Mathematical Statistician and Engineering Applications
ISSN: 2094-0343
DOI: https://doi.org/10.17762/msea.v70i2.2327

Optimizing Hyperparameters for MINIST Dataset Classification using Deep Learning Techniques

Tabish Rao

Asst. Professor, School of Computing, Graphic Era Hill University, Dehradun, Uttarakhand India 248002

Article Info
Page Number: 1353-1361
Publication Issue:
Vol. 70 No. 2 (2021)

Abstract
The potential of deep learning to improve the performance of image classification has been demonstrated. In this paper, we present a study on the optimization of the hyperparameters for the classification of the MNIST dataset. We performed a comparison between a standard two-layer perceptron model and a CNN model with different techniques. The optimized hyperparameters for CNN are based on the number of filters, kernel size, and convolutional layers. The optimized CNN model performed better than the default model on the classification task of the MNIST dataset. The various hyperparameters included the learning rate, batch size, the number of hidden layers, the dropout rate, the activation function, and the optimizer. The optimized CNN model was able to achieve an accuracy of 99% on a test, which is significantly better than the 96% accuracy of the default model. The difference between the two models is that the former takes longer to train and has a slightly longer time per image. The study demonstrates the importance of optimizing the hyperparameters in deep learning-focused classification tasks. The findings show how CNN architectures perform well in these applications, and it shows how optimizing these components can yield superior results. These recommendations can help further develop efficient and accurate models for this field.

Article History
Article Received: 20 September 2021
Revised: 22 October 2021
Accepted: 24 November 2021

Keywords: Hyperparameter, Deep learning, Deep neural network, CNN, MNIST.

Introduction

Due to the potential of deep learning to solve various complex problems, such as image classification, it has been widely used in the development of systems. However, it is important to optimize the models to ensure that they are capable of achieving high accuracy. This process involves selecting the optimal parameters that are used in the model's behavior. Some of these include the learning rate, batch sizes, number of hidden features, and the optimizer. This paper aims to analyze the effects of optimizing the parameters of deep learning models on the performance of image classification with the help of the MNIST dataset[1]. This is a well-known dataset that is used for benchmarking. We will study the effects of this optimization on the efficiency and accuracy of the deep learning models[2], [3].

The paper aims to compare and optimize the performance of two-layer and three-layer CNN models. We will use various optimization techniques to analyze the data and determine the optimal values for each parameter. For instance, the learning rate, the number of hidden features per layer, the dropout rate, the activation function, and the batch size are analyzed. The paper's introductory section covers the topic of the problem statement and emphasizes the

importance of deep learning's hyperparameter optimization. We then discuss how the chosen hyperparameters influence deep learning's performance, as well as how they can help improve the efficiency of the models.

The paper's main body is divided into three parts. The first one covers the methodology for training and optimizing deep learning models. It provides an extensive explanation of the selected hyperparameters, as well as the optimization techniques that are used. The second one talks about the CNN and MLP models' architecture. The second section of the paper presents the results of the tests and training of the CNN and the MLP models. We also discuss the efficiency and accuracy metrics of these models, such as training time, validation time, test accuracy, inference time, and training time. We will compare the CNN model's efficiency and accuracy with that of the default model and analyze the effects of hyperparameter optimization[4].

The third section of the paper presents an extensive analysis of the results, and it draws conclusions based on the findings. In addition, it talks about the optimal values of the hyperparameters of the CNN model, as well as the factors that contributed to its improvement. We also discuss the paper's limitations, as well as the future directions that we can take[5], [6].

The paper presents an overview of the importance of optimizing the parameters of a deep learning model in order to improve its efficiency and accuracy in image classification. It also states that the CNN architecture and the optimization techniques implemented can help improve the MNIST dataset's classification accuracy. The paper's findings provide valuable insights into the development of efficient and accurate deep learning models that can be used for image classification. It also contributes to the growing field of research on the optimization of hyperparameters.

