Automatically Avoiding Overfitting in Deep Neural Networks by Using Hyper-Parameters Optimization Methods

https://doi.org/10.3991/ijoe.v19i05.38153

Zahraa Saddi Kadhim(✉), Hasanen S. Abdullah, Khalil I. Ghathwan
Department of Computer Science, University of Technology, Baghdad, Iraq
cs.20.44@grad.uotechnology.edu.iq

Abstract—Overfitting is one issue that deep learning faces in particular. It leads to highly accurate classification results, but they are fraudulent. As a result, if the overfitting problem is not fully resolved, systems that rely on prediction or recognition and are sensitive to accuracy will produce untrustworthy results. All prior suggestions helped to lessen this issue but fell short of eliminating it entirely while maintaining crucial data. This paper proposes a novel approach to guarantee the preservation of critical data while eliminating overfitting completely. Numeric and image datasets are employed in two types of networks: convolutional and deep neural networks. Following the usage of three regularization techniques (L1, L2, and dropout), apply two optimization algorithms (Bayesian and random search), allowing them to select the hyperparameters automatically, with regularization techniques being one of the hyperparameters that are automatically selected. The obtained results, in addition to completely eliminating the overfitting issue, showed that the accuracy of the image data was 97.82% and 90.72% when using Bayesian and random search techniques, respectively, and was 95.3% and 96.5% when using the same algorithms with a numeric dataset.

Keywords—deep learning, hyper-parameters optimization, regularization, overfitting

1 Introduction

The outstanding performance of deep neural network (DNN) models in a variety of tasks has generated a lot of interest. However, Overfitting is a universal issue that DNN is dealing with. This effect is noticed when a learning algorithm memorizes the noise and characteristics of the training data set because it matches the training dataset so well. When evaluated with an unknown data set, learning algorithms assessment of the performance according to their findings [1]. There are many well-known ways to get rid of overfitting. Regularization is one of these methods, which makes parameters heavier by adding penalties and lessens the impact of overfitting, such as L1 and L2 overfitting. However, it did not entirely stop it [2]. Another method of preventing overfitting is a dropout, which involves randomly removing specific neurons from a DNN during model training [3], [4]. On the other hand, some issues could be solved differently depending on

the ML hyperparameters. Random Search (RS) is the hyperparameter selection method for machine learning algorithms that is most frequently used [5], [6]; additionally, For minimizing or maximizing objective functions in optimization scenarios, Bayesian optimization algorithms are excellent optimization strategies [7]. In this work, in the first step, three regularization techniques performed to the model with the overfitting issue in the second step, using two optimization algorithms Bayesian and random search algorithms are propose, and having them treat regularization methods as a type of hyperparameter. Then propose choosing the best types of hyperparameter that guarantee prevent overfitting completely while giving high and real accuracy at the same time.

The remaining sections of this project are structured as follows: The second section is literature on earlier approaches that have an overfitting problem. The method and tools used in the proposed structure are described in the third section. The measurements used to evaluate the suggested system are explained in the fourth section. The proposed system structure is presented in section five—the discussion and results of the effort are presented in section six. The conclusion is provided in section seven.

2 Literature review

Overfitting can be prevented by data augmentation, which enlarges the data to include more photos in the dataset. A number of techniques can be used to augment images, such as translating, flipping, scaling, rotating, adding noise, altering brightness, etc. [8]. for instance [9] implemented an augmentation method to the training data, enhancing the suggested architecture's reliability and resistance to data memory. They attained a 97.772% testing accuracy. However, in the case of overfitting, they reduce it but cannot overcome it completely. While [4] L1, L2, and dropout regularization techniques are employed. They provide a thorough rundown of optimization and regularization methods. Their findings demonstrate that the L1 and L2 are not effective dropouts for preventing overfitting, but they did improve model accuracy. The Early stopping approach is one technique utilized in order to avoid over-fitting. It is employed while working with large datasets. However, the model does not use all of the training data that are available, so it will be useless, particularly in cases when there is little training data available [10], [11] used the early stopping as a way to prevent overfitting in their work. Although the best result they obtained but they have difficulty deciding the best time to stop training by only looking at learning curve and possibility to overfitting training data in the case where the training session is not stopped at the correct point. To address the overfitting issue, the genetic algorithm and particle swarm optimization are two evolutionary algorithms that [12] invented in order to accomplish the autonomous selection of optimal hyperparameter values. In the task of function minimization, both the PSO and the GA show their aptitude for determining the ideal parameter value. Nevertheless, they successfully address the overfitting issue. In addition [13] and [14] both addressed overfitting problem by using regularization methods l2, dropout although they performance the accuracy but not grantee prevent overfitting also they fall in the local minima problem. lastly [15], proposed A method based on Bayesian optimization was used to automatically optimize hyperparameters to ensure the optimum DNN design, but it encountered an overfitting issue. They used dropout technique to solve it.

They completed their investigation, obtained good results free of the overfitting problem but they had to randomly eliminating neurons during training.

3 Methods

The methodologies and tools that served as the foundation for the proposed system are examined in this section.

