Smart Agricultural Technology 11 (2025) 101016

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Impact of hyper-parameter tuning on CNN accuracy in agricultural image classification

João Mendes [a,b,*], José Lima [a,1], Lino Costa [b,1], Eligius M.T. Hendrix [c,1], Ana I. Pereira [a,1]

[a] Research Centre in Digitalization and Intelligent Robotics (CeDRI), Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, 5300-253, Bragança, Portugal
[b] ALGORITMI Research Centre/LASI, University of Minho, 4710-057, Braga, Portugal
[c] Universidad de Málaga, Málaga, Spain

ARTICLE INFO

Keywords:
Hyper-parameter optimization
Convolutional neural networks
Sensitivity analysis

ABSTRACT

This study explores the impact of hyper-parameter optimization on the performance of convolutional neural networks (CNNs) for olive cultivar classification using transfer learning. Pre-trained ImageNet models such as VGG16, InceptionV3, and ResNet50 were adapted to a proprietary dataset, with VGG16 selected for detailed evaluation. Key hyper-parameters, including layer count, neurons per layer, dropout rate, learning rate, and batch size, were tuned using random search. The best configuration achieved a validation accuracy of 87.5%, significantly outperforming the control model. Sensitivity analyses with Morris and Sobol methods identified the number of layers as the most influential factor, followed by dropout and learning rates through interaction effects. These findings demonstrate the importance of tailoring CNN architecture and regularization settings to the problem domain. These results underscore the importance of tuning architectural depth and regularization mechanisms for performance optimization. As a practical guideline, models with fewer layers and intermediate dropout levels demonstrated higher robustness and generalization, offering an effective strategy for adapting CNNs to agricultural classification tasks.

1. Introduction

Artificial intelligence (AI) algorithms have become remarkably prominent in recent years and are being applied to many fields. However, programming AI algorithms presents significant challenges, often requiring experienced technicians to achieve the desired result. Choices range from selecting the right algorithm or structure to configuring hundreds of hyper-parameters for effective training. This process can be complex and time-consuming.

In recent years, several techniques have been studied and adopted to systematize and simplify the training process of these algorithms. Among these techniques, *deep learning* approaches, particularly *transfer learning*, have been widely used due to their ability to save computational resources and time. These techniques are based on the simple yet powerful idea of reusing previously acquired knowledge.

Specifically, in the case of image classification using CNNs, the use of *transfer learning* techniques has become one of the most common approaches. Typically, knowledge is acquired from the ImageNet image classification competition and then adapted to a wide range of other classification problems. The success of this approach can be explained by the fact that the initial layers of convolutional networks tend to extract information about edges and shapes, while the deeper layers specialize in recognizing more abstract and problem-specific features [1].

Although these techniques have simplified model development, some fundamental variables remain, particularly those related to the network's hyper-parameters. These hyper-parameters are critical configurations that must be defined before training, encompassing aspects such as the learning rate, the number of layers, the number of neurons per layer, and the batch size. Unlike parameters that the network automatically adjusts during learning, hyper-parameters control the learning

* Corresponding author at: Research Centre in Digitalization and Intelligent Robotics (CeDRI), Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, 5300-253, Bragança, Portugal.
E-mail addresses: joao.cmendes@ipb.pt (J. Mendes), jllima@ipb.pt (J. Lima), lac@dps.uminho.pt (L. Costa), eligius@uma.es (E.M.T. Hendrix), apereira@ipb.pt (A.I. Pereira).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.atech.2025.101016
Received 11 December 2024; Received in revised form 14 May 2025; Accepted 14 May 2025
Available online 22 May 2025

process and require careful selection to ensure optimal model performance.

Given this complexity, this study explores the importance of hyper-parameters in the *transfer learning* process applied to supervised classification of olive cultivars from digital photographs of their leaves. Olive cultivar classification is a field that has been developed during recent years, with significant contributions in the use of *machine learning* [2] and *deep learning* [3] algorithms. The primary goal is to develop a tool that is easily accessible and free of charge for users, with classification based on the tree leaves, without the need for manual feature extraction, as presented in other works. The process we present is entirely automatic, where the user simply takes photographs of an olive leaf, and through supervised algorithms, the image is classified into the categories for which the model has been trained. Among the main challenges are the visual similarity of the leaves and the limited availability of data.

This study focuses on the impact of hyper-parameters on the classification process. The aim is to optimize the algorithms in the future, making them more reliable and versatile across different scenarios. This case study illustrates the impact of the hyper-parameters in the architecture of a given CNN.

The primary objective of this study is to explore how hyper-parameter tuning affects the performance of CNNs in the context of olive cultivar classification. Specifically:

- Evaluate and compare the performance of a control model and an optimized CNN using transfer learning;
- Identify the most influential hyper-parameters using quantitative sensitivity analysis techniques;
- Propose practical recommendations for CNN architecture design based on model interpretability and generalization performance.

This paper is organized as follows. After a literature review, presented in Section 2, Section 3 outlines a systematic methodology. Considering the number of variables that can influence the training and validation of the models, several tests were carried out to ensure the reliability of results as described in Section 4. Section 5 discusses obtained results and the conclusions are presented in the Section 6, where the main findings and suggestions for future work are addressed.

2. Related work

The tuning of hyper-parameters is, without doubt, one of the most crucial steps in the implementation of an AI algorithm. In particular, for CNNs, several authors emphasize the importance of proper hyper-parameter tuning to extract optimal performance from the final model. Several researchers have proven and reported that hyper-parameters play a significant role in the model convergence time, performance, robustness, and generalization capability [4]. However, despite their importance, defining optimal values for these hyper-parameters remains a complex and challenging task, often considered one of the most difficult aspects of applying learning algorithms.

There are two main approaches to hyper-parameter search: choosing them either manually or automatically. A manual search is typically associated with experienced practitioners, requiring a deep understanding of how hyper-parameters influence the model behavior. The other option is automatic search methods, which are more commonly used against a high computational cost, as they involve systematically exploring different hyper-parameter combinations and evaluating the resulting models based on performance metrics.

Literature on automatic search ranges from traditional methods like grid search [5], which systematically evaluates a predefined grid of hyper-parameters, to random search [6], where hyper-parameter combinations are randomly selected within a defined search space. Bayesian optimization [7] employs probabilistic methods to model the loss function and select the most promising hyper-parameters for the next evaluation. Evolutionary algorithms [8] apply natural selection principles to evolve a population of hyper-parameter populations over several generations. Gradient-based methods [9] attempt to calculate or estimate the gradient of the loss function concerning the hyper-parameters, guiding the search in the right direction. Other emerging techniques include the use of reinforcement learning [10,11] or Q-learning [12].

While these search methodologies are well-documented in the literature, the direct influence of hyper-parameters on model evaluation metrics is less thoroughly explored. Several studies discuss hyper-parameter optimization using several techniques and datasets, but few explore the relationship between different hyper-parameters and key model performance indicators. Given this gap, this study focuses on researching the relation between hyper-parameters and model performance. An extensive search was conducted across major databases (Web of Science, Scopus, and IEEE), where only two papers specifically addressing this topic were found.

