Enhancing Accuracy for Classification Using the CNN Model and Hyperparameter Optimization Algorithm

Dai Nguyen Quoc[1], Ngoc Tran Thanh[2]
[1,2]Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, Vietnam

Article Info

Article history:

Received Apr 21, 2024
Revised Sep 4, 2024
Accepted Sep 11, 2024

Keywords:

CNN
RS
BO-GP
BO-TPE

ABSTRACT

The Convolutional Neural Network (CNN) is a widely employed deep learning model, particularly effective for image recognition and classification tasks. The performance of a CNN is influenced not only by its architecture but also critically by its hyperparameters. Consequently, optimizing hyperparameters is essential for improving CNN model performance. In this study, the authors propose leveraging optimization algorithms such as Random Search, Bayesian Optimization with Gaussian Processes, and Bayesian Optimization with Tree-structured Parzen Estimators to fine-tune the hyperparameters of the CNN model. The performance of the optimized CNN is compared with traditional machine learning models, including Random Forest (RF), Support Vector Classification (SVC), and K-Nearest Neighbors (KNN). Both the MNIST and Olivetti Faces datasets are utilized in this research. In the training procedure, on the MNIST dataset, the CNN model achieved a minimum accuracy of 97.85%, surpassing traditional models, which had a maximum accuracy of 97.50% across all optimization techniques. Similarly, on the Olivetti Faces dataset, the CNN achieved a minimum accuracy of 94.96%, while traditional models achieved a maximum accuracy of 94.00%. In the training-testing procedure, the CNN demonstrated impressive results, achieving accuracy rates exceeding 99.31% on the MNIST dataset and over 98.63% on the Olivetti Faces dataset, significantly outperforming traditional models, whose maximum values were 98.69% and 97.50%, respectively. Furthermore, the study compares the performance of the CNN model with three optimization algorithms. The results show that integrating CNN with these optimization techniques significantly improves prediction accuracy compared to traditional models.

Corresponding Author:

Tran Thanh Ngoc
Faculty of Electrical Engineering Technology
Industrial University of Ho Chi Minh City
12 Nguyen Van Bao, Ward 4, Go Vap District, Ho Chi Minh City, Vietnam
Email: tranthanhngoc@iuh.edu.vn

1. INTRODUCTION

Machine learning, a subset of Artificial Intelligence, gives computers the ability to learn and improve on their own from experience, without needing explicit programming. This field focuses on creating computer programs that can extract knowledge from data and use this information to make better decisions in the future. Machine learning has a wide range of applications, impacting areas like computer vision and image recognition, natural language processing, recommendation systems, and fraud detection [1–4].

Deep learning, an advanced technique within machine learning, extends this concept by utilizing artificial neural networks with multiple layers to learn complex patterns in large amounts of data. One of the most widely used architectures within deep learning is Convolutional Neural Network (CNN), which are especially effective in computer vision tasks such as image recognition and classification [5–6]. CNN consist of multiple layers, including convolutional, pooling, and fully connected layers. Convolutional layers use filters to process input data and extract relevant features, while pooling layers reduce the spatial dimensions of the data. Fully connected layers then utilize these features to make predictions or classifications. Through the hierarchical structure of CNN, they automatically learn hierarchical representations of data, beginning with simple features and progressively advancing to more complex and abstract features. This capability makes CNN highly effective in tasks like object detection, facial recognition, and medical image analysis [7–10].

The performance of CNN models is significantly influenced by hyperparameters, which are set before training and are not learned during the process. Hyperparameters in CNNs are divided into two categories: architecture-related parameters (e.g., number of layers, filter size, pooling size) and training-related parameters (e.g., learning rate, number of epochs, batch size) [11–12]. Optimizing these hyperparameters is crucial for maximizing CNN performance. Optimization algorithms enhance machine learning and deep learning models by fine-tuning hyperparameters and model parameters [13]. These algorithms are typically classified into three categories: unstructured (e.g., Grid Search (GS) and Random Search (RS)) [14–15], structured (e.g., Bayesian Optimization with Gaussian Processes (BO-GP) and Tree-structured Parzen Estimators (BO-TPE)) [16–17], and metaheuristic approaches (e.g., Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Grey Wolf Optimizer (GWO)) [18–19]. Each category offers distinct advantages in addressing complex, high-dimensional optimization problems.