3.1 Dataset description

- MNIST Dataset

It is an image collection of pictures of handwritten numbers from 0 to 9. It has ten classes and a 28 by 28 matrix. Accordingly, there are 784 characteristics, and labels come in a single hot form with the shapes 1 through 10.

- Smart grid stability

It is numeric data classify the electrical grid stability. The dataset consists of 60000 rows × 14 columns, and the labels come in binary form, stable or unstable.

3.2 Neural network

To get the best set of weights and biases in the quickest amount of time, an ANN must map input into output during its training process, which could be specified as a continuous optimization process [16]. The non-linear qualities that their ANN activation functions provide could let the net learn any complex relationship between input and output data, or what is known as a universal approximation [17].

Deep neural network (DNN). The typical DNN structure consists of multiple hidden layers, one input layer, and one output layer. The circles are used to represent neurons, and each link between neurons is a cause-and-effect chain that can be educated and learned. Because the layers are completely coupled, every neuron in one layer is connected to every neuron in the next layer. A linear function in Equation (1) and the following activation function makes up the entire DNN model.

$$a = \sum w_i x_i + b_i \tag{1}$$

Where x_i represents each neuron's input value, w_i represents the linear relationship's coefficient, and b_i represent the bias. Considering that there are L hidden layers in DNN, the calculation regarding the output value could be expressed in the following Equation (2)

$$f(x) = W^L x + b \tag{2}$$

which L represents the number of layers, x represents the input variable matrix for several of the non-linear models, (b) and (W) are high-dimensional matrices, and $f(x)$ is an activation function introduced to increase the NN's non-linearity [18].

Conventional neural network (CNN). One of the most popular types of deep neural networks is convolutional neural networks (CNN) [19]. Convolution describes a mathematically straightforward linear operation between matrices. The various layers of CNN include convolutional, non-linear, pooling, and fully connected layers, to name just a few. Convolutional and fully connected layers have parameters, whereas pooling and non-linearity layers do not. CNN does quite well in machine learning problems. In particular, the results from image-related applications, including the largest image classification data set (Image Net), computer vision, and natural language processing (NLP), were astonishing [20], [21]. Adding additional parameters makes overfitting more likely, particularly on small training datasets. This has made it difficult to apply such complicated models to various practical problems [22].

3.3 Regularization

In general, a model's output can be impacted by a number of characteristics. With more elements, the model becomes more complicated. An overfitting model has the propensity to consider every feature, even while some only have a minor impact on the outcome. Alternatively, what is worse, some of them are just noises that do not affect the output. Limiting the effects of those unneeded features is necessary to prevent these situations. However, since the useless features are unknown, all of them are restricted by minimizing the model's cost function. By incorporating a "penalty term" into the cost function as indicated in Equations (3–4)

$$\tilde{J}(\omega; X, y) = J(\omega; X, y) + \alpha\Omega(\omega) \tag{3}$$

$$\tilde{J}(\omega; X, y) = \frac{1}{2m}\left\|X_w - y\right\|_2^2 + \alpha\Omega(\omega) \tag{4}$$

Where $J(\omega; X, y)$ represents the original cost function, α is the regularization coefficient, m is the training set size, ω means the weight, the training set by y is the actual value, and $\alpha\Omega(\omega)$ is the penalty term. As shown in the formula (4), the bigger m is, the smaller $\lambda\frac{\omega^{(k)}}{m}$ Will be. In other words, the danger of overfitting and the regularization effect decreases as the size of the training set increases [23]. The most significant types of regularization methods are:

- **L1 regularization** penalty term with the coefficients "absolute value of magnitude" applied to the loss function [24]. Since L1 regularization offers sparse solutions, it is the preferred number when there are many features.

$$\Omega(\omega) = \lambda\sum_{i=1}^{m}\left|w_i\right| \tag{5}$$

Where λ is the parameter of regularization. L1 sets some feature weights to zero. In other words, it eliminates some characteristics from the model and retains only the most valuable ones. On this basis, a model that is simpler and easier to understand can be

obtained. However, certain beneficial traits that have less of an impact on the outcome are also lost.

- **L2 regularization** is one of the regularization methods that are most prevalent. Summarizing the squared magnitudes of all the parameters, including the weights and biases, would enable the cost function to be used, as in Equation (6). This total cost function is further decreased using an optimizer [25].

$$\Omega(\omega) = \frac{\lambda}{2} \sum_{i=1}^{m} w_i^2 \tag{6}$$

That is add a term ½ λw^2 for every weight, where λ represents the regularization parameter. A factor of ½ multiplied to simplify the gradient term. The networks prefer to learn features with small weights when L2 regularization is used. Give those fewer valuable traits lower weights as opposed to eliminating them. As a result, the model gathers as much data as it can. Only features that significantly improve the original cost function can be given large weights [26].

- **Dropout** signifies that there is a chance that a neural network neuron will be turn off during training. At each stage of the training phase, the likelihood of each neuron passing is assessed using Bernoulli's distribution, which introduces some unpredictability into the process. Dropped neural networks are more generalizable than regular neural networks, according to the original research [2].