Taylor et al. [13] present an innovative approach to rank hyper-parameter influence in deep learning using sensitivity analysis (SA). The study applies two formal SA methods, Morris and Sobol, to the hyper-parameters of four deep learning models: DNN, ResNet18, AlexNet, and GoogleNet, tested across three image classification datasets: MNIST, MNIST-Fashion, and CIFAR-10. The hyper-parameters evaluated consist of the optimizer, learning rate, momentum, learning rate decay, learning rate decay step, batch size, and number of epochs. Results indicate that learning rate decay is the most influential hyper-parameter, affecting model performance regardless of architecture or dataset. In contrast, the initial learning rate was found to have low importance, contrary to existing literature. Additionally, hyper-parameter influence was closely tied to model architecture: shallower models were more sensitive to hyper-parameters affecting stochasticity in learning, while deeper models were influenced by those impacting convergence speed.

Similarly, the study by Wojciuk et al. [14] explores the impact of hyper-parameter optimization on the performance of CNN models. The authors explore the effectiveness of several hyper-parameter optimization methods, including Grid Search, Random Search, Bayesian Optimization, and the Asynchronous Successive Halving Algorithm (ASHA). Experiments were conducted on three datasets: CIFAR-100, Stanford Dogs, and MIO-TCD, evaluating the importance of different hyper-parameters such as learning rate, dropout rate, and the number of fine-tuned layers. The key findings include the identification of learning rate and dropout rate as the most influential hyper-parameters for optimizing CNN performance. The ASHA and Bayesian Optimization methods generally outperformed Grid and Random Search strategy in terms of achieving higher classification accuracy and requiring fewer iterations. The study also highlighted that using a balanced subset of training data, or applying techniques like class weighting and data augmentation, can effectively optimize hyper-parameters and improve model performance without a need to use the entire dataset. The authors emphasize the critical role of customized hyper-parameter optimization in enhancing the classification accuracy of CNN models.

Both studies underscore the importance of hyper-parameter selection and optimization in the performance of deep learning models, although they employ different approaches and methods. Taylor et al. [13] focus on sensitivity analysis to classify hyper-parameters, while Wojciuk et al. [14] compare several hyper-parameters optimization techniques and highlight the importance of advanced methods and data balancing in the optimization process. None of these works specifically uses the number of neurons in each fine-tuned layer as a hyper-parameter for comparison, nor did they apply a dataset tailored to a specific problem domain. This study aims to fill these gaps by applying hyper-parameter optimization to a dataset specific to a single problem, including the number of neurons in the fine-tuned layers as one of several hyper-parameters under investigation. Additionally, this research is applied to a specific case study, providing a detailed and practical analysis of how hyper-parameter optimization can be adapted to real-world scenarios with proprietary data, thus expanding the understanding of hyper-parameter impact in specific and complex contexts.

Recent studies have demonstrated the growing applicability of deep learning models in agriculture, particularly for plant disease detection. For instance, one study applied an Inception-v3 CNN architecture to classify apple leaf diseases, incorporating advanced segmentation to improve the identification of disease regions in uncontrolled field conditions. The model achieved an accuracy of 94.76%, outperforming previous approaches and confirming the potential of CNN-based models for real-world scenarios [15]. Another study proposed a custom 5-layer CNN for detecting potato leaf diseases, which surpassed both MobileNet and a simpler 4-layer CNN in all evaluated metrics. The model reached 97.16% accuracy and demonstrated strong performance in F1-score, recall, and precision, as well as in object detection tasks using Faster R-CNN [16]. While these works highlight the effectiveness of CNN architectures in agricultural tasks, they focus primarily on classification performance and lack a systematic analysis of how individual hyper-parameters affect model outcomes. In contrast, the present study complements this research by integrating sensitivity analysis techniques, namely SHAP, Sobol, and Morris, to identify the most impactful hyper-parameters in CNNs applied to cultivar classification. This approach aims to enhance the transparency, reproducibility, and performance optimization of deep learning models in agriculture.

3. Materials and methods

This section aims to provide a clear and concise overview of all the methods and materials employed to address the research question. First, we provide a detailed description of the used dataset, followed by an explanation of the methodology adopted for conducting the experiments. Finally, an introduction to the methods used in the analysis and processing of the data will be provided, ensuring a thorough understanding of the implemented approaches.

3.1. Dataset

In this study, the original dataset, previously developed and validated in [3] for cultivar identification, was adapted for further analysis. To explore the influence of hyper-parameters on model performance in olive cultivar classification in a systematic way, the dataset was divided into five smaller sub-datasets. Each sub-dataset maintains the same proportion of images per category as the original dataset, but its size is limited to 400 images each. This division facilitates a more efficient evaluation process by reducing the computational load and time required for training and validating models. This enables more extensive exploration of hyper-parameter configurations.

The dataset was originally designed to capture the variability in olive leaf morphology and was collected from olive groves where previous studies had already been conducted. The collection has been done in collaboration with certified technical personnel, ensuring that only samples from the desired cultivars were collected. The collection process was meticulously planned to span all seasons of the year, capturing the growth and recession phases of the leaves, as well as other factors influencing their development, such as water or nutrient stress. This approach guarantees correct identification regardless of the stage of growth and condition of the tree.

To ensure randomness and representativeness in the sampling process, leaves were collected from both the interior and exterior of the canopy, including both new and older leaves. On average, 20-25 leaves per tree were collected from four selected cultivars: *Cobrançosa*, *Madural*, *Negrinha de Freixo*, and *Verdeal Transmontana*. These cultivars were chosen because they are indigenous to Trás-os-Montes (Portugal) and are included in the protected designation of origin (Azeite de Trás-os-Montes DOP), making their identification essential for enhancing the value of the olive oil produced in these regions.

The area selected for the collection process was the parish of Sucães (41.49°N, 7.26°W), with an average altitude of 350 meters, located in the district of Bragança and the municipality of Mirandela. This choice was made based on the presence of the desired cultivars, accessibility, and the area's prominence as one of the main olive oil producers in the region. Based on these criteria, four collections were conducted:

1. Carried out on 2021-10-11 (Initial harvest stage): 149 leaves of *Cobrançosa*, 89 of *Madural*, 101 of *Negrinha de Freixo*, and 106 of *Verdeal Transmontana*;
2. Carried out on 2021-12-15 (Final harvest stage): 169 leaves of *Cobrançosa*, 231 of *Madural*, 219 of *Negrinha de Freixo*, and 214 of *Verdeal Transmontana*;
3. Carried out on 2022-06-30 (Fruit growth stage): 380 leaves of *Cobrançosa*, 237 of *Madural*, 384 of *Negrinha de Freixo*, and 402 of *Verdeal Transmontana*;
4. Carried out on 2022-10-3 (Initial harvest stage): 348 leaves of *Cobrançosa*, 392 of *Madural*, 362 of *Negrinha de Freixo*, and 473 of *Verdeal Transmontana*.

After collection, the leaves were transported to the laboratory for immediate photographing to preserve their physical characteristics. The leaves from the first two collections were photographed with a digital camera (Xiaomi Redmi Note 9 Pro) at a resolution of 2610×4640 pixels, while the remaining collections were photographed with a similar camera (Xiaomi 11T) at a resolution of 3000×4000 pixels. All samples were photographed on a white background in RGB format, as illustrated in Fig. 1, to facilitate their subsequent processing and application in the algorithms. The entire dataset is publicly available in [17].