In this study, the authors propose using optimization techniques, including Random Search, Bayesian Optimization with Gaussian Processes, and Tree-structured Parzen Estimators, to fine-tune the hyperparameters of CNN models. Although these techniques are relatively straightforward, they are highly effective for optimizing the hyperparameters of neural networks. The performance of the optimized CNN model will be compared with traditional machine learning models, such as Random Forest (RF), Support Vector Classification (SVC), and K-Nearest Neighbors (KNN). To ensure the reliability of the findings, this study will use two datasets: MNIST and Olivetti Faces.

The structure of the paper is organized as follows: Section 2 introduces CNN model and the research method. Section 3 presents the results and discussion. Finally, Section 4 concludes the paper.

2. RESEARCH METHOD

2.1. Convolutional neural network

The CNN is one of the most widely used deep learning models in computer vision, particularly in image recognition and classification tasks. A typical structure of a basic CNN model, consisting of three layers: convolutional, pooling, and fully connected, is illustrated in Figure 1 below. The convolutional layer, the core building block of a CNN, applies a set of filters to the input data to create a feature map. Pooling layers then summarize the data, reducing its size and consequently the number of parameters and the computational complexity of the model. Finally, the fully connected layers perform high-level reasoning tasks [6], [20].

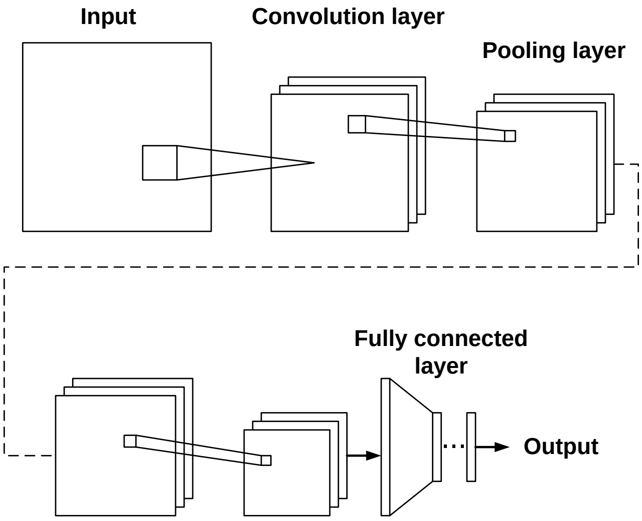


Figure 1. The configuration of a basic CNN model

Similar to other machine learning models, hyperparameters play a crucial role in optimizing the performance of CNN. Hyperparameters are parameters that are set before the training process begins and remain constant throughout. They tune the model's architecture and behavior, affecting its accuracy, efficiency, and generalization ability. The basic hyperparameters of the CNN model used in this study are [12], [21–22]:

Filters: The number of filters is a crucial parameter in CNN, dictating the number of feature maps learned by the convolutional layers. This parameter determines the number of output channels produced by the convolution operation. Typical values for setting the number of filters in a CNN are 32, 64, 128, and so on.

Batch size: The batch size is a hyperparameter of gradient descent that determines the number of training samples used to update the model's internal parameters. It also plays a role in optimizing memory usage and influencing training speed.

Epochs: The number of epochs is a crucial hyperparameter in CNN training that controls how many times the entire training dataset is passed through the network. This parameter significantly impacts the model's ability to learn and generalize well.

Optimizer: Optimizers are techniques or methods for adjusting the characteristics of a neural network, such as weights and learning rates, to decrease losses. They are employed to solve optimization problems by minimizing the loss function. Some commonly used optimization algorithms include SGD (Stochastic Gradient Descent), RMSProp (Root Mean Square Propagation), Adam, and Adamax.

Activation: The activation function in neural networks determines how the weighted sum of inputs is transformed into output from a single node or multiple nodes in a layer of the network. It is responsible for introducing non-linearity into the network and enabling it to model complex patterns and relationships in the data. Commonly used activation functions include ReLU, Tanh, Softmax, Linear, and others.

2.2. Hyperparameter optimization techniques

There are many hyperparameter optimization algorithms applicable to machine learning models, which can be categorized into classes such as: Model-free Algorithms, including Grid Search and Random Search; Bayesian Optimization, including Bayesian Optimization with the Gaussian process and Bayesian Optimization with the Tree-structured Parzen estimator; Multi-fidelity Optimization Algorithms, including Successive Halving, Hyperband, and Genetic Algorithms; Metaheuristic Algorithms, including Genetic Algorithm and Particle Swarm Optimization [13], [18], [23–24].