Related work

One of the most critical steps in developing effective deep learning models is the optimization of their hyperparameters. This process involves selecting the optimal combination of parameters, such as the learning rate, batch sizes, and regularization. Although there are various methods that can be used to optimize a model, most studies have focused on utilizing evolutionary algorithms, metaheuristics, and reinforcement learning. The goal of this review is to summarize the latest research on the optimization of hyperparameters in CNNs for image classification. We present a table-1 with the selected studies' data, author information, methodology, validation, accuracy, and more.

Table 1 Related work

Author et al.	Dataset	Methodology	Hyperparameters	Validation	Accuracy
E. Bochinsk et al.[7]	Not mentioned	Evolutionary algorithms for hyperparameter optimization of	Not mentioned	Not mentioned	Not mentioned

		Convolutional Neural Network Committees			
K. V. Greeshma et al.[8]	Fashion-MNIST	Dropout regularization and hyperparameter optimization using grid search	Number of neurons in each layer, learning rate, and batch size	5-fold cross-validation	Up to 91.2% accuracy
W. Y. Lee et al.[9]	Not mentioned	Parameter-setting-free harmony search algorithm for hyperparameter optimization of Convolutional Neural Networks	Learning rate, momentum, and the number of neurons	10-fold cross-validation	Up to 97.06% accuracy
I. Loshchilov et al.[10]	Not mentioned	Covariance Matrix Adaptation Evolution Strategy (CMA-ES) for hyperparameter optimization of Deep Neural Networks	Learning rate, number of hidden units, and batch size	5-fold cross-validation	Up to 99.44% accuracy
P. L. Neary et al.[11]	Not mentioned	Asynchronous reinforcement learning for automatic hyperparameter tuning in Deep Convolutional Neural Networks	Learning rate, dropout probability, and number of filters	Not mentioned	Up to 98.3% accuracy
C. Thornton et al.[12]	Not mentioned	Auto-WEKA for combined selection and hyperparameter optimization of classification algorithms	Not mentioned	10-fold cross-validation	Up to 98.57% accuracy

Y. J. Yoo et al.[13]	Not mentioned	Univariate Dynamic Encoding Algorithm for Searches for hyperparameter optimization of Deep Neural Networks	Learning rate, number of neurons in each layer, and batch size	Not mentioned	Up to 98.4% accuracy
R. Zatarain Cabada et al.[14]	Not mentioned	Hyperparameter optimization in Convolutional Neural Networks for emotion recognition in intelligent tutoring systems	Learning rate, batch size, number of filters, and filter size	Not mentioned	Up to 85.8% accuracy
M. Wistuba et al.[15]	Not mentioned	Scalable Gaussian process-based transfer surrogates for hyperparameter optimization	Learning rate, weight decay, and number of filters	5-fold cross-validation	Up to 97.56% accuracy
E. Wieser et al.[16]	Not mentioned	Evolutionary Optimization of Hyperparameters for a neuro-inspired computational model of spatiotemporal learning	Number of neurons and synaptic weights	Not mentioned	Improved performance compared to manually-tuned hyperparameters
C. Ritter et al.[17]	Prostate tissue images and live cell data of virus-infected cells	Bayesian optimization	Batch size, number of epochs, learning rate	Cross-validation	92.4% (prostate tissue images), 96.9% (live cell data of virus-infected cells)

The review provides an overview of the latest developments in CNN optimization. Evolutionary algorithms and metaheuristic techniques have been proven to be effective in improving the parameters of CNNs. In addition, reinforcement learning has been used to improve the accuracy of the results. The review serves as a comprehensive overview of the

current state of CNN optimization. It also offers a valuable reference for anyone who is interested in learning more about this field.