3.4 Hyper-parameters optimization methods

Selecting the ideal set of hyper-parameters for a learning algorithm is a process that is frequently referred to as tuning. A hyper-parameter is a parameter whose value used to regulate the manner in which learning takes place [27]. To generalize various data patterns, the same ML model may need various weights, constraints, or learning rates. Hyper-parameters are those parameters, and they need to be adjusted for the model to work best at solving the ML problem. Through the use of hyper-parameter optimization, a model is produced that reduces a predetermined loss function on a set of independent data [28].

Random search. These methods function by first establishing a hyper-parameter search space, after that finding the hyper-parameters combination within it, and then choosing the hyper-parameters combination that performs the best. Rather than exhaustively listing every possible combination, it selects a few options at random. In the case when just a few hyper-parameters have a number on the ML algorithm's final performance, it can outperform the competition [29]. In this case, the optimization problem is thought to have a low intrinsic dimensionality. Random search, despite being straightforward, continues to be a crucial benchmark for evaluating the effectiveness of new hyper-parameters optimization techniques [30].

Bayesian optimization. Bayesian optimization (BO) is one of the probabilistic optimization techniques that aim to minimize a global objective's black-box function [15]. Accordingly, BO identified the optimal hyper-parameters combination in fewer iterations than RS did. By estimating the next value of the hyper-parameter based on

previous findings regarding the values of the tested hyper-parameters, the models could avoid making many unnecessary assessments [31].

4 Evaluation measurement

In the case when evaluating a model's output, some parameters are utilized to examine the activities of the model. The consistency of the files, the amount of training data, and—most significantly—the kind of supervised ML algorithm employed all have an impact on the results. The next parameters are used to evaluate the models' effectiveness [32]:

- Accuracy: the percentage of instances properly identified among all those provided is calculated in the following way [33]:

$$\text{Accuracy} = \frac{a+b}{a+b+c+d} \tag{7}$$

- Precision (Pre): the percentage of true instances regarding class x for all those who are designated as such; the result is as follows [34]:

$$\text{Precision} = \frac{a}{a+c} \tag{8}$$

- Recall (*Rc*): the following formula is used to calculate the proportion of examples in class x out of all instances:

$$\text{Recall} = \frac{a}{a+d} \tag{9}$$

- F-measure: the harmonic mean of precision and recall is calculated in the following way [35]:

$$\text{F1} = 2 * \frac{\text{Pre} * Rc}{\text{Pre} + Rc} \tag{10}$$

Where (*a*) stands for true positives or the number of samples that tested positive when it was not anticipated that they would. The percentage of pieces that were expected to be negative but ended up being negative is referred to as "true negatives" and is displayed in (*b*). Additionally, (*c*) the number of samples that were predicted to be positive but turned out to be negative. The number of samples that expected to be negative but turned out to be positive shown in (*d*).

5 The proposed system design

Based on what was discussed previously, we advise selecting the optimum hyper-parameters with the use of Bayesian and random search techniques to prevent overfitting from occurring, as shown in (Figure 1).

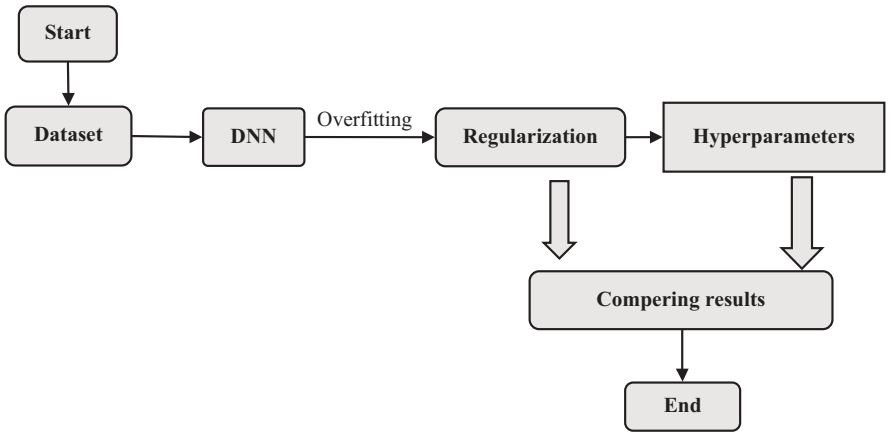
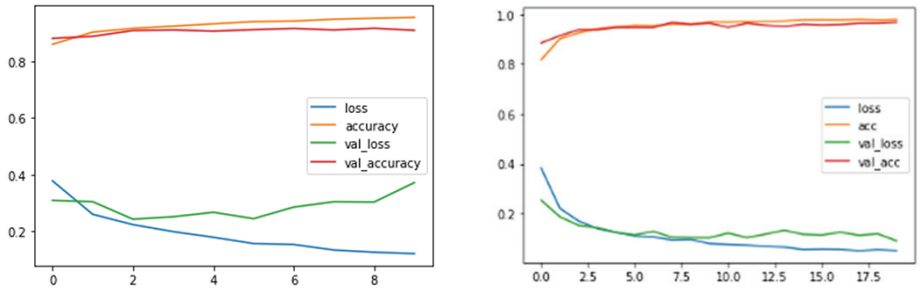


Fig. 1. The general overview of the proposed system's architecture

A proposed system's overall architecture is depicted in Figure 1 in this part. It went through three stages to make sure that work was precise, beginning with selecting a dataset, followed by developing a model that is more complex than the chosen data set to ensure an overfitting appearance. Three methods of regularization (L1, L2, and dropout) were applied in order to discover the strength of the proposed theory and whether these methods are able to completely get rid of excess on their own or not. Then, optimization algorithms were applied and made to control the hyper-parameters automatically, where two algorithms, Random search and Bayesian used, and then the results were compared.