3.1.1. Dataset pre-processing

Pre-processing techniques were applied to reduce size and resolution of images to optimize the training and validation process. This was done using an automated process implemented in Python (version 3.8.12) with the OpenCV library (version 4.0.1). The pre-processing steps applied to the dataset are detailed below and illustrated in Fig. 2:

1. **Loading and Ratio Calculation**: After loading the image, its original ratio and size are calculated. This step ensures that the leaf proportions are maintained during cropping.
2. **Grayscale Conversion**: The image is converted to grayscale to simplify subsequent calculations and reduce computational complexity.
3. **Edge Detection**: Edge detection is performed using the John F. Canny algorithm, implemented through the OpenCV library. This process applies five parameters: the input image, low and high threshold values (set to 50 and 90), the kernel size for the Sobel filter (3×3), and the L2-gradient equation $Edge\_Gradient(G) = |G_x| + |G_y|$.
4. **Noise Reduction**: A blur function is applied to reduce noise, using a 5×5 kernel to convolve the image with a low-pass filter.
5. **Adaptive Thresholding**: Adaptive thresholding is applied using a Gaussian-weighted sum of neighborhood values to determine the threshold. The area value is then calculated by counting all non-zero pixels in the resulting matrix.
6. **Contour Definition**: Different contour sizes are defined based on the calculated area, ratio, and size. This promotes consistent handling of images of varying dimension and ensuring the entire leaf is captured.
7. **Contour Outlining**: Contours are outlined, with only objects larger than a specified contour size being selected to ensure accuracy.
8. **Leaf Detection and Cropping**: Bounding box coordinates are determined to enable cropping while maintaining the original image ratio, with a margin of 0.01% of the initial area preserved along the yy axis.

After pre-processing, the dataset was divided into five sub-datasets, each containing approximately 400 images, balanced as the original dataset. These sub-datasets were then used for hyper-parameter testing,

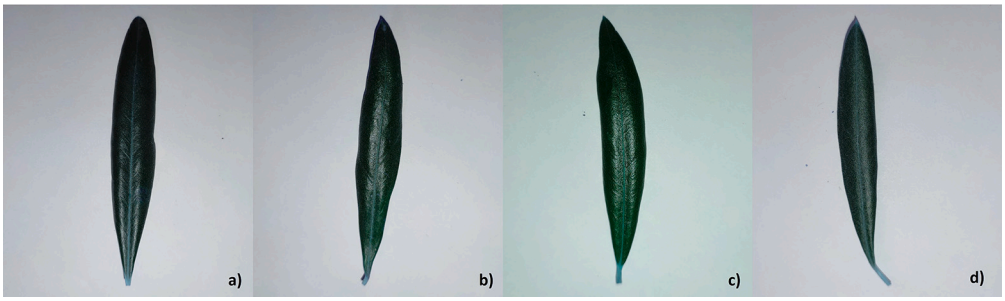

Fig. 1. Dataset cultivars: a) *Cobrançosa*; b) *Madural*; c) *Negrinha*; d) Verdeal [3].

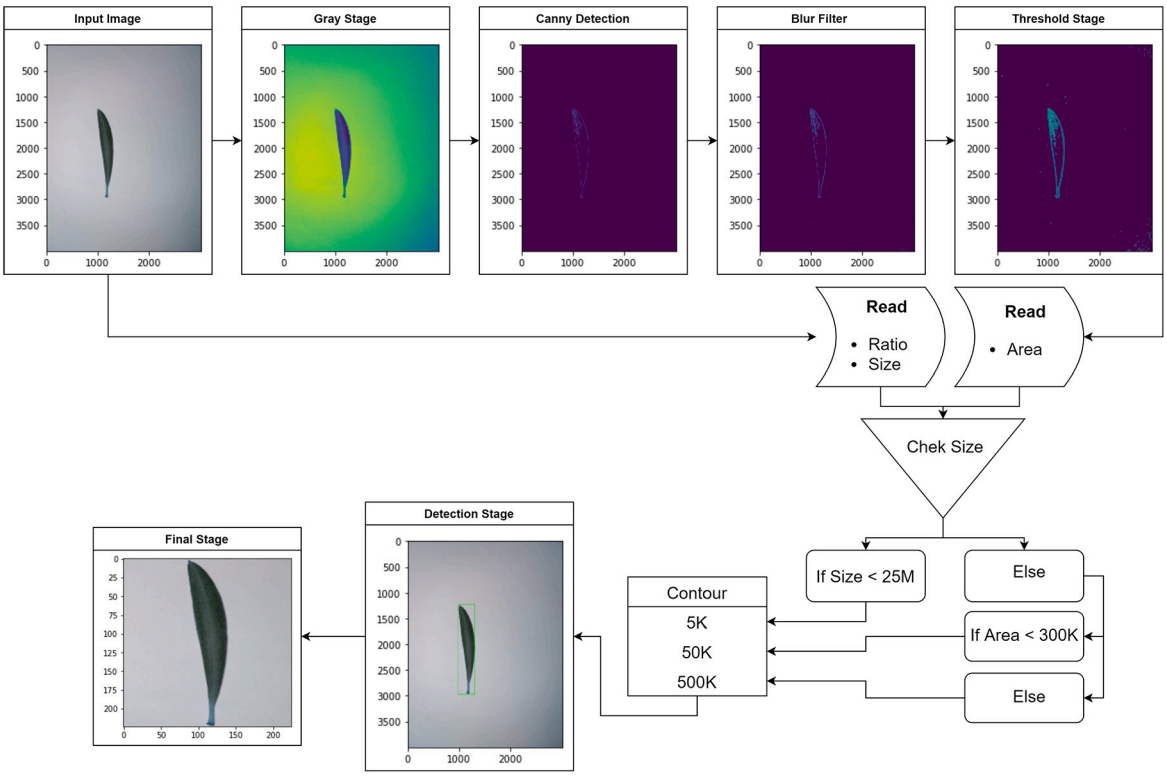

Fig. 2. Outline of pre-processing steps [3].

with 70% of each sub-dataset allocated for training and the remaining 30% for model validation.

3.2. Methods and methodology

Assessing the impact of a hyper-parameter on CNN performance requires a systematic methodology to ensure reliable and replicable results. This section details the entire process, covering the methods and techniques applied throughout the study.

Our research relies on transfer learning techniques to explore the influence of hyper-parameters on the performance of CNNs. Transfer learning is a technique widely employed in deep learning and enhances model training by leveraging knowledge from pre-trained networks on large-scale datasets such as ImageNet, which in 2014 contained over 14 million labeled images across 1000 classes [18]. This method allows models to use pre-learned features such as edges, textures, and shapes that are broadly applicable across domains, accelerating learning on new, task-specific datasets. By freezing the initial layers of a model to retain these general features and fine-tuning the final layers, transfer learning reduces training time significantly in terms of computational costs and need for extensive labeled data.

Three CNN architectures, VGG16, InceptionV3, and ResNet50, were selected to evaluate the impact of hyper-parameters on model accuracy. Each architecture offers a distinct approach to feature extraction, providing insights into performance outcomes. The following sections present these architectures in detail, highlighting their design features and methods to contextualize their use in this study.