In this study, the authors propose using Hyperparameter Optimization Algorithms (HPO), including Random Search, Bayesian Optimization with the Gaussian process, and Bayesian Optimization with the Tree-structured Parzen estimator, to optimize the hyperparameters of the CNN model.

Random Search:

Random Search (RS) is a widely used optimization algorithm in machine learning for tuning hyperparameters, classified under the category of unstructured optimization methods. Unlike Grid Search, which exhaustively evaluates all possible combinations of predefined hyperparameter values, RS randomly samples values from a defined range for each hyperparameter. This approach significantly reduces runtime, making it more efficient for large datasets or problems with numerous hyperparameters. However, because the selection of hyperparameter combinations is random, RS may not always find the absolute optimal value [23]. Additionally, Random Search does not utilize information from previous results to guide future searches. Instead, it performs searches in a relatively arbitrary manner, which can be less efficient when navigating large search spaces. As a result, RS is typically employed for optimization tasks within smaller, more manageable ranges [15], [25].

Bayesian Optimization:

Bayesian Optimization (BO) has gained significant popularity in recent years due to its effectiveness in handling functions that are expensive to evaluate, which are common in machine learning. BO iteratively builds a probabilistic model of the objective function and intelligently selects points for evaluation, aiming to find the global minimum (or maximum) with minimal evaluations. Bayesian optimization consists of two main components: surrogate models for modeling the objective function and an acquisition function that measures the potential value from evaluating the objective function at a new point [26-27].

Bayesian optimization models come in different flavors, including Bayesian Optimization with the Gaussian process (BO-GP) and Bayesian Optimization with the Tree-structured Parzen estimator (BO-TPE). In BO-GP algorithms, Gaussian processes (GP) have become the standard surrogate for modeling the objective function in Bayesian optimization. BO-GP mainly supports continuous and discrete hyperparameters (by rounding them) but does not support conditional hyperparameters [17, 28]. Meanwhile, BO-TPE does not define a predictive distribution over the objective function but creates two density functions that act as generative models for all domain variables. BO-TPE can handle categorical, discrete, continuous, and conditional hyperparameters [23, 29-31].

2.3. Method proposed

In this study, which aims to enhance the accuracy of deep learning models in solving classification problems, the authors propose the application of Hyperparameter Optimization (HPO) for the CNN network, as depicted in Figure 2 below. According to this flowchart, the entire dataset (X and y) are used during the

HPO training process. The structure of the CNN model and the search space for hyperparameters are defined, serving as inputs to the training process. The output of this process is the optimized model ($model_{opt}$), which corresponds to the optimized hyperparameters, and the accuracy associated with this optimized model. To evaluate the model's effectiveness, this study employs the accuracy metric, which is determined using Formula (1) below [32]. Comparing and analyzing the accuracy values of the CNN model with other traditional models such as RF, SVC, and KNN allows us to assess their performance.

$$Accuracy(y,\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} 1(\hat{y}_i = y_i) \tag{1}$$

where $\hat{y}$ is the predicted value of the ith sample and y is the corresponding true value, and n is the number of samples.