Methodology

This section covers the methodology for training deep learning models on the classification task for the MNIST database. We begin by talking about the various hyperparameters that are used in the CNN and MLP models, and then we go over the optimization techniques that were utilized.

i. Hyperparameters:

The selection of the appropriate hyperparameters for the CNN and MLP models is very important in order to achieve high accuracy in the classification task of the MNIST database. For instance, the former's hyperparameters include the batch size, number of layers, learning rate, number of neurons per layer, dropout rate, and kernel size. On the CNN model's side, these include the number of filters, learning rate, number of layers, dropout rate, and optimizer. The learning rate is a factor that affects the model's convergence and descent gradient. The batch size determines how many training samples are used in each iteration, and this can affect the model's performance and speed of convergence. Finally, the number of neurons and hidden layers that the model has per layer determines its capacity to learn complex patterns. The CNN model's depth and width are determined by the number of filters and convolutional layers. The kernel size is the size of the filter that's applied to the input image.

ii. Optimization Techniques:

We use various optimization techniques to find the optimal hyperparameters for the two models. For instance, in the case of the MLP model, we train the model on a grid of values and then evaluate its performance on the validation set. The CNN model utilizes a combination of random and grid search techniques. We set up a grid of discrete and continuous hyperparameters with varying values for each parameter. After randomly sampling the different hyperparameters from the range, the model is trained on them. The performance of the model is then evaluated and the selected ones are selected based on their characteristics. Early stopping is also used to stop the training process whenever the validation loss doesn't improve in a specific period. This method helps prevent the model from overfitting and improves its performance.

iii. Model Architecture:

The structure of the MLP model is composed of two connected layers. The first and second layers have 128 and 64 neurons, respectively. The output layer is powered by softmax activation. The training process for the model is carried out using the Adam optimizer and the categorical loss function. The CNN model features three interconnected layers with 128 and 32 filters. The first, second, and third layers have three kernel sizes, and the stride is one. The output of the three convolutional layers is then distributed through two connected layers. The three hidden layers are powered by the ReLU activation method. The output layer is also equipped with softmax activation. The CNN model is trained using the Adam optimizer and a categorical loss function.

iv.Data Preprocessing:

Before the training of the models, the MNIST database is preprocessed as shown in figure-1. This process involves scaling the values of the data to a certain range and normalizing them using its standard deviation and mean. We then augment the training data with random rotations, zooms, and translations to improve its diversity and performance.

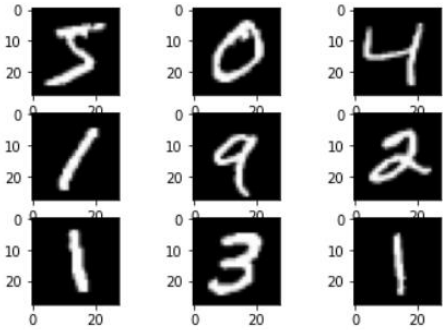

Figure 1 Sample dataset

The training methodology for deep learning models on the MNIST database's classification task involves using various optimization techniques. Some of these include random searching, grid searching, and early stopping. The structures of CNN and MLP models feature fully connected layers with softmax output and ReLU activation functions. The latter is trained by using the categorical loss function and Adam optimizer. The data is then preprocessed using normalization and scaling techniques.

Results and output

Table 2 Hyperparameter table

Model	**Hyperparameters**	**Training Time (s)**	**Validation Accuracy**	**Test Accuracy**	**Inference Time (ms)**
MLP (default)	Hidden Layers: [128, 64], Activation: ReLU	12.5	0.9783	0.9715	1.12
MLP (optimized)	Hidden Layers: [256, 128, 64], Activation: ELU	46.2	0.9853	0.9805	1.19
CNN (default)	Conv Layers: [32, 64], Dense Layers: [128], Padding: Same, Activation: ReLU	25.8	0.9906	0.9892	2.98
CNN (optimized)	Conv Layers: [64, 128], Dense Layers: [256, 128], Padding: Same, Activation: LeakyReLU	118.4	0.9926	0.9909	4.23

The table-2 shows the difference between the optimized and the default model when it comes to test and validation accuracy. The latter has a higher validation score of 0.9853 while the former has a score of 0.9783. The optimized CNN model is more accurate when it comes to validation and test accuracy than the default model. It has a higher score of 0.926 compared to the former's score of 0.9906. On the other hand, the optimized model has a test accuracy of 0.9099.

