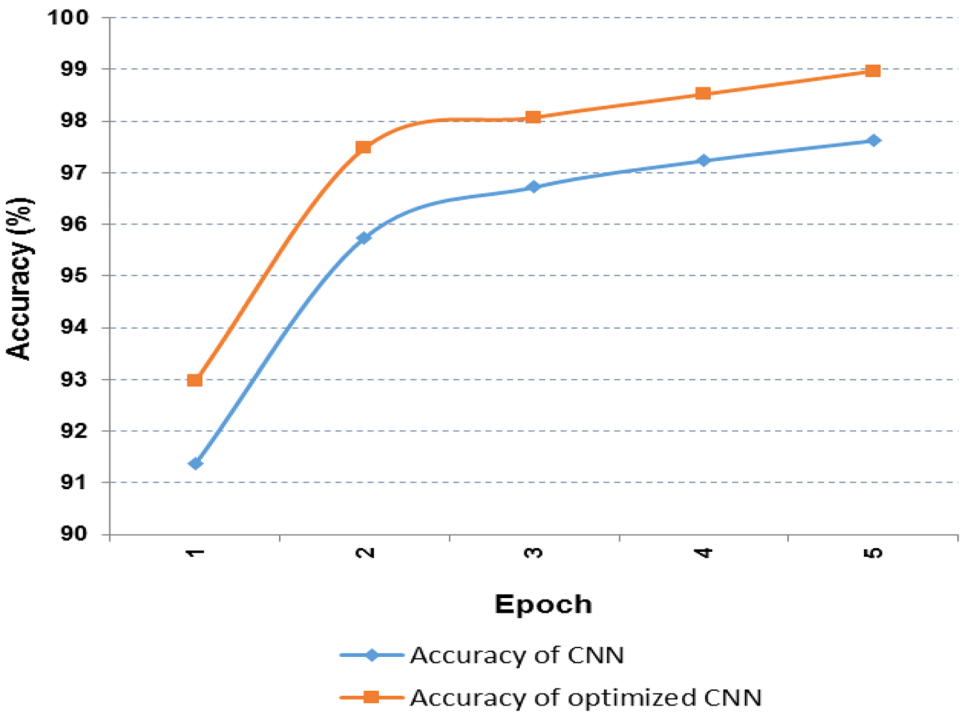

Fig. 7 Accuracy without optimization vs. Accurracy with ROA using MNIST dataset

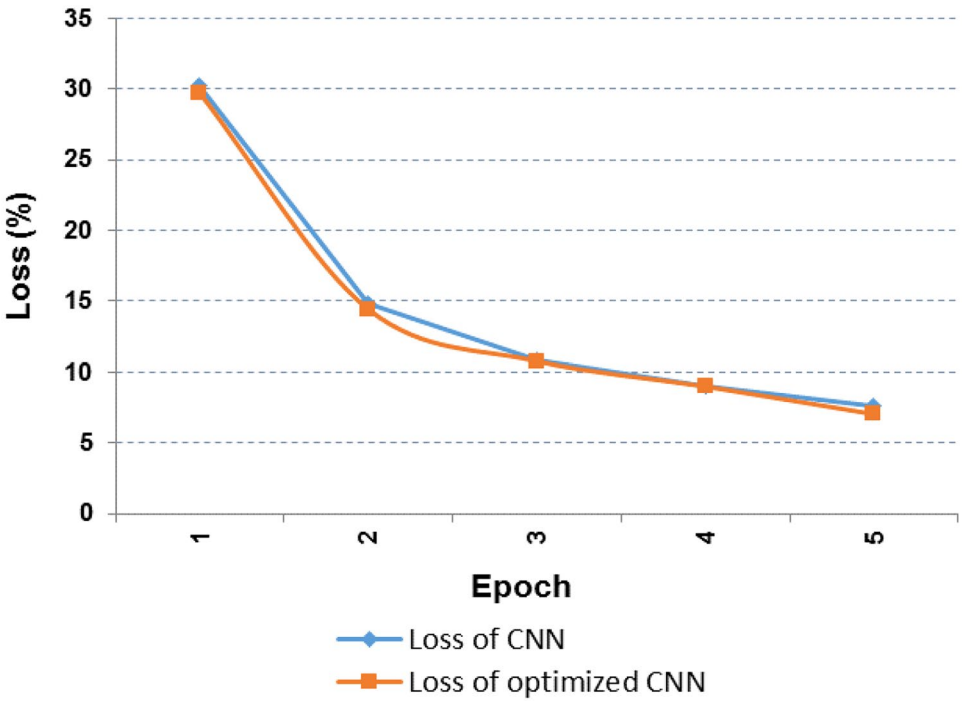

Fig. 8 Loss without optimization vs. Loss with ROA using MNIST dataset

Table 4 Accuracy and Loss of CNN without optimization and CNN optimized by ROA using CIFAR-10 dataset