5.1 The model structure

The appearance of the overfitting when applying deep learning to datasets is shown in (Figure 2).



CNN-MINST DNN-Smart grid stability

Fig. 2. Appearance of the overfitting in CNN and DNN with a different types of data

As seen in Figure 2, the overfitting is evident by the rise in the ratio of the Val-loss to the loss after it intersected with it at a point. In addition, as the difference increases, the overfitting increases.

5.2 Regularization

To avoid the over-fitting problem, use regularization approaches with (L2, L1, and dropout) types, as depicted in Figures 3 and 4 for the MINST and Smart grid stability datasets, respectively.

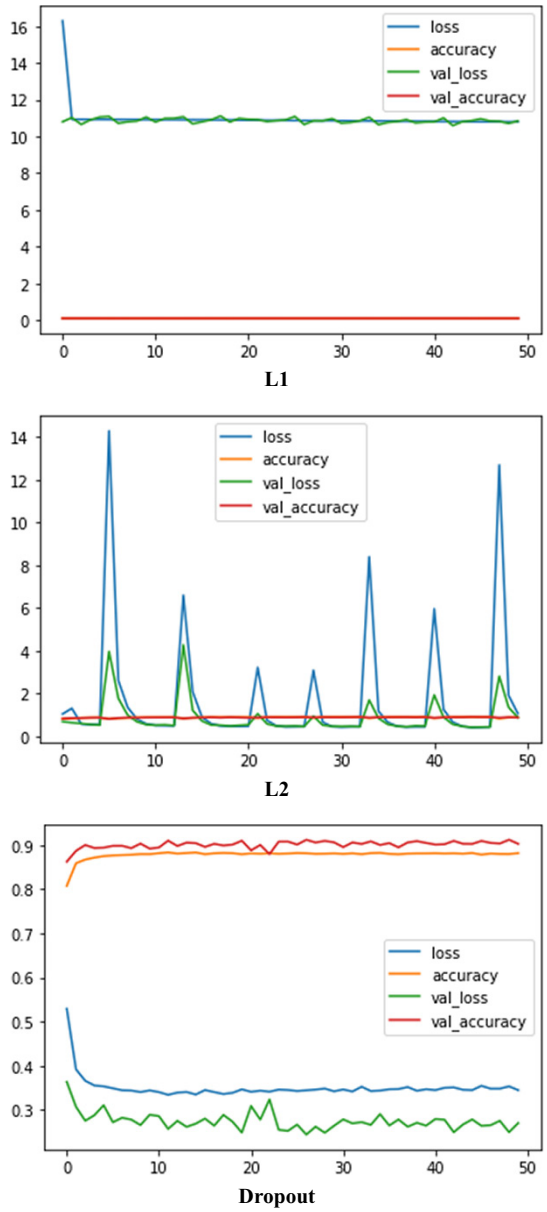


Fig. 3. Regularization techniques' effect on the MINST dataset

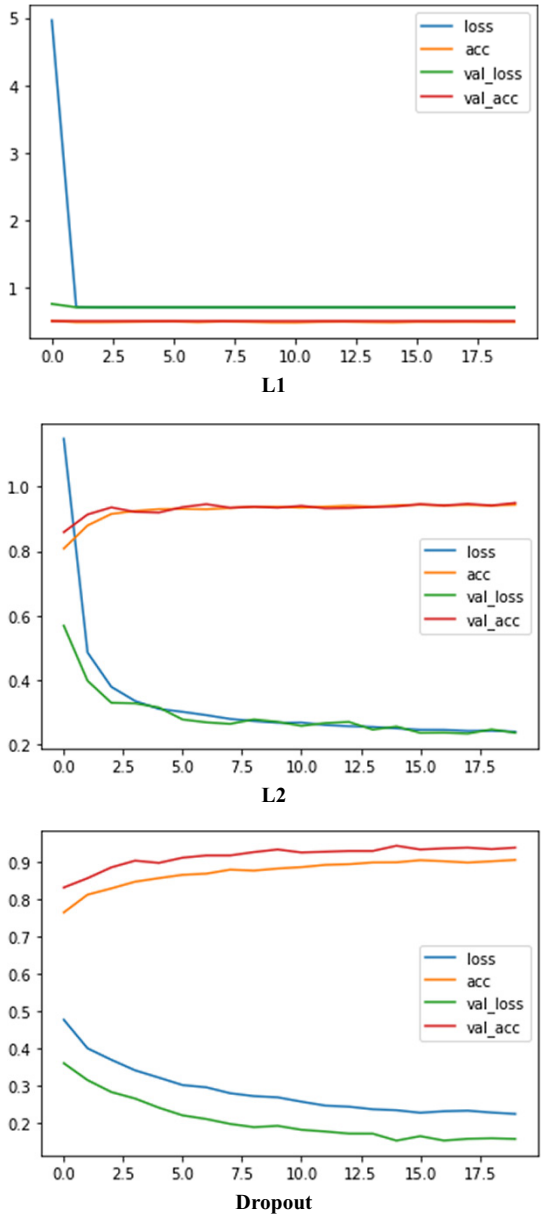


Fig. 4. Regularization techniques' effect on the smart grid stability dataset

5.3 Hyper-parameters optimization

At this point, the overfitting problem was guaranteed to be resolved, and the high accuracy regarding the classification within the model was maintained by combining the two techniques Bayesian or Random search with regularization methods. The regularization

methods are handled as hyperparameters that are chosen automatically, without human intervention, in each layer, in addition to the other hyperparameters.

6 Results and discussion

The result and the discussion shown in Table 1 on the presence of the overfitting and the range of regularization approaches are taken into consideration for treating it using what was previously explained in Figures 3 and 4.

Table 1. Statement of the position of the existence of overfitting and discussion

Dataset	Regularization Methods	Overfitting Appearance	Discuss the Result
MINST	L1	No	The model had a high (loss) value, which caused underfitting and signaled that it had not been effectively taught, even though overfitting did not occur.
	L2	No	Although there is no overfitting, the spikes lead to not good results.
	dropout	No	There are good results, no problems, and no overfitting.
Smart Grid Stability	L1	No	The emergence of an issue with underfitting suggested that the model had not adequately learned.
	L2	Yes	After the overfitting was eliminated, it reappeared again during learning.
	dropout	No	It improved continuously, eliminating overfitting.

As seen in Table 1, the dropout was successful in eliminating overfitting; however, there was a problem with randomness when nodes that would be significant in the classification process were dropped.

The algorithms random search and Bayesian algorithm utilized, the Tables 2 and 3 show the results of the best hyperparameters that were chosen after grantee preventing overfitting.