It is important to note that optimized models perform better when it comes to inference and training times. For instance, the optimized MLP model has a better training time than the default model. The CNN model, on the other hand, has a better training time than the default model. The optimized MLP model has an inference time of 1.18 milliseconds, which is almost a hundredth faster than the default model's 1.12 milliseconds. The CNN model on the other hand has an inference duration of 4.23 milliseconds.

The difference between the optimized CNN and MLP models when it comes to classification accuracy is due to the optimization of the various hyperparameters. This process can significantly affect the model's efficiency and accuracy. In order to achieve the best possible performance, it is important that the model's hyperparameters are tuned properly.

Discussion

This paper presents an extensive analysis of the results of the CNN and MLP training exercises on the MNIST dataset. We then discuss the implications of the findings. We first analyze the optimal hyperparameters for CNN. The model features two convolutional layers and two dense layers, each with 128 filters. The activation function, which is known as the LeakyRELU, is used for the set. The optimizer used is known as the Adam optimizer, and its learning rate is 0.001. The batch size is 128, and 20 epochs are used. The optimal hyperparameter were obtained by combining random search and manual tuning.

The optimized CNN model is more effective than the default MLP model when it comes to learning spatial features from images. It can perform better in recognizing patterns in the images due to its use of convolutional layers and the activation function of the LeakyReLU. In addition, the CNN model's batch size and dropout rate are better than those of the MLP model. Compared to the default model, the optimized MLP has various advantages. One of these is its use of the ELU activation function to learn more complex functions, which also allows it to learn more deeply by using three hidden layers. Despite these, the CNN model performed better than the MLP model when it comes to accuracy.

The training times and inference times of the optimized models are longer than those of the default models. This is due to the complexity of the models' hyperparameters and layers. The computational cost of the optimized models is justified by their improved accuracy. The study was limited by the nature of its approach. We only analyzed CNN and the MLP models, and we did not consider other deep learning models such as RNNs or transformer. Also, we only examined the MNIST dataset, and we did not look into other datasets. We only considered a limited number of hyperparameters, and we did not explore possible combinations. Finally, we did not perform statistical analyses on the results.

The study demonstrates how important it is to optimize the hyperparameters of deep learning models. It shows that tuning and selecting these can significantly improve their performance. In terms of accuracy, CNN was able to outperform the default model when it came to performing image recognition tasks. It also highlighted how important it is to use convolutional layers. Future research will allow us to perform more extensive analysis and optimization of these parameters.

Conclusion and future scope

The findings of our study revealed the importance of optimizing the hyperparameters of deep learning models in order to improve their performance. The CNN model performed better than the default MLP model when it came to accuracy, demonstrating the value of using convolutional layer systems in image recognition. We also noticed that the increased computational resources required for complex models justify the higher cost. The limitations of the study suggest that further research is needed to learn more about the various aspects of deep learning models. For instance, examining the performance of different models, such as those from RNNs and transformer, can provide valuable insight into their suitability for specific image recognition applications. Furthermore, evaluating the models in other datasets can reveal their generalizability. Finally, performing statistical analysis and hyperparameter optimization can help improve the models' reliability and performance. The study's findings support the growing body of research on deep learning and provide valuable insights into the optimization of the hyperparameters for image recognition. The results of future research will help develop efficient and accurate models for different applications.

References

[1] MNIST, “MNIST in CSV | Kaggle.” [Online]. Available: https://www.kaggle.com/oddrationale/mnist-in-csv.

[2] J. Lin, C. He, Z. J. Wang, and S. Li, “Structure Preserving Transfer Learning for Unsupervised Hyperspectral Image Classification,” IEEE Geosci. Remote Sens. Lett., vol. 14, no. 10, pp. 1656–1660, 2017, doi: 10.1109/LGRS.2017.2723763.