Epoch	CNN without optimization		CNN optimized by ROA	
	Accuracy	Loss	Accuracy	Loss
1	52.62	1.594	63.96	1.3115
2	63.19	1.074	64.54	1.0522
3	65.14	1.009	66.50	0.9927
4	68.39	0.9319	69.76	0.9136
5	68.75	0.9154	70.13	0.8974
6	70.59	0.8682	71.98	0.8515
7	70.59	0.8839	72.02	0.8664
8	70.89	0.8798	72.33	0.8623
9	71.05	0.8729	72.51	0.8554
10	71.73	0.8888	73.40	0.8686

Figure 9 illustrates the accuracy of the CNN without optimization vs. the accuracy of the CNN optimized by ROA using CIFAR-10 dataset. Figure 10 illustrates the loss of the CNN without optimization vs. the loss of the CNN optimized by ROA using CIFAR-10 dataset.

From Table 4 and Fig. 9, it is shown that the accuracy of the CNN optimized by ROA when learning 10 epoch is 73.40 percent, which is greater than the 71.73 percent of the CNN without optimization. When examining any epoch, it is discovered that the CNN optimized by ROA has higher accuracy than the CNN without optimization.

5 Conclusions

In this paper, a new Q-learning RL-based Optimization Algorithm (ROA) for CNN hyperparameter optimization was proposed. Two datasets were used to test the proposed RL model (MNIST dataset, and CIFAR-10 dataset). Due to the use of RL for hyperparameter optimization, very competitive results and good performance were produced. RL overcomes the limitations of the traditional evolutionary techniques. RL algorithms are predicted to breathe fresh life into the DOPs community. From the experimental results, it is observed that the CNN optimized by ROA has higher accuracy than CNN without optimization. When using MNIST dataset, it is shown that the accuracy of the CNN optimized by ROA when learning 5 epoch is 98.97 percent, which is greater than the 97.62 percent of the CNN without optimization. When using CIFAR-10 dataset, it is shown that the accuracy of the CNN optimized by ROA when learning 10 epoch is 73.40 percent, which is greater than the 71.73 percent of the CNN without optimization.

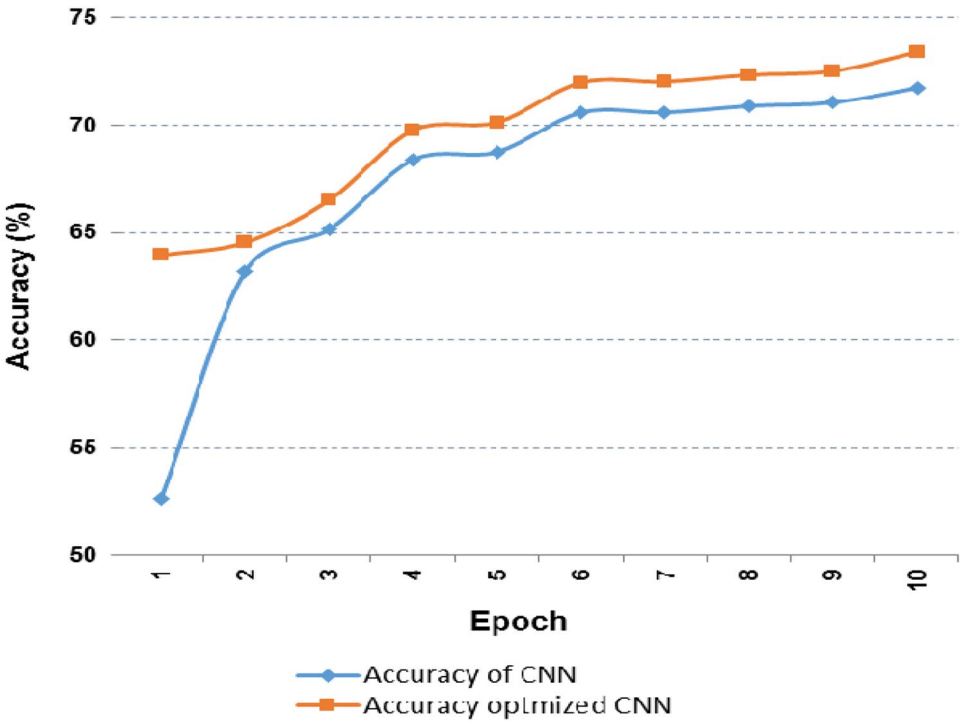

Fig. 9 Accuracy without optimization vs. accurracy with ROA using CIFAR-10 dataset