3.2.1. VGG16

Based on preliminary testing and a review of available architectures, VGG16 was selected as one of the options for this study due to its simplicity and proven effectiveness. Introduced by Simonyan and Zisserman in 2014 for the ImageNet Challenge [19], VGG16 consists of 13 convolutional layers and three fully connected layers, culminating in a softmax activation function for classification. Its streamlined design of stacked 3x3 convolutional layers with pooling layers achieves depth with minimal complexity, facilitates efficient transfer learning. The regular architecture structure and linearity contribute to its robust performance

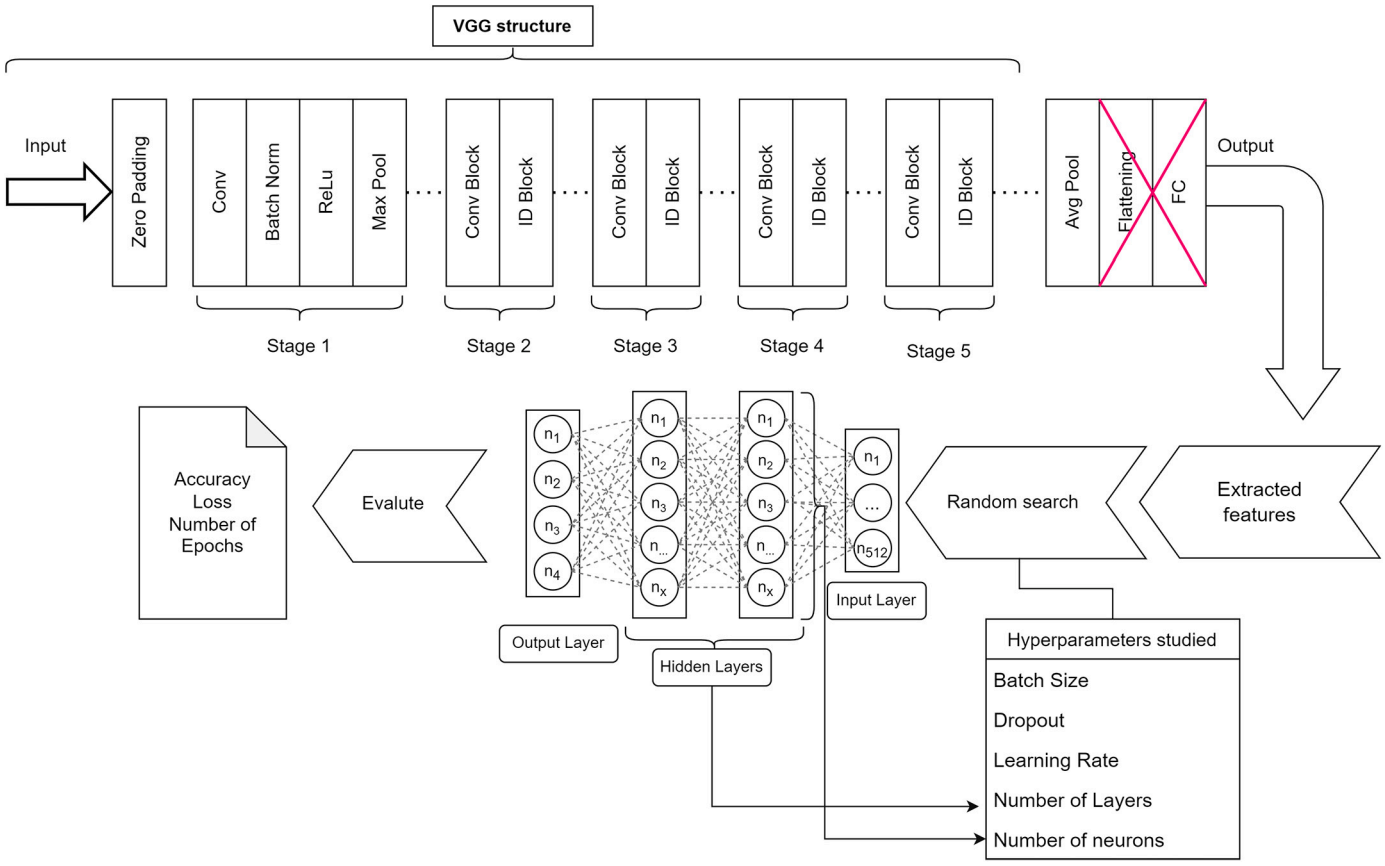

Fig. 3. Summary of the used modified CNN architecture.

in image classification, making it a widely studied and preferred choice for applications requiring high accuracy and interpretability.

3.2.2. *Inceptionv3*

InceptionV3, developed by Google in 2015 [20], was selected for this study due to its innovative structure and computational efficiency. The architecture incorporates inception modules that apply multiple convolutional filters of varying size in parallel, allowing the network to capture information at different scales within each layer. This design reduces computational cost by using factorized convolutions. This makes InceptionV3 an efficient choice for transfer learning. Its modular structure enhances adaptability and generalization, allowing for effective feature extraction across diverse datasets and minimizing overfitting risks.

3.2.3. *ResNet50*

ResNet50, introduced by He et al. in 2015 [21], was selected for this study due to its depth and its innovative use of residual connections, which address the degradation problem common in deep networks. Comprising 50 layers, ResNet50 incorporates residual blocks that enable gradients to flow more smoothly during backpropagation, facilitating effective training of deeper networks. This architecture enhances learning efficiency and accuracy, making it well-suited for transfer learning applications that require complex feature extraction. The ResNet50 residual connections promote high performance with increasing depth, thus providing valuable insights for Hyper-parameter tuning.

3.3. *Methodology*

Several modifications were made to the base architecture to adapt the CNN models to the specific requirements of this study. Based on findings in [3], which used the same dataset, the final two fully connected layers of the CNN were replaced by an average pooling layer. This adjustment reduces the number of parameters to train. This enables expanding the hyper-parameter search space that better targets the study objective. The modified architecture is shown in Fig. 3.

As depicted in Fig. 3, the network structure is not fully defined, as one of the hyper-parameters to be studied is the number of fully connected layers. Depending on the configuration, the structure can vary in a range of one single output layer (with no additional fully connected layers) to five fully connected layers. Moreover, the number of neurons within each fully connected layer also varies from 2 to 2048, following a geometric progression with a ratio of two (e.g., 2, 4, 8, etc.). Additionally, the presence or absence of dropout layers, with values ranging from 0.0 to 0.99, was tested. Other tested hyper-parameters include the learning rate and the batch size. Table 1 provides a summary of the hyper-parameters and their respective ranges. The ranges defined for each hyper-parameter in Table 1 were selected based on empirical evidence from previous studies [13,14] and preliminary testing on the dataset used. The number of layers ranged from 0 to 4 to evaluate shallow and moderately deep architectures while avoiding vanishing gradients in deeper networks. The neurons per layer followed a geometric progression from 2 to 2048 (i.e., 2^1 to 2^{11}) to cover a wide range of capacities without excessive memory usage. The dropout range from 0.0 to 0.99 ensures inclusion of both no regularization and strong regularization regimes. Learning rates were chosen between 10^{-7} and 10^{-2} to span from very conservative updates to more aggressive ones. Finally, batch sizes between 32 and 128 were tested, as values below 32 showed no performance gain and higher values slowed convergence in initial trials.