Table 2. Hyper-parameters that be chosen automatically

Bayesian Algorithm							
MINST Dataset				**Smart Grid Stability Dataset**			
Dropout	**Activation Function**	**Learning Rate**	**Number of Layers**	**Dropout**	**Activation Function**	**Learning Rate**	**Number of Layers**
yes	Leaky relu	0.0009	7	yes	Elu	0.0001	5
Random Search Algorithm							
MINST Dataset				**Smart Grid Stability Dataset**			
Dropout	**Activation Function**	**Learning Rate**	**Number of Layers**	**Dropout**	**Activation Function**	**Learning Rate**	**Number of Layers**
no	relu	0.0002	7	no	Elu	0.001	5

Table 3. Automatically chosen number of neurons and type of regularization methods in each layer

Layers	Bayesian Algorithm				Random Search			
	MINST Dataset		Smart Grid Stability Dataset		MINST Dataset		Smart Grid Stability Dataset	
	No. of Neurons	Regularization Type	No. of Neurons	Regularization Type	No. of Neurons	Regularization Type	No. of Neurons	Regularization Type
1	40	L1L2	488	L1L2	40	L1L2	392	L1L2
2	72	L2	104	L1L2	200	L2	360	L1L2
3	40	L1L2	488	L1	136	L2	488	L2
4	264	L1L2	136	L1L2	360	L2	296	L2
5	328	L1L2	40	L1L2	296	L1L2	296	L1L2
6	392	L1L2	–	–	360	L1L2	–	–
7	296	L2	–	–	264	L1L2	–	–

The results of using the optimization approaches, as given in Tables 2 and 3, undoubtedly overfitting is eliminated, and that happened by automatically selecting the best hyperparameters after several iterations. Based on the hyperparameters optimization method, some notes are discusses as follows:

1) The dropout hyperparameter has direct control over the number of nodes. As a result, it is clear that the nodes are fewer in the Bayesian model than there are in the Random search since the Bayesian model chose to employ the dropout among the selected hyperparameters.
2) It can see that there is a combination of L2 and L1 as a single hyperparameter done in some layers, where the L2 is typically more accurate than L1 and it also simpler to modify. L1 can, however, work with sparse feature areas and aids in feature selection.
3) The learning rate for the MINST dataset is lower when the optimization techniques are used than it is for the smart dataset. This is because the learning rate is lower when the algorithm has access to more data points, allowing for early large weight changes and later smaller changes or fine-tuning. In light of the fact that the photos required a little more work to achieve higher classification accuracy, the value of the learning rate hyperparameter is lower.

6.1 Comparison of proposal methods results

In this section, all the approaches used (L2, L1, dropout, random, and Bayesian) have been put to compare for both data (MINST) and (Smart grid stability) as in Figures 5 and 6.