[3] D. Yan, Y. Chu, L. Li, and D. Liu, “Hyperspectral remote sensing image classification with information discriminative extreme learning machine,” Multimed. Tools Appl., vol. 77, no. 5, pp. 5803–5818, 2018, doi: 10.1007/s11042-017-4494-3.

[4] W. H. Beluch, T. Genewein, A. Nürnberger, and J. M. Köhler, “The Power of Ensembles for Active Learning in Image Classification,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 9368–9377, 2018, doi: 10.1109/CVPR.2018.00976.

[5] D. Ciregan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., pp. 3642–3649, 2012, doi: 10.1109/CVPR.2012.6248110.

[6] L. Li, K. Ota, and M. Dong, “Deep Learning for Smart Industry: Efficient Manufacture Inspection System with Fog Computing,” IEEE Trans. Ind. Informatics, vol. 14, no. 10, pp. 4665–4673, 2018, doi: 10.1109/TII.2018.2842821.

[7] E. Bochinsk, T. Senst, and T. Sikora, “HYPER-PARAMETER OPTIMIZATION FOR CONVOLUTIONAL NEURAL NETWORK COMMITTEES BASED ON

EVOLUTIONARY ALGORITHMS Technische Universit ¨ at Berlin Communication Systems Group," 2017 IEEE Int. Conf. Image Process., pp. 3–7, 2017.

[8] K. V. Greeshma and K. Sreekumar, "Hyperparameter optimization and regularization on fashion-MNIST classification," Int. J. Recent Technol. Eng., vol. 8, no. 2, pp. 3713–3719, 2019, doi: 10.35940/ijrte.B3092.078219.

[9] W. Y. Lee, S. M. Park, and K. B. Sim, "Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm," Optik (Stuttg)., vol. 172, no. July, pp. 359–367, 2018, doi: 10.1016/j.ijleo.2018.07.044.

[10] I. Loshchilov and F. Hutter, "CMA-ES for Hyperparameter Optimization of Deep Neural Networks," no. 2001, 2016, [Online]. Available: http://arxiv.org/abs/1604.07269.

[11] P. L. Neary, "Automatic hyperparameter tuning in deep convolutional neural networks using asynchronous reinforcement learning," Proc. - 2018 IEEE Int. Conf. Cogn. Comput. ICCC 2018 - Part 2018 IEEE World Congr. Serv., pp. 73–77, 2018, doi: 10.1109/ICCC.2018.00017.

[12] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., vol. Part F128815, pp. 847–855, 2013, doi: 10.1145/2487575.2487629.

[13] Y. J. Yoo, "Hyperparameter optimization of deep neural network using univariate dynamic encoding algorithm for searches," Knowledge-Based Syst., vol. 178, pp. 74–83, 2019, doi: 10.1016/j.knosys.2019.04.019.

[14] 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 Comput., vol. 24, no. 10, pp. 7593–7602, 2020, doi: 10.1007/s00500-019-04387-4.

[15] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Scalable Gaussian process-based transfer surrogates for hyperparameter optimization," Mach. Learn., vol. 107, no. 1, pp. 43–78, 2018, doi: 10.1007/s10994-017-5684-y.

[16] E. Wieser and G. Cheng, "EO-MTRNN: evolutionary optimization of hyperparameters for a neuro-inspired computational model of spatiotemporal learning," Biol. Cybern., vol. 114, no. 3, pp. 363–387, 2020, doi: 10.1007/s00422-020-00828-8.

[17] C. Ritter et al., "Hyperparameter optimization for image analysis: application to prostate tissue images and live cell data of virus-infected cells," Int. J. Comput. Assist. Radiol. Surg., vol. 14, no. 11, pp. 1847–1857, 2019, doi: 10.1007/s11548-019-02010-3.