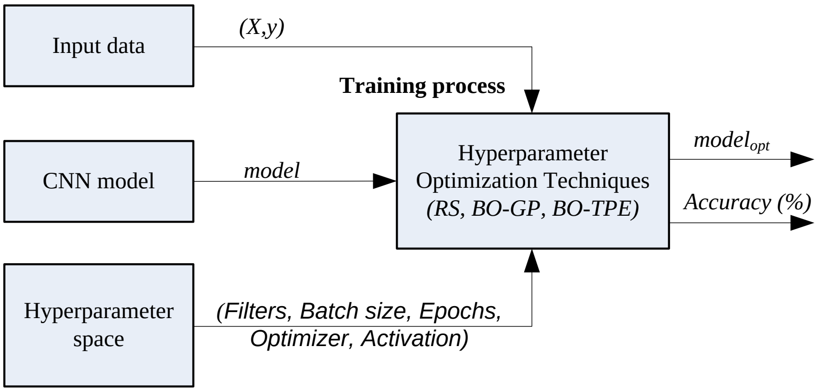


Figure 2. The training procedure

The distinctive feature of the training procedure described above is that it utilizes the entire dataset X and y during the training process, and the accuracy of the optimized models is determined based on this procedure. To enhance confidence in evaluating the model's performance, the authors also propose applying a training-testing procedure as depicted in the flowchart in Figure 3 below. According to this flowchart, the dataset is split into training data and testing data. The training data (X_{train}, y_{train}) is used for the training process, while the testing data (X_{test}, y_{test}) is used for the testing process, highlighting the key difference between the training procedure and the training-testing procedure. In the training process, the input is the training data, and the output is the optimized model ($model_{opt}$), which also serves as the input to the testing process. Another input for the testing process is the testing data. In this phase, the predicted values $y_{predict}$ are determined based on the optimized model and X_{test}. The accuracy of the testing process is determined by comparing the true values y_{test} with $y_{predict}$. Thus, the outputs of the training-testing procedure are the accuracy rates of both the training and testing processes. It's worth noting that the testing dataset in this training-testing procedure is independent of the training dataset mentioned earlier; hence, its prediction results will help increase confidence in evaluating the performance of the CNN model as well as the HPO.

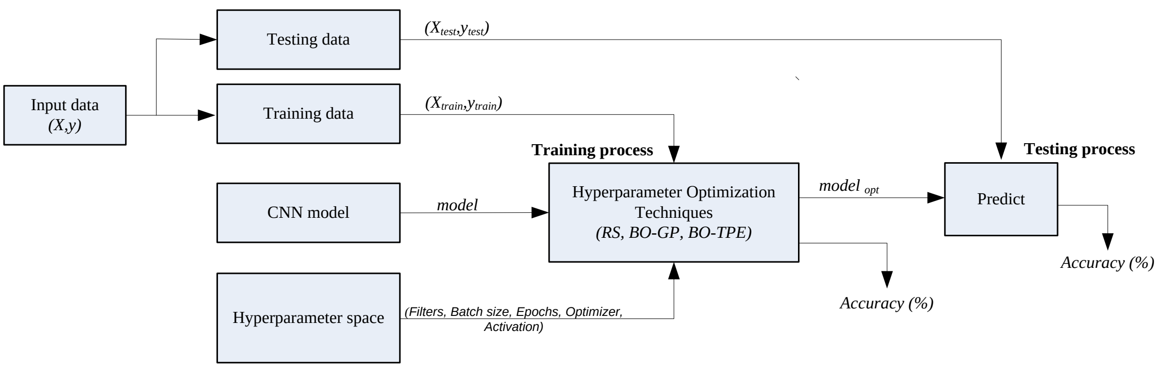
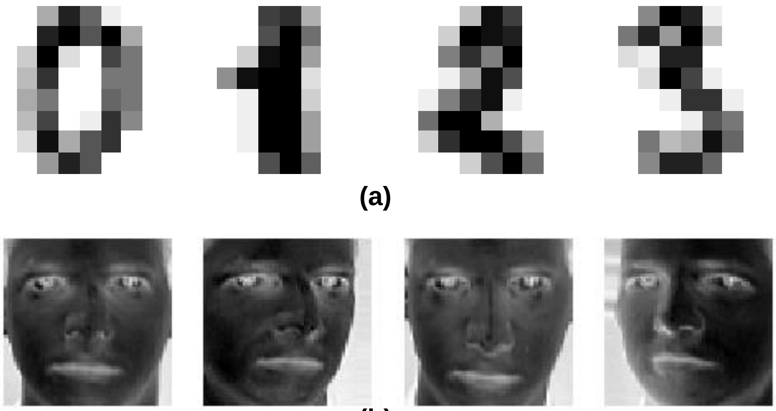