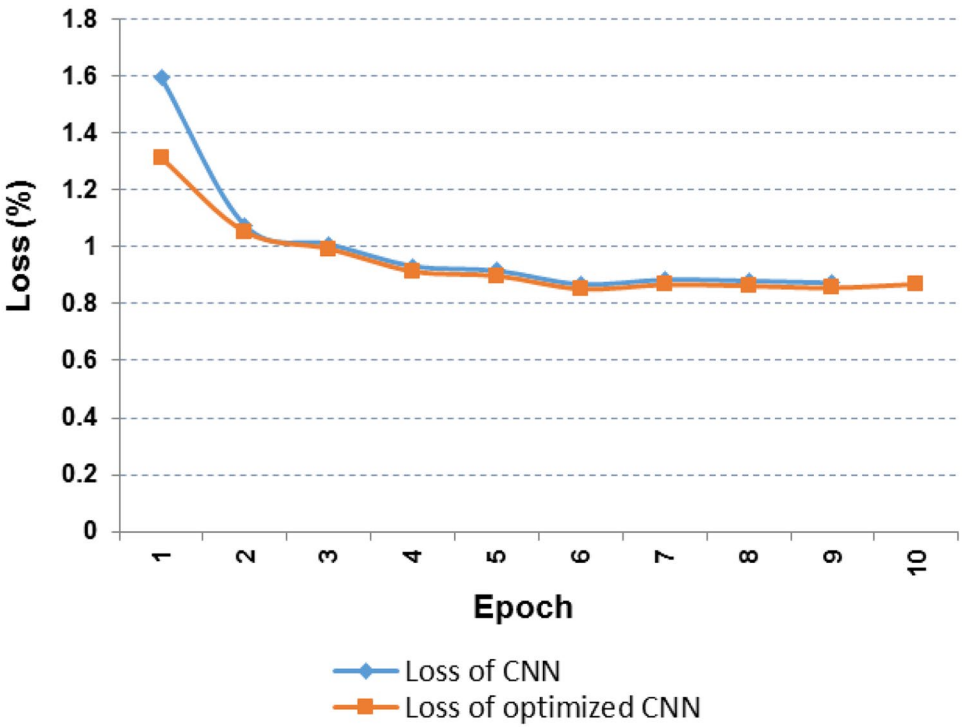

Fig. 10 Loss without optimization vs. loss with ROA using CIFAR-10 datasett

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

References

Abiodun OI, Jantan A, Omolara AE, Dada KV, Mohamed NA, Arshad H (2018) State-of-the-art in artificial neural network applications: a survey. Heliyon 4(11):e00938

Andonie R, Florea AC (2020) Weighted random search for CNN hyperparameter optimization. arXiv preprint arXiv:2003.13300.

Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Research 13(2).

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. Adv Neural Inform Process Syst 24.

Bergstra J, Yamins D, Cox D (2013) Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In: International conference on machine learning (pp. 115–123). PMLR.

Calabrese S, Donati E, Chousidis C (2020) Prediction of hearing loss through application of Deep Neural Network. In: Audio Engineering Society Convention 148. Audio Engineering Society.

Chang X, Nie F, Wang S, Yang Y, Zhou X, Zhang C (2015) Compound rank-$ k $ projections for bilinear analysis. IEEE Trans Neural Netw Learn Syst 27(7):1502–1513

Chicco D (2017) Ten quick tips for machine learning in computational biology. BioData Mining 10(1):1–17

Chiong R, Beng OK (2007) A comparison between genetic algorithms and evolutionary programming based on cutting stock problem. Eng Lett 14(1):72–77

de Freitas N (2016) Bayesian optimization in a billion dimensions via random embeddings.

Duong D, Waikel RL, Hu P, Tekendo-Ngongang C, Solomon BD (2022) Neural network classifiers for images of genetic conditions with cutaneous manifestations. Hum Gen Genom Adv 3(1):100053

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT press.

Li L, Talwalkar A (2020) Random search and reproducibility for neural architecture search. In: Uncertainty in artificial intelligence (pp. 367–377). PMLR.