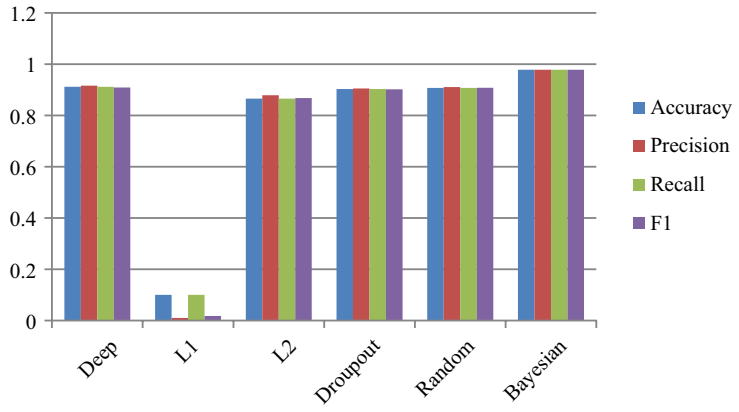


Fig. 5. Accuracy measurements for MINST dataset

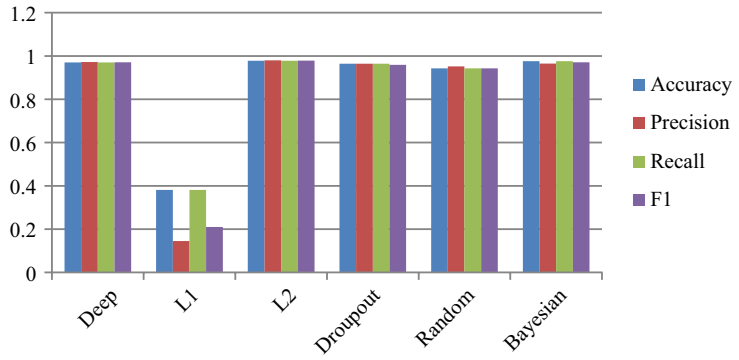


Fig. 6. Accuracy measurements for smart grid stability dataset

Through the previous two Figures 5 and 6. The accuracy value with the overfitting problem was equivalent to (91.16%) when the MINST image dataset was used. When it was treated using regularization techniques, the accuracy value with dropout reached its highest value which was (90.29%). Although it prevents overfitting, accuracy became less. However, when employing the suggested technique, accuracy while using Bayesian was (97.82%), and when using Random, it was (90.72%), in addition to fully eliminating overfitting.

While the accuracy without preventing the overfitting problem was (96%) when using the Smart Grid Stability dataset, the maximum accuracy obtained from L2 when utilizing regularization methods as it was equal to (95%). However, the overfitting issue was not entirely resolved, and the accuracy became less. While accuracy for the Bayesian and the random search increased to (96.5%) and (95.3%), respectively, when optimization approaches were used in addition to eliminating overfitting

6.2 Comparison by other works

To make the comparison more accessible, we have summarized what was mentioned in section 2 in Table 4.

Table 4. The summary of previous works

Ref. Year	NN Type	Optimization Algorithm	Overcome the Overfitting Problem	Results	Drawbacks
[9] 2018	CNN	data augmentation	Using redundant data with different rotations or colors or scaling	Decrees the overfitting and optimizes the accuracy.	The overfitting did not overcome completely
[4] 2020	CNN	No optimization method	Using Dropout, L1, and L2	cannot overcome overfitting completely	Even if it is slight, overfitting renders the accuracy unreliable.

(Continued)

Table 4. The summary of previous works *(Continued)*

Ref. Year	NN Type	Optimization Algorithm	Overcome the Overfitting Problem	Results	Drawbacks
[11] 2021	CNN RNN	Early stopping	Used stopping the training before completing all iterations	Prevent overfitting	risk of choosing the right time to stop
[12] 2021	ANN	PSO GA	Used Fewer nodes and manually calculated momentum rates in PSO and mutation rates in GA. Initial weight determined by the user	In the function minimization task, the PSO and the GA exhibit their capacity to identify the ideal parameter value.	Local minimum Arbitrarily large mutation slow speed
[13] 2021	ANN	Bayesian	Using L2	cannot overcome overfitting completely	falling in bad local minima
[14] 2021	DNN	No optimization method	Using L2, dropout	Overfitting cannot be entirely eliminated	To lessen overfitting, the cost function was modified using the L2 regularization technique. The dropout approach alters the network itself to reduce overfitting.
[15] 2022	DNN	Bayesian	dropout	Prevent overfitting completely	Their work required adding additional layers because They suffered from a lack of information due to random dropping

Based on the summary in Table 4 on earlier publications, using optimization algorithms or regularization techniques separately does not ensure the removal of overfitting. Compared to the proposed method, when using the regularization methods individually as a first step, as in previous works, the elimination of overfitting did not guarantee. However, when using optimization algorithms with the methods and considering the methods as hyperparameters, it ensures that there is no overfitting while also maintaining the high accuracy of the classification.

7 Conclusion

The major objective of this work is to find effective methods for preventing overfitting completely, in addition to determining how optimization techniques affect overfitting and its prevention. By selecting the proper hyper-parameters automatically and without human guidance, the optimization algorithms have demonstrated their value in preventing overfitting problems from arising in the model. Although the two techniques utilized used the optimal hyperparameters to ensure that the issue does not occur, Bayesian was able to demonstrate its significance in improving accuracy when compared to the Random search algorithm, as the accuracy increased by about (6%) in the first and (0.5%) in the second model.