Figure 3. The training-testing procedure

3. RESULTS AND DISCUSSION

3.1. Experimental setup

In this study, the authors concurrently utilize two datasets: the MNIST dataset, which is a collection of handwritten digit images [23], and the Olivetti Faces dataset, which consists of facial images [33]. These datasets are widely used in machine learning research for recognition and classification tasks and are readily available in the sklearn.datasets library. Figure 4 presents some samples from the MNIST dataset (a) and the Olivetti Faces dataset (b). As depicted in Figure 2, the entire dataset (X, y) is used for the HPO algorithm in the training procedure. As shown in Figure 3, the dataset (X, y) is divided into two parts: training data (X_{train}, y_{train}) and testing data (X_{test}, y_{test}), with a ratio of 80% and 20% for each part, respectively. One of the inputs to the HPO is the Hyperparameter space, which defines the boundaries of the hyperparameters being searched. In this study, the Hyperparameter space is defined consistently for both the training procedure in Figure 2 and the training-testing procedure in Figure 3. The Hyperparameter space for the MNIST dataset is presented in Table 1, and for the Olivetti Faces dataset in Table 2. The HPO techniques in this paper (RS, BO-GP, and BO-TPE) are also set up based on a cross-validation procedure with cv=3 to enhance the reliability of the results [23].



(a)

(b)

Figure 4. Some samples: (a) MNIST dataset, (b) Faces dataset

Table 1. Hyperparameter space: Mnist dataset

Model	Hyperparameter	Type	Optimal algorithms RS	BO-GP	BO-TPE
RF	n_estimators	Integer	(10, 100)	(10, 100)	(10, 100)
	max_features	Integer	(1, 64)	(1, 64)	(1, 64)
	max_depth	Integer	(5, 50)	(5, 50)	(5, 50)
	min_samples_split	Integer	(2, 11)	(2, 11)	(2, 11)
	min_samples_leaf	Integer	(1, 11)	(1, 11)	(1, 11)
	criterion	String	['gini', 'entropy']	['gini', 'entropy']	['gini', 'entropy']
SVC	C	Float	(0, 50)	(0.01, 50)	(0, 50)
	kernel	String	['linear', 'poly', 'rbf', 'sigmoid']	['linear', 'poly', 'rbf', 'sigmoid']	['linear', 'poly', 'rbf', 'sigmoid']
KNN	n_neigHPOrs	Integer	(1, 20)	(1, 20)	(1, 20)
CNN	optimizer	String	['adam', 'rmsprop', 'sgd', 'Adamax']	['adam', 'rmsprop', 'sgd', 'Adamax']	['adam', 'sgd', 'rmsprop', 'Adamax']
	activation	String	['relu', 'tanh', 'softmax', 'linear']	['relu', 'tanh' , 'softmax', 'linear']	['relu', 'tanh', 'sigmoid', 'linear']
	batch_size	Integer	(16, 128)	(16, 128)	(16, 128)
	filters	Integer	(10, 300)	(10, 300)	(10, 300)
	epochs	Integer	(200, 800)	(200, 800)	(200, 800)

Table 2. Hyperparameter space: Olivetti Faces dataset

Model	Hyperparameter	Type	Optimal algorithms RS	BO-GP	BO-TPE
RF	n_estimators	Integer	(10, 200)	(10, 200)	(10, 200)
	max_features	Integer	(1, 128)	(1, 128)	(1, 128)
	max_depth	Integer	(5, 100)	(5, 100)	(5, 100)
	min_samples_split	Integer	(2, 11)	(2, 11)	(2, 11)
	min_samples_leaf	Integer	(1, 11)	(1, 11)	(1, 11)
	criterion	String	['gini', 'entropy']	['gini', 'entropy']	['gini', 'entropy']
SVC	C	Float	(0, 100)	(0.01, 100)	(0, 100)
	kernel	String	['linear', 'poly', 'rbf', 'sigmoid']	['linear', 'poly', 'rbf', 'sigmoid']	['linear', 'poly', 'rbf', 'sigmoid']
KNN	n_neigHPOrs	Integer	(1, 15)	(1, 15)	(1, 15)
CNN	filters	Integer	(16, 64)	(16, 64)	(16, 64)
	optimizer	String	['adam', 'rmsprop', 'sgd', 'Adamax']	['adam', 'rmsprop', 'sgd', 'Adamax']	['adam', 'rmsprop', 'sgd', 'Adamax']
	activation	String	['relu', 'tanh', 'softmax', 'linear']	['relu', 'tanh', 'softmax', 'linear']	['relu', 'tanh', 'softmax', 'linear']
	epochs	Integer	(60, 120)	(60, 120)	(60, 120)
	batch_size	Integer	(8, 32)	(8, 32)	(8, 32)

3.2. Experimental results

Table 3 presents the experimental results of the training procedure as shown in Figure 2, depicting the accuracy of the RS, BO-GP, and BO-TPE algorithms. These results correspond to the RF, SVC, KNN, and CNN models for the MNIST and Olivetti Faces datasets. It is worth noting that the accuracy values for the traditional machine learning models (RF, SVC, and KNN) for the MNIST dataset are reference values from the literature [23], as given in Table 3.