After defining the hyper-parameter search space, the model was compiled using the Adam optimizer and trained for approximately 50 epochs. An early stopping criterion was also implemented, automatically halting the training process if the validation accuracy does not improve for eight consecutive epochs.

Table 1
Hyper-parameter ranges.

Hyper-parameter	Range
Number of layers	{0, ..., 4}
Neurons per layer	$\{0, 2^1, ..., 2^{11}\}$
Dropout	[0.0, 0.99]
Learning rate	$[10^{-7}, 10^{-2}]$
Batch size	$\{2^1, ..., 2^7\}$

To optimize the hyper-parameters and explore their influence on model performance, it was initially decided to use a computationally simple method that promotes a wide exploration of the search space, followed by the use of an evolutionary algorithm when the search space gets more restricted. The chosen method for the initial search was random search, which, despite its stochastic nature, provides a robust means of exploring a wide range of hyper-parameter combinations, increasing the likelihood of identifying a near-optimal configuration within the defined evaluation limits. This approach aims to identify the hyper-parameter configuration that maximizes model validation accuracy. Throughout each evaluation, the eight pre-defined hyper-parameters were optimized without any imposed constraints. All implementations were carried out using Python (Version 3.9.19) with TensorFlow (Version 2.9.3) and JMetalPy (Version .6.0) libraries. The random search was implemented using the RandomSearch class from the JMetalPy library, which generates candidate solutions by randomly sampling hyper-parameter combinations within predefined bounds. Each configuration was evaluated through a custom Problem class, where the objective function trained the CNN with the given parameters and returned the validation accuracy. However, the distribution of results obtained from Random Search did not reveal any dominant regions or convergence patterns, the performance was uniformly spread across all hyper-parameter values. This lack of concentration in high-performing zones prevented the effective use of evolutionary algorithms, as the search space remained too large and undirected for meaningful exploitation.

After defining the convolutional structure and optimization algorithm, a next step is to establish a robust framework for evaluating the hyper-parameter. For this purpose, the most commonly used metrics are model accuracy, loss, and the number of epochs required for the model to converge. For the latter, given the implementation of the early stopping criterion described earlier, the number of epochs, until training is halted, was considered the final value, especially in cases where the models did not fully converge. If the early stopping criterion is not met, the model continues training for a maximum of 50 epochs.

In terms of computational cost, approximately 500 evaluations are used for the random search algorithm for each of the five training subsets. To ensure the replicability of the results, a five-fold cross-validation was applied, resulting in approximately 12,500 evaluations. Considering that each model takes an average of 15 epochs to complete training, the total number of computed epochs is approximately 190,000.

From this set of results, the values of the five tested hyper-parameter, along with the accuracy, loss, and the number of epochs achieved by each model, were recorded. Several analyses were conducted to extract the maximum amount of information from these results. Although the primary focus of this work is on the influence of hyper-parameter on the performance of a transfer learning model, the optimization of the model itself is another subject of study.

In the process of analyzing the results obtained during the hyper-parameter optimization of the CNN, both linear and nonlinear interactions between hyper-parameter and model performance were explored. Initially, histograms were generated to examine the distribution of the primary hyper-parameter. Additionally, the distribution of validation accuracy and model loss was assessed, providing an initial overview of the data characteristics.

Following the optimization, a detailed analysis was conducted to understand the relationships between hyper-parameter and CNN performance. A correlation matrix was calculated to study the linear interactions between hyper-parameters. Such an analysis helps to identify direct correlations between hyper-parameter and highlighted potential interactions that can affect network performance.

However, a correlation analysis does not identify necessarily relations that are more complex and nonlinear. Therefore, we applied additional techniques to capture these interactions, such as machine learning methods, specifically Random Forest (RF) [22]. RF is a decision tree-based machine learning technique that constructs multiple decision trees during training and combines their predictions to improve accuracy and reduce overfitting.

For a regression task, the prediction of a sample x is given by:

$$\hat{y}(x) = \frac{1}{T}\sum_{t=1}^{T} h_t(x) \tag{1}$$

where h_t is the prediction of the t-th decision tree, and T is the total number of trees. Feature importance is derived from the average decrease in impurity (e.g., mean squared error) across all trees, allowing us to estimate how much each hyper-parameter contributes to the validation accuracy [23].

This method is particularly effective in capturing nonlinear interactions between variables, making it ideal for this study. The choice of RF was primarily motivated by its robustness and its ability to provide an estimate of variable importance, indicating how each hyper-parameter contributes to the prediction validation accuracy.

To complement the hyper-parameter importance analysis, the SHAP (Shapley Additive Explanations) tool was used. SHAP is a game theory-based technique that assigns a value to each feature (in this case, hyper-parameter) representing its contribution to the model prediction [24].

Formally, the SHAP value ϕ_i for a feature i is defined as:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f(S \cup \{i\}) - f(S)] \tag{2}$$

where F is the set of all features, S is any subset not containing i, and $f(S)$ is the prediction based only on the features in S. SHAP values provide interpretable and consistent explanations of model behavior, especially in settings with complex, nonlinear interactions between variables [25].

SHAP is widely recognized for providing interpretable and consistent explanations of machine learning model behavior, especially in cases where variable relationships are complex and nonlinear.

Applying SHAP, we were able to break down the contribution of each hyper-parameter to the RF model prediction. This provided a detailed insight into how each parameter affects validation accuracy, both in absolute terms and relative to other variables. SHAP not only confirms the findings obtained with RF but also revealed subtle and complex interactions between hyper-parameter that might not have been found with less sophisticated analyses.

To further understand the sensitivity of the model to its hyper-parameters, we applied the Morris and Sobol methods. The Morris method [26] is a screening technique based on the computation of elementary effects:

$$EE_i = \frac{f(x_1, ..., x_i + \Delta, ..., x_k) - f(x)}{\Delta} \tag{3}$$

where Δ is a small step in the input space for parameter x_i. The method produces metrics such as the mean of absolute values μ^*, standard deviation σ, and mean μ of the effects, which help identify influential parameters and potential nonlinear or interaction effects. The Sobol method [27], in contrast, is a global sensitivity analysis technique that decomposes the total output variance V into contributions from individual parameters and their interactions. The first-order and total-order Sobol indices are defined as:

$$S_i = \frac{V_i}{V}, \quad S_{Ti} = \frac{V_i + V_{int}}{V} \tag{4}$$

where V_i is the variance attributed to x_i alone, and V_{int} represents the combined variance from all interactions involving x_i. These indices allow a more detailed and quantitative understanding of the influence of each hyper-parameter on model performance. The Morris method was used as a preliminary screening to highlight the most influential hyper-parameters. Following this, the Sobol method was applied to quantify and decompose the sensitivity into main and interaction effects. Together, these analyses provided a deeper insight into the factors that drive CNN performance under a transfer learning approach.

4. Results

This section presents results in two parts. The first part presents results of hyper-parameter optimization and its impact on the final model. Given that initial evaluations showed similar performance trends across the three architectures, only results for the VGG16 model are exposed, as this simpler architecture allows a straightforward interpretation and discussion. In a second part, we discuss a sensitivity analysis of the hyper-parameters.