8 References

[1] H. K. Jabbar and R. Z. Khan, "Methods to avoid over-fitting and under-fitting in supervised machine learning (comparative study)," no. December 2014, pp. 163–172, 2015, https://doi.org/10.3850/978-981-09-5247-1_017

[2] C. F. G. dos Santos and J. P. Papa, "Avoiding overfitting: A survey on regularization methods for convolutional neural networks," *ACM Comput. Surv.*, vol. 54, no. 10, 2022, https://doi.org/10.1145/3510413

[3] A. V. Miceli Barone, B. Haddow, U. Germann, and R. Sennrich, "Regularization techniques for fine-tuning in neural machine translation," *EMNLP 2017 – Conf. Empir. Methods Nat. Lang. Process. Proc.*, pp. 1489–1494, 2017, https://doi.org/10.18653/v1/D17-1156

[4] I. Marin, A. K. Skelin, and T. Grujic, "Empirical evaluation of the effect of optimization and regularization techniques on the generalization performance of deep convolutional neural network," *Appl. Sci.*, vol. 10, no. 21, pp. 1–30, 2020, https://doi.org/10.3390/app10217817

[5] C. Witt, "Worst-case and average-case approximations by simple randomized search heuristics," *Lect. Notes Comput. Sci.*, vol. 3404, no. Sfb 531, pp. 44–56, 2005, https://doi.org/10.1007/978-3-540-31856-9_4

[6] F. Hutter, *Meta-learning*, vol. 498. 2019.

[7] M. G. M. Abdolrasol *et al.*, "Artificial neural networks based optimization techniques: A review," *Electron*, vol. 10, no. 21, 2021, https://doi.org/10.3390/electronics10212689

[8] A. L. I. Muquri and S. Konstholm, "Data augmentation and related opportunity cost for managing the contemporary data sparsity data augmentation and related opportunity cost for managing the contemporary data sparsity," 2021.

[9] N. E. Khalifa, M. Hamed Taha, A. E. Hassanien, and I. Selim, "Deep galaxy V2: Robust deep convolutional neural networks for galaxy morphology classifications," *2018 Int. Conf. Comput. Sci. Eng. ICCSE 2018 – Proc.*, pp. 1–6, 2018, https://doi.org/10.1109/ICCSE1.2018.8374210

[10] C. Cranganu, M. E. Breaban, and H. Luchian, *Artificial intelligent approaches in petroleum geosciences*, 2015. https://doi.org/10.1007/978-3-319-16531-8

[11] P. Kumar, S. Batra, and B. Raman, "Deep neural network hyper-parameter tuning through twofold genetic approach," *Soft Comput.*, vol. 25, no. 13, pp. 8747–8771, 2021, https://doi.org/10.1007/s00500-021-05770-w

[12] L. Tani, D. Rand, C. Veelken, and M. Kadastik, "Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics.pdf" 2021. https://doi.org/10.1140/epjc/s10052-021-08950-y

[13] M. Miani *et al.*, "Bituminous mixtures experimental data modeling using a hyperparameters-optimized machine learning approach," *Appl. Sci.*, vol. 11, no. 24, 2021, https://doi.org/10.3390/app112411710

[14] M. Gupta, K. Rajnish, and V. Bhattacharjee, "Impact of parameter tuning for optimizing deep neural network models for predicting software faults," *Sci. Program.*, 2021, https://doi.org/10.1155/2021/6662932

[15] M. Masum *et al.*, "Bayesian hyperparameter optimization for deep neural network-based network intrusion detection," no. December, pp. 5413–5419, 2022, https://doi.org/10.1109/BigData52589.2021.9671576

[16] Terrence L. Fine, "Feedforward neural network methodology," *Feed. Neural Netw. Methodol.*, 1999, https://doi.org/10.1007/b97705

[17] N. Ketkar, "Convolutional neural networks," *Deep Learn. with Python*, pp. 63–78, 2017, https://doi.org/10.1007/978-1-4842-2766-4_5

[18] Y. Zhang, Y. Xie, Y. Zhang, J. Qiu, and S. Wu, "The adoption of deep neural network (DNN) to the prediction of soil liquefaction based on shear wave velocity," *Bull. Eng. Geol. Environ*, vol. 80, no. 6, pp. 5053–5060, 2021, https://doi.org/10.1007/s10064-021-02250-1

[19] M. A. A. Siddique, J. Ferdouse, M. T. Habib, M. J. Mia, and M. S. Uddin, "Convolutional neural network modeling for eye disease recognition," *Int. J. online Biomed. Eng.*, vol. 18, no. 9, pp. 115–130, 2022, https://doi.org/10.3991/ijoe.v18i09.29847

[20] Z. S. Kadhim, H. S. Abdullah, and K. I. Ghathwan, "Artificial neural network hyperparameters optimization: A survey," vol. 18, no. 15, pp. 59–87, 2022, https://doi.org/10.3991/ijoe.v18i15.34399