Table 3. Experimental results of training procedure

Optimal algorithm	Mnist dataset				Olivetti Faces dataset			
	RF	SVC	KNN	CNN	RF	SVC	KNN	CNN
RS	93.38	97.35	96.33	97.85	91.25	94.00	90.26	94.96
BO-GP	93.38	97.50	96.83	97.89	92.25	94.00	90.26	95.04
BO-TPE	93.88	97.44	96.83	97.89	90.75	94.00	90.26	94.96

The analysis of results in Table 3 demonstrates that the accuracy of the CNN deep learning model surpasses that of traditional models. Specifically, for the MNIST dataset using the RS optimization algorithm, the accuracy values for the traditional models are as follows: Random Forest (RF) = 93.38%, Support Vector Classification (SVC) = 97.35%, and K-Nearest Neighbors (KNN) = 96.33%. All of these are lower than the CNN model's accuracy of 97.85. Moreover, when comparing the CNN model combined with the BO-GP and BO-TPE algorithms, the performance is superior to that achieved with RS. The accuracy values for BO-TPE (97.89%) and BO-GP (97.89%) are both higher than that for RS (97.85%). Similar results are obtained for the BO-GP and BO-TPE algorithms, including the Olivetti Faces dataset.

Table 4 presents the experimental results of the training-testing procedure, as shown in Figure 3 for the MNIST dataset. It illustrates the accuracy scores the RS, BO-GP, and BO-TPE algorithms achieved. These scores correspond to the RF, SVC, KNN, and CNN models for the training and testing processes. Similarly, Table 5 showcases analogous results for the Olivetti Faces dataset.

Table 4. Experimental results of training-testing procedure: Mnist dataset

Optimal algorithm	Training process				Testing process			
	RF	SVC	KNN	CNN	RF	SVC	KNN	CNN
RS	96.02	98.82	98.40	99.18	97.17	98.64	98.33	99.31
BO-GP	96.21	98.80	98.43	99.14	97.36	98.69	98.17	99.39
BO-TPE	95.80	98.82	98.35	99.20	97.33	98.64	98.19	99.31

Table 5. Experimental results of training-testing procedure: Olivetti Faces dataset

Optimal algorithm	Training process				Testing process			
	RF	SVC	KNN	CNN	RF	SVC	KNN	CNN
RS	84.78	92.50	86.41	93.56	90.13	97.50	91.00	98.63
BO-GP	84.57	92.44	84.40	93.47	90.37	97.37	89.62	98.75
BO-TPE	85.60	92.50	81.31	93.81	91.38	97.50	87.13	98.63

An analysis of the results from Table 4 for the MNIST dataset using the RS optimization algorithm shows that the accuracy of the traditional models is consistently lower than that of the CNN model in both the training and testing process. Specifically, the traditional models achieve the following accuracy rates: RF = 96.02% (training) and 97.17% (testing), SVC = 98.82% (training) and 98.64% (testing), and KNN = 98.40% (training) and 98.33% (testing). In contrast, the CNN model attains higher accuracy rates of 99.18% during training and 99.31% during testing. When examining the CNN model integrated with different optimization algorithms, the BO-TPE algorithm achieves the highest training accuracy of 99.20%, slightly surpassing the RS algorithm's 99.18%. However, the BO-GP algorithm underperforms in training, with an accuracy of 99.14%. In the testing phase, the CNN model with the BO-GP algorithm achieves the highest accuracy of 99.39%, exceeding that of the RS algorithm (99.31%). In contrast, the BO-TPE algorithm shows no significant improvement, reaching 99.31%. These trends are similarly observed for the Olivetti Faces dataset, as shown in Table 5, further reinforcing the conclusion that the CNN model generally outperforms traditional models, particularly when optimized with the BO-GP algorithm.