Li Z, Nie F, Chang X, Nie L, Zhang H, Yang Y (2018a) Rank-constrained spectral clustering with flexible embedding. IEEE Trans Neural Netw Learn Syst 29(12):6073–6082

Li Z, Nie F, Chang X, Yang Y, Zhang C, Sebe N (2018b) Dynamic affinity graph construction for spectral clustering using multiple features. IEEE Trans Neural Netw Learn Syst 29(12):6323–6332

Liashchynskyi P, Liashchynskyi P (2019) Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv:1912.06059.

Luo M, Nie F, Chang X, Yang Y, Hauptmann AG, Zheng Q (2017a) Adaptive unsupervised feature selection with structure regularization. IEEE Trans Neural Netw Learn Syst 29(4):944–956

Luo M, Chang X, Nie L, Yang Y, Hauptmann AG, Zheng Q (2017b) An adaptive semisupervised feature analysis for video semantic recognition. IEEE Trans Cybern 48(2):648–660

Minaee S, Boykov YY, Porikli F, Plaza AJ, Kehtarnavaz N, Terzopoulos D (2021) Image segmentation using deep learning: a survey. IEEE Trans Pattern Anal Mach Intell.

Muhlenbein H (1991) Asynchronous parallel search by the parallel genetic algorithm. In: Proceedings of the third IEEE Symposium on Parallel and Distributed Processing (pp. 526–533). IEEE.

Ren P, Xiao Y, Chang X, Huang PY, Li Z, Chen X, Wang X (2021) A comprehensive survey of neural architecture search: challenges and solutions. ACM Comput Surv (CSUR) 54(4):1–34

Rojas R (1996) Neural networks-a systematic introduction. Springer, Berlin, New-York

Sehgal A, La H, Louis S, Nguyen H (2019) Deep reinforcement learning using genetic algorithm for parameter optimization. In: 2019 Third IEEE International Conference on Robotic Computing (IRC) (pp. 596–601). IEEE.

Snoek J, Rippel O, Swersky K, Kiros R, Satish N, Sundaram N, Adams R (2015) Scalable bayesian optimization using deep neural networks. In: International conference on machine learning (pp. 2171–2180). PMLR.

Springenberg JT, Klein A, Falkner S, Hutter F (2016) Bayesian optimization with robust Bayesian neural networks. Adv Neural Inform Process Syst 29.

Tahir W (2021) Deep learning for large-scale holographic 3d particle localization and two-photon angiography segmentation (Doctoral dissertation, Boston University).

Thanga Selvi R, Muthulakshmi I (2021) An optimal artificial neural network based big data application for heart disease diagnosis and classification model. J Ambient Intell Humaniz Comput 12(6):6129–6139

Thornton C, Hutter F, Hoos HH, Leyton-Brown K (2013) Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 847–855).

Tian Y (2020) Artificial intelligence image recognition method based on convolutional neural network algorithm. IEEE Access 8:125731–125744

Tian C, Fei L, Zheng W, Xu Y, Zuo W, Lin CW (2020) Deep learning on image denoising: an overview. Neural Netw 131:251–275

Xiao X, Yan M, Basodi S, Ji C, Pan Y (2020) Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm. arXiv preprint arXiv:2006.12703.

Xu J, An W, Zhang L, Zhang D (2019) Sparse, collaborative, or nonnegative representation: which helps pattern classification? Pattern Recogn 88:679–688

Yamauchi Y (2020) Effective data augmentation method with sequence numbers from chaos phenomenon for convolution neural network learning Yuji Yamauchi, Yuichi Miyata, Yoko Uwate and Yoshifumi Nishio.

Yan C, Chang X, Li Z, Guan W, Ge Z, Zhu L, Zheng Q (2021) Zeronas: differentiable generative adversarial networks search for zero-shot learning. IEEE Trans Pattern Anal Mach Intell.

Yu E, Sun J, Li J, Chang X, Han XH, Hauptmann AG (2018) Adaptive semi-supervised feature selection for cross-modal retrieval. IEEE Trans Multimedia 21(5):1276–1288

Yuan Q, Shen H, Li T, Li Z, Li S, Jiang Y, Zhang L (2020) Deep learning in environmental remote sensing: achievements and challenges. Remote Sens Environ 241:111716

Zhang L, Luo M, Liu J, Chang X, Yang Y, Hauptmann AG (2019) Deep top-k ranking for image-sentence matching. IEEE Trans Multimed 22(3):775–785

Zhu Z, Zhao H (2021) A survey of deep rl and il for autonomous driving policy learning. IEEE Trans Intell Transp Syst.