4.1. Hyper-parameter optimization results

The control model, used as a baseline for comparison, was based on the same VGG16 architecture with transfer learning applied. In this model, the convolutional base from ImageNet was kept frozen, and only the top classification layers were trained. Specifically, two fully connected (dense) layers were added after the global average pooling layer: the first with 512 neurons and ReLU activation, and the second with 256 neurons also using ReLU. A dropout layer with a rate of 0.4 was inserted between them. The final output layer used a softmax activation function for multi-class classification. The batch size was fixed at 32, and the optimizer used was Adam, with an initial learning rate of 10^{-5}, reduced on plateau with a factor of 0.6 and a minimum threshold of 10^{-7}. The model was trained using early stopping with a patience of 8 epochs. This configuration was kept constant across all five data subsets to provide a reliable benchmark.

Initial results were collected from a control model to evaluate optimization's effects on the final model. This model has a similar structure to the one studied but differs with respect to a fixed number of two fully connected layers. The first layer has 512 neurons and the second 256. Moreover, it includes a dropout rate among the links between these two of 0.4. The batch size was set to 32, and the learning rate was initially set at 10^{-5}, with this value being adjusted based on monitoring the validation accuracy, updated after two epochs of tolerance with an update factor of 0.6, down to a minimum of 10^{-7}. The same datasets presented earlier were used for the evaluation of the control model, with a five-fold cross-validation.

Table 2 shows that the values of the LOSS_Val and ACC_Val (loss value and accuracy on validation set) for the control models remain quite consistent, reflecting model reliability. The highest standard deviation among the five cross-validation assessments for loss was approximately 0.067 on an average loss of 0.911, as reflected in LOSS_Val_std. Similarly, the highest standard deviation in accuracy, approximately 0.052, was observed on an average accuracy of 0.625, as detailed in ACC_Val_std. These values confirm the representativeness of the created data subsets, with standard deviations across the average loss and accuracy of all models being 0.036 and 0.019, respectively.

The random search method was used with 500 evaluations for the random search algorithm for each of the five training subsets and a five-fold cross-validation. This results in approximately 2,500 evaluations for each data subset. The best results for each subset were then selected and presented in Table 3.

Table 3 shows that the validation loss of the five subsets ranges from 0.455 to 0.812. This indicates a significant variation in the models' ability to minimize the loss function. Subset 1 has the lowest validation loss (0.455), suggesting that the model trained with this subset was the most effective in terms of generalization. In contrast, subset 4 has the highest loss (0.812), which may indicate greater difficulty for this model in learning the data characteristics without overfitting.

Table 2
Summary of results from the control group.

Evaluation		Control				
		Batch1	Batch2	Batch3	Batch4	Batch5
1	LOSS_Val	0.900	0.996	1.007	0.979	0.985
	ACC_Val	0.633	0.575	0.592	0.567	0.608
2	LOSS_Val	0.874	0.880	1.060	0.863	0.811
	ACC_Val	0.642	0.650	0.542	0.633	0.683
3	LOSS_Val	0.884	0.927	0.949	0.919	0.866
	ACC_Val	0.650	0.658	0.625	0.667	0.675
4	LOSS_Val	1.041	0.944	1.042	0.823	0.924
	ACC_Val	0.525	0.675	0.567	0.700	0.650
5	LOSS_Val	0.855	0.907	0.930	0.926	0.929
	ACC_Val	0.675	0.617	0.658	0.658	0.642
LOSS_Val_mean		0.911	0.931	0.998	0.902	0.903
LOSS_Val_std		0.067	0.039	0.051	0.054	0.060
ACC_Val_mean		0.625	0.635	0.597	0.645	0.652
ACC_Val_std		0.052	0.035	0.041	0.045	0.027

Table 3
Results of random search process for hyper-parameter tuning.

Subsets	Validation Loss	Validation Accuracy
1	0.455	0.875
2	0.667	0.833
3	0.607	0.833
4	0.812	0.808
5	0.567	0.817

Values for validation accuracy range from 0.808 to 0.875. We observe the same tendency; subset 1 achieves the best performance, with an accuracy of 0.875, while subset 4 reveals the lowest performance, with an accuracy of 0.808. This pattern reinforces the consistency observed in loss minimization, where the model with the lowest loss also achieves the highest accuracy.

Comparing these results with those of the control model discussed earlier, a significant improvement is observed both in loss minimization and accuracy maximization. The results of the control model show a loss range from 0.8226 to 1.0416, and in accuracy from 0.5249 to 0.6997. With random search, the loss is substantially reduced, and accuracy increases significantly. This comparison demonstrates the positive impact of applying optimization.

In comparison to the control model, the standard deviation of loss and accuracy among the optimized subsets is higher. A high standard deviation indicates greater variability in the results, which can be attributed to the diversity of solutions found through the random search method. The greater variability can be interpreted as evidence that random search explored a broader range of models, capturing different aspects of the data. However, it may also indicate that the optimized models are more sensitive to variations in the dataset, especially concerning more complex patterns or outliers present in the training set. Therefore, while random search improved the average performance metrics, it also introduced greater uncertainty in the results, which should be considered when evaluating the robustness of the models.

4.2. Sensitivity analysis of hyper-parameters

This section focuses on results of the sensitivity analysis to explore the influence of hyper-parameters on the final accuracy. Several techniques described in Section 3.3 were used. We first studied the data distribution. As data is obtained through the Random Search optimization algorithm, data might not be uniformly distributed, leading to potential gaps in intervals. Such an analysis aims at identifying possible areas of undersampling in the hyper-parameter space, ensuring that no critical