Overall, the above analysis across the MNIST and Olivetti Faces datasets consistently demonstrates that the CNN deep learning model outperforms traditional machine learning models such as RF, SVC, and KNN in terms of accuracy. This trend is evident in the training and training-testing procedure for all optimization algorithms evaluated (RS, BO-GP, and BO-TPE).

4. CONCLUSION

In this study, the authors propose a novel approach using RS, BO-GP, and BO-TPE algorithms to optimize the hyperparameters of a CNN model, aiming to enhance its performance. The effectiveness of both the training and training-testing procedures is assessed using the MNIST and Olivetti Faces datasets. The performance of the CNN model, optimized with the RS, BO-GP, and BO-TPE algorithms, is compared to the results reported in [23] during the training phase. Additionally, the CNN model's performance in the training-testing phase is evaluated against traditional models across all three optimization algorithms, including RF, SVC, and KNN. The findings demonstrate that the CNN model consistently achieves higher accuracy than traditional models in all evaluated scenarios, highlighting its potential for various applications in image recognition and classification tasks.

REFERENCES

[1] I. H. Sarker, "Machine Learning: Algorithms, Real-World Applications and Research Directions", *SN Computer Science*, Vol. 2, No. 3, pp. 160, 2021.

[2] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving", *IATSS Research*, Vol. 43, No. 4, pp. 244-252, 2019.

[3] W. J. Wong and S. H. Lai, "Multi-task CNN for restoring corrupted fingerprint images", *Pattern Recognition*, Vol. 101, pp. 107203, 2020.

[4] H. H. Luong, T. T. Khanh, M. D. Ngoc, M. H. Kha, K. T. Duy, and T. T. Anh, "Detecting Exams Fraud Using Transfer Learning and Fine-Tuning for ResNet50", *In: Communications in Computer and Information Science, Ho Chi Minh City, Vietnam*, Vol. 1688, pp. 747-754, 2022.

[5] M. M. Taye, "Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions", *Computation*, Vol. 11, No. 3, pp. 52, 2023.

[6] S. Almabdy and L. Elrefaei, "Deep Convolutional Neural Network-Based Approaches for Face Recognition", *Applied Sciences (Switzerland)*, Vol. 9, No. 20, pp. 4397, 2019.

[7] D. Beohar and A. Rasool, "Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network (ANN) and CNN", *In: International Conference on Emerging Smart Computing and Informatics (ESCI)*, Pune, India, pp. 542-548, 2021.

[8] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra and J. M. Z. Maningo, "Object Detection Using Convolutional Neural Networks", *In: IEEE Region 10 Annual International Conference*, Proceedings TENCON, Jeju, Korea (South), pp. 2023-2027, 2018.

[9] S. Kumaar, R. M. Vishwanath, S. N. Omkar, A. Majeedi and A. Dogra, "Disguised Facial Recognition Using Neural Networks", *In: 2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP), Shenzhen, China*, pp. 28-32, 2018.

[10] S. M. Anwar, M. Majid, A. Qayyum, M. Awais, M. Alnowami, and M. K. Khan, "Medical Image Analysis using Convolutional Neural Networks: A Review", *Journal of Medical Systems,* Vol. 42, No. 11, pp. 226, 2018.

[11] R. Lateef and A. Abbas, "Tuning the Hyperparameters of the 1D CNN Model to Improve the Performance of Human Activity Recognition", *Engineering and Technology Journal*, Vol. 40, No. 4, pp. 547-554, 2022.

[12] TN Tran, "Grid Search of Convolutional Neural Network model in the case of load forecasting", *Archives Of Electrical Engineering*, Vol. 70, No. 1, pp. 25-36, 2021.

[13] M. A. K. Raiaan et al., "A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks," Decision Analytics Journal, vol. 11, p. 100470, Jun. 2024, doi: 10.1016/J.DAJOUR.2024.100470.

[14] Y. Zhao, W. Zhang, and X. Liu, "Grid search with a weighted error function: Hyper-parameter optimization for financial time series forecasting," *Applied Soft Computing*, vol. 154, p. 111362, Mar. 2024, doi: 10.1016/J.ASOC.2024.111362.

[15] N. Sharma, L. Malviya, A. Jadhav, and P. Lalwani, "A hybrid deep neural net learning model for predicting Coronary Heart Disease using Randomized Search Cross-Validation Optimization," *Decision Analytics Journal*, vol. 9, p. 100331, Dec. 2023, doi: 10.1016/J.DAJOUR.2023.100331.