[21] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018, pp. 1–6, 2018, https://doi.org/10.1109/ICEngTechnol.2017.8308186

[22] B. Wu, Z. Liu, Z. Yuan, G. Sun, and C. Wu, "Reducing overfitting in deep convolutional neural networks using redundancy regularizer," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10614 LNCS, pp. 49–55, 2017, https://doi.org/10.1007/978-3-319-68612-7_6

[23] C. Series, "An overview of overfitting and its solutions an overview of overfitting and its solutions," 2019, https://doi.org/10.1088/1742-6596/1168/2/022022

[24] Jason Brownlee, "Better deep learning: Train faster, reduce overfitting, and make better ... – Jason Brownlee – google books," https://books.google.iq/books?hl=en&lr=&id=T1-nDwAAQBAJ&oi=fnd&pg=PP1&dq=reduce+overfitting+in+neural+networks&ots=tDTP-8mh1GT&sig=JzwbEU08hvT5olsYRlsAftWr5sE&redir_esc=y#v=onepage&q=reduce-overfitting in neural networks&f=false (accessed Oct. 21, 2022).

[25] Z. Liu, X. Li, B. Kang, and T. Darrell, "Regularization matters in policy optimization," pp. 1–44, 2019, [Online]. Available: http://arxiv.org/abs/1910.09191

[26] S. Gupta, R. Gupta, M. Ojha, and K. P. Singh, "A comparative analysis of various regularization techniques to solve overfitting problem in artificial neural network," *Commun. Comput. Inf. Sci.*, vol. 799, pp. 363–371, 2018, https://doi.org/10.1007/978-981-10-8527-7_30

[27] M. Feurer and F. Hutter, *Hyperparameter Optimization*, 2019. https://doi.org/10.1007/978-3-030-05318-5_1

[28] M. Claesen and B. De Moor, "Hyperparameter search in machine learning," pp. 10–14, 2015, [Online]. Available: http://arxiv.org/abs/1502.02127

[29] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[30] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. De Freitas, "Bayesian optimization in a billion dimensions via random embeddings," *J. Artif. Intell. Res.*, vol. 55, no. 1, pp. 361–367, 2016, https://doi.org/10.1613/jair.4806

[31] K. Eggensperger *et al.*, "Towards an empirical foundation for assessing bayesian optimization of hyperparameters," *BayesOpt Work. @ NeurIPS*, pp. 1–5, 2013.

[32] A. Tahseen Ali, H. S. Abdullah, and M. N. Fadhil, "Finger veins recognition using machine learning techniques," *Mater. Today Proc.*, 2021, https://doi.org/10.1016/j.matpr.2021.04.076

[33] A. Tahseen Ali, H. S. Abdullah, and M. N. Fadhil, "Voice recognition system using machine learning techniques," *Mater. Today Proc.*, 2022, https://doi.org/10.1016/j.matpr.2021.04.075

[34] A. T. Ali, H. S. Abdullah, and M. N. Fadhil, "Impostor detection based finger veins applying machine learning methods," *Iraqi J. Comput. Commun. Control Syst. Eng.*, vol. 21, no. 3, pp. 98–111, 2021, https://doi.org/10.33103/uot.ijccce.21.3.9

[35] A. S. Issa, Y. H. Ali, and T. A. Rashid, "An efficient hybrid classification approach for COVID-19 based on Harris Hawks optimization and Salp Swarm optimization," vol. 18, no. 13, pp. 113–130. https://doi.org/10.3991/ijoe.v18i13.33195

9 Authors

Zahraa Saddi Kadhim: she obtained her B.Sc. in computer science from University of Technology, Iraq in 2015. Currently studying M.Sc. in computer science. she worked as a trainer for a non-governmental group teaching young people how to code robots in 2017–2019 her interested is mobile application, Artificial Intelligence, Machine Learning, Pattern Recognition, Data encryption and robotic (email: cs.20.44@grad.uotechnology.edu.iq).

Hasanen S. Abdullah: Assist Professor qualified to direct research at University of Technology, Iraq, and other Iraqi Institutions. He got his B.Sc., M.Sc. and Ph.D. in computer science from University of Technology, Iraq in 2000, 2004 and 2008 respectively. His area of interests is Artificial Intelligence, Machine Learning, Swarm Intelligence, Pattern Recognition, Data Mining & warehouse, and Business Intelligence (email: Hasanen.S.Abdullah@uotechnology.edu.iq).

Khalil I. Ghathwan PhD/M.Sc./HD (AI)/B.S./Dip: Senior University of Technology – Iraq Computer sciences. His Skills Computer Science Education Computer Programming NET technology, research interest is broadly in Science in Computing (Artificial Intelligence (AI)/Machine Learning (ML)), Computer Network, Security and data mining algorithm. This includes Routing, optimization and swarm intelligence. Currently, he involved in several projects relating to artificial Intelligence and routing algorithm (email: Khalil.i.ghathwan@uotechnology.edu.iq).

Article submitted 2022-12-13. Resubmitted 2023-01-27. Final acceptance 2023-02-06. Final version published as submitted by the authors.