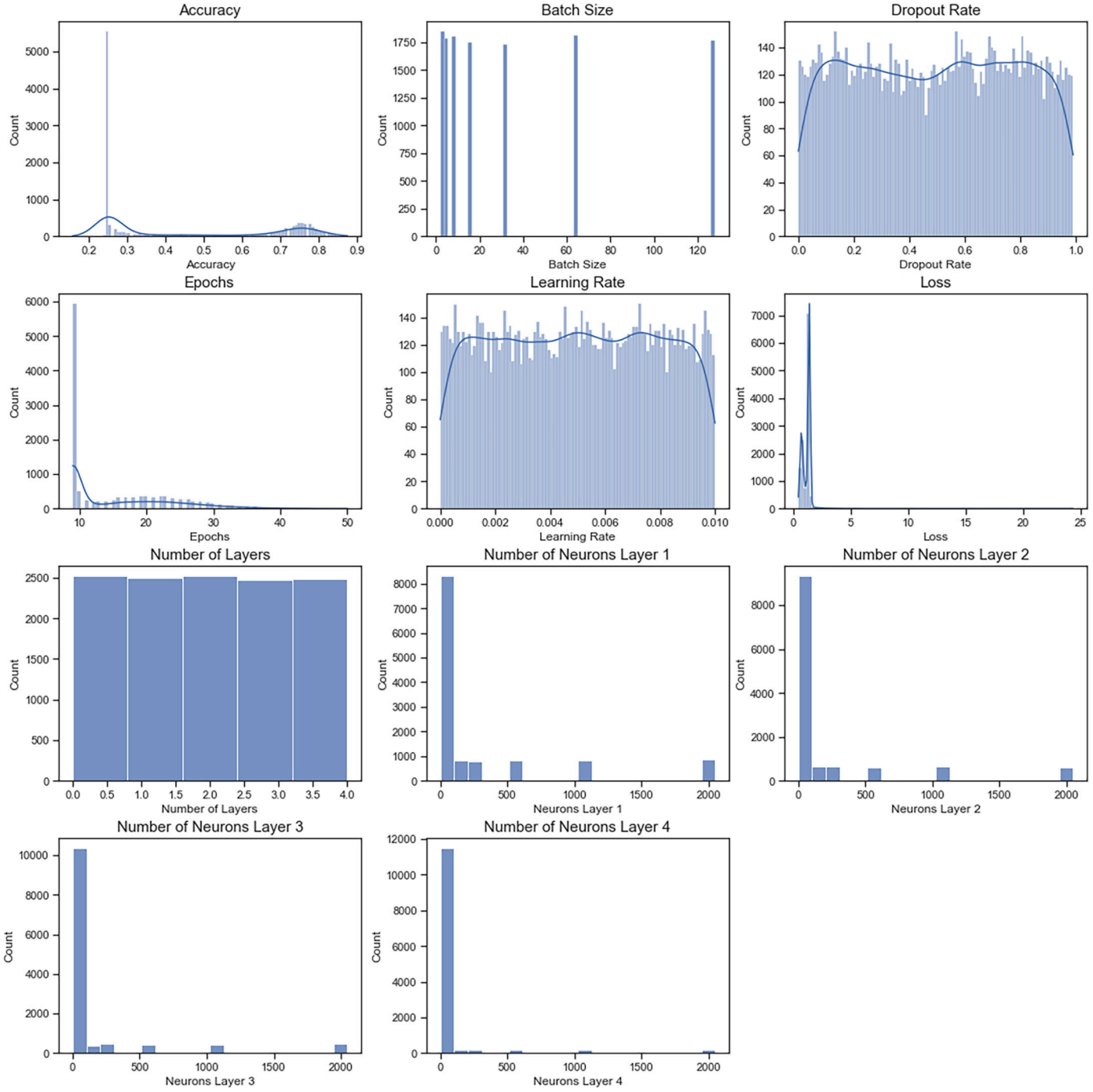

Fig. 4. Distribution of the hyper-parameters.

interval is overlooked, which can compromise the sensitivity evaluation and, consequently, the interpretation of the relative importance of each hyper-parameter on the model performance.

Fig. 4 presents the distributions of the main hyper-parameter used in the sensitivity analysis for the tested CNN model. Additionally, it shows performance metrics such as accuracy and loss. Observing each case, it is possible to affirm that validation accuracy exhibits a multimodal distribution, with a global maximum around 0.25 and some local maxima between 0.75 and 0.80. The high density of models with accuracies around 0.2-0.3 suggests that many hyper-parameter combinations do not result in optimal performance, although some specific configurations achieve high accuracy, indicating that the optimization may have found a good configuration.

The parameter of the batch size shows a discrete distribution, reflecting a geometric progression with a factor of 2, as expected. The value distribution is well spread among the studied possibilities, revealing that the Random Search method consistently explored all intended intervals.

The dropout rate, in turn, exhibits a fairly uniform distribution, which is positive for sensitivity analysis. The broad coverage of values ensures that the influence of this hyper-parameter was comprehensively analyzed without significant gaps.

The distribution of epochs clusters around lower values, suggesting that many models converged quickly. However, it does not clarify whether they converged to acceptable accuracy values or remained in the group of models that only reached 25% of accuracy. Notice that this value is the minimum. Similarly, the learning rate also shows a relatively

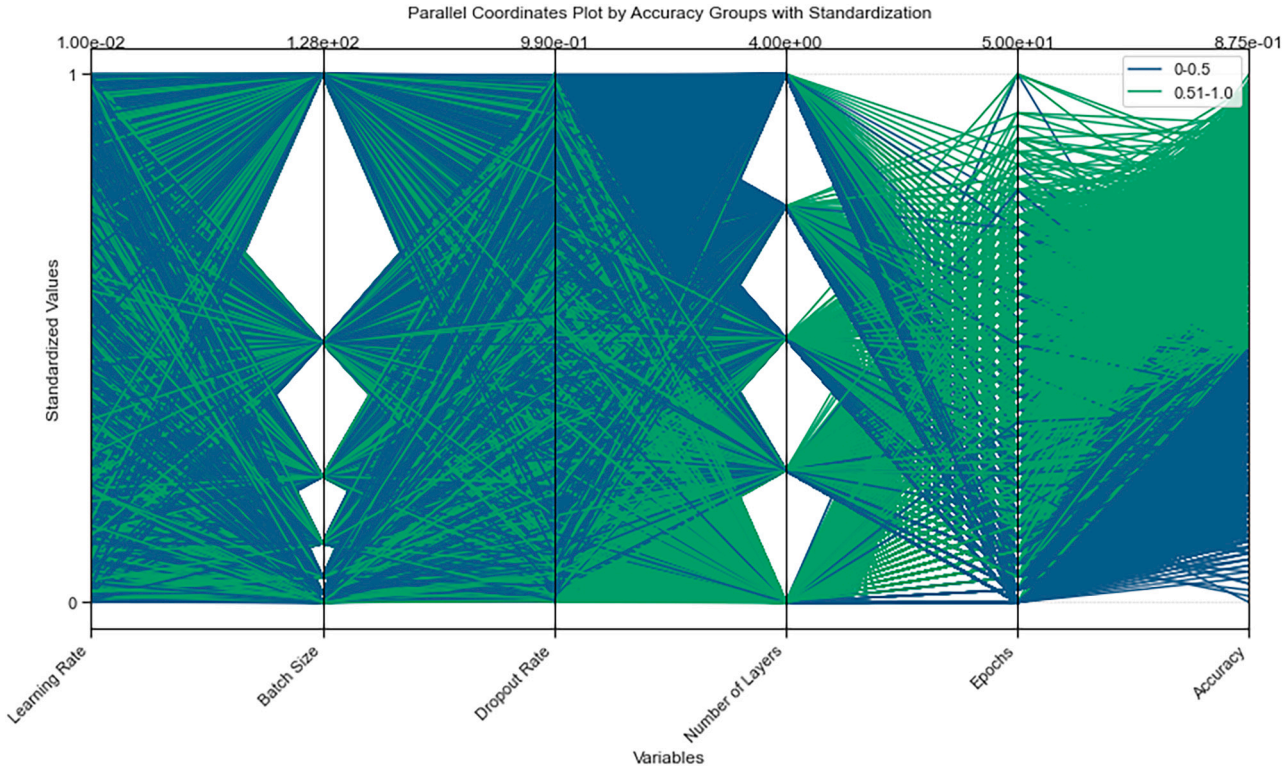

Fig. 5. Hyper-parameter parallel coordinates plot.

uniform distribution, ensuring that different orders of magnitude were adequately explored. This is essential for understanding how a small variation in the learning rate impacts model performance.