[16] D. Shakya, V. Deshpande, M. J. S. Safari, and M. Agarwal, "Performance evaluation of machine learning algorithms for the prediction of particle Froude number (Frn) using hyper-parameter optimizations techniques," Expert Systems with Applications, vol. 256, p. 124960, Dec. 2024, doi: 10.1016/J.ESWA.2024.124960.

[17] H. Sadoune, R. Rihani, and F. S. Marra, "DNN model development of biogas production from an anaerobic wastewater treatment plant using Bayesian hyperparameter optimization," Chemical Engineering Journal, vol. 471, p. 144671, Sep. 2023, doi: 10.1016/J.CEJ.2023.144671.

[18] A. Sabouri and C. S. Perez-Martinez, "Design of electrostatic lenses through genetic algorithm and particle swarm optimisation methods integrated with differential algebra," Ultramicroscopy, vol. 266, p. 114024, Dec. 2024, doi: 10.1016/J.ULTRAMIC.2024.114024.

[19] P. K. Chahal, K. Kumar, and B. S. Soodan, "Grey wolf algorithm for cost optimization of cloud computing repairable system with N-policy, discouragement and two-level Bernoulli feedback," *Mathematics and Computers in Simulation*, vol. 225, pp. 545–569, Nov. 2024, doi: 10.1016/J.MATCOM.2024.06.005.

[20] N. M. Aszemi and P. D. D. Dominic, "Hyperparameter optimization in convolutional neural network using genetic algorithms," *International Journal of Advanced Computer Science and Applications*, Vol. 10, No. 6, pp. 269-278, 2019.

[21] R. Zatarain Cabada, H. Rodriguez Rangel, M. L. Barron Estrada, and H. M. Cardenas Lopez, "Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems," *Soft Computing*, Vol. 24, No. 10, pp. 7593-7602, 2020.

[22] K. and N. R. O'Shea, "An Introduction To Convolutional Neural Networks", *International Journal for Research in Applied Science and Engineering Technology*, Vol. 10, No. 12, 2015.

[23] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, Vol. 415, pp. 295-316, 2020.

[24] A. Morales-Hernández, I. van Nieuwenhuyse, and S. Rojas Gonzalez, "A survey on multi-objective hyperparameter optimization algorithms for machine learning," *Artificial Intelligence Review*, Vol. 56, No. 8, pp. 8043-8093, 2023.

[25] W. Zhang, C. Wu, H. Zhong, Y. Li, and L. Wang, "Prediction of undrained shear strength using extreme gradient boosting and random forest based on Bayesian optimization," Geoscience Frontiers, vol. 12, no. 1, pp. 469–477, Jan. 2021, doi: 10.1016/J.GSF.2020.03.007.

[26] Frazier, P. I., "A Tutorial on Bayesian Optimization", Art. no. arXiv:1807.02811, 2018. doi:10.48550/arXiv.1807.02811.

[27] J. Rodemann and T. Augustin, "Imprecise Bayesian optimization," *Knowledge-Based Systems*, vol. 300, p. 112186, Sep. 2024, doi: 10.1016/J.KNOSYS.2024.112186.

[28] E. Brochu, V. M. Cora, and N. D. Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *ArXiv*, vol. abs/1012.2599, 2010.

[29] K. Eggensperger, M. Feurer, F. Hutter, J. Bergstra, J. Snoek, H. Hoos, and K. Leyton-Brown, "Towards an empirical foundation for assessing Bayesian optimization of hyperparameters," *BayesOpt Work.*, pp. 1–5, 2013.

[30] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Adv. Neural Inf. Process. Syst.*, vol. 24, 2011.

[31] K. Shen, H. Qin, J. Zhou, and G. Liu, "Runoff probability prediction model based on natural gradient boosting with tree-structured parzen estimator optimization," *Water*, vol. 14, no. 4, p. 545, 2022. [Online]. Available: https://doi.org/10.3390/w14040545.

[32] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, Jan. 2009, doi: 10.1016/J.PATREC.2008.08.010.

[33] W. Dhifli and A. B. Diallo, "Face Recognition in the Wild," *Procedia Computer Science*, Vol. 96, pp. 1571-1580, 2016.