Another evaluation metric is the model loss, where the distribution shows that most models have low values, suggesting that the explored hyper-parameter configurations were effective in minimizing error during training. The presence of some outliers with high loss might indicate that certain hyper-parameter combinations appeared not to be suitable and could have been explored more carefully.

As for the distribution of the number of layers, the results reveal a uniform distribution among integers, with approximately 2,500 values for each studied interval. Regarding the distribution of the number of neurons in each layer, it shows a fairly discrete distribution clustered around specific values corresponding to the defined progression. Notice that the number of neurons close to zero increases with the number of layers, and it is crucial to highlight that the number of neurons in deeper layers (1 to 4) directly depends on the total number of layers in the model. This means that in models with few layers, the upper layers will necessarily have a number of neurons equal to zero. This dependency is clearly reflected in the distribution of neurons, which becomes progressively more concentrated around zero in the deeper layers.

After a detailed analysis of hyper-parameter distributions, we investigated the direct connection between these parameters and model performance. Visualizing the interactions between variables enables the identification of hidden patterns that may not be evident in univariate analyses, facilitating the understanding of how different hyper-parameter combinations affect final accuracy. In this context, a parallel coordinate plot is presented, which offers a holistic view of the complex interactions between multiple variables.

The parallel coordinate plot in Fig. 5 illustrates a relation between hyper-parameter and final model accuracy. The lines in the plot are color-coded to represent different accuracy ranges to facilitate the quick identification of hyper-parameter configurations associated with different performance levels. It is important to note that the number of neurons in each layer was excluded from this analysis as they add significant noise to the plot. We remind the reader that data was standardized to ensure that all hyper-parameter values are within the same range from 0 to 1.

The analysis presented by the Parallel Coordinates Plot shows that the lowest accuracy values were frequently associated with hyper-parameter configurations where the number of layers was moderate to high (2, 3, 4 layers). The learning rate and batch size appear to be evenly distributed, as does the dropout rate. The number of epochs, on the other hand, tends to be low. The group with higher accuracy (above 50%) reveals, as already noticeable in the lower accuracy group, that the number of layers is concentrated in fewer layers 0, 1, with the remaining hyper-parameter well-distributed throughout their range. Another type of analysis was performed—studying the linear interactions between variables using the correlation matrix of all hyper-parameter and evaluation metrics. The correlation matrix is depicted in Fig. 6.

The correlation matrix shown in Fig. 6 provides a comprehensive view of the linear relations among the main hyper-parameter of the CNN model and performance metrics such as accuracy, loss, and the number of epochs. Each cell in the matrix shows the Pearson correlation coefficient, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation), with values close to 0 with no linear correlation.

As perceived from the image analysis, the strongest positive correlation is between the number of epochs taken for the model to converge and the model's final accuracy. Observing the negative correlations, the number of layers appears with a correlation of approximately -0.67 with model accuracy, suggesting that for this dataset and model configuration, deeper networks do not necessarily result in better performance, as already demonstrated by the parallel coordinates plot (Fig. 5). Additionally, there is a substantial negative correlation with model loss (-0.57), which is expected, as a decrease in loss is generally associated with increased accuracy, indicating that the model is better fitted to the training data. Regarding the number of epochs needed to converge,

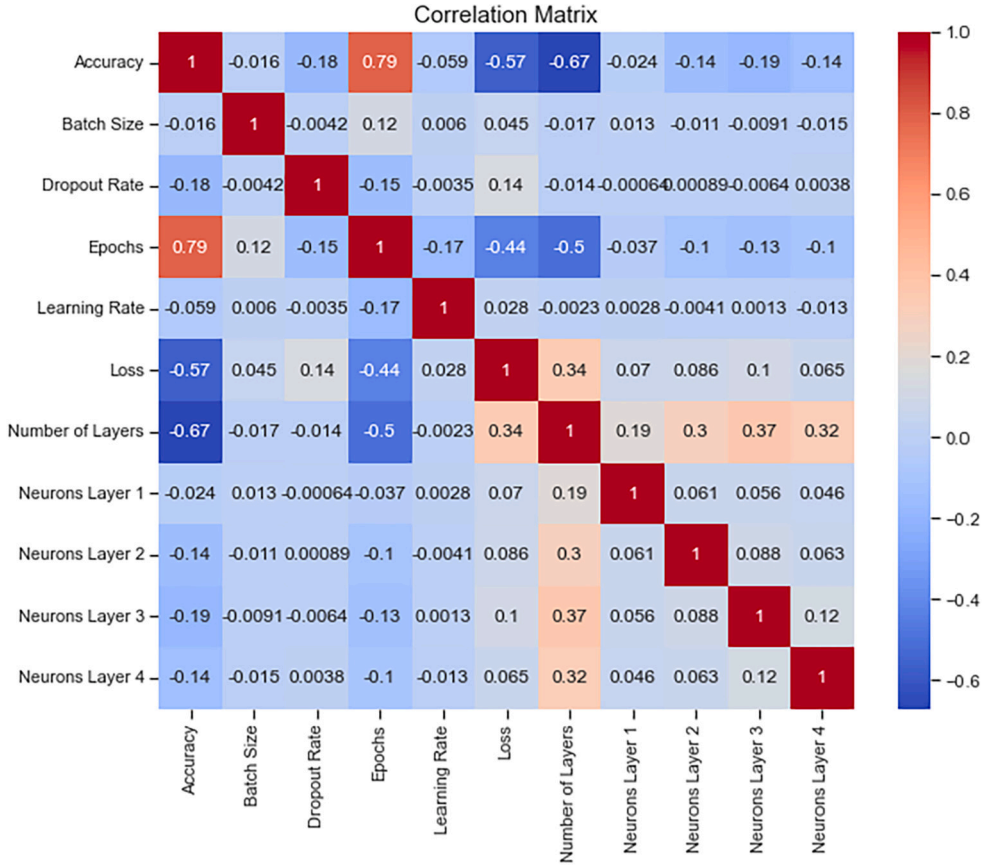

Fig. 6. Hyper-parameter correlation matrix.

some conclusions can be drawn, such as the weak negative correlation (-0.5) between the number of epochs and the number of layers, indicating that configurations with more layers tend to use fewer epochs, possibly to avoid overfitting or due to the additional complexity of deeper networks.

It is interesting to observe the moderate correlation between the number of neurons in different layers, which suggests that model configurations tend to maintain similar proportions of neurons between consecutive layers, possibly to keep the network structure cohesive.

In contrast to these findings, batch size does not significantly correlate with accuracy or with any other hyper-parameter. This suggests that its impact is more dependent on interactions with other specific model configurations. Similarly, the dropout rate does not show strong correlations with accuracy, indicating that, in this case, its influence may be less direct and more related to preventing overfitting in specific configurations.

In conclusion, the correlation matrix analysis reveals that the most influential hyper-parameter for model accuracy is the number of layers, as observed before. Other hyper-parameters, such as batch size and dropout rate, do not exhibit significant correlations, suggesting that their impact on accuracy is more indirect or dependent on complex interactions with other factors. To find such effects, examining how these parameters influence model performance non-linearly is crucial. For this purpose, we used machine learning techniques to capture complex interactions between variables.

To explore the non-linear influence of hyper-parameter on accuracy, a RF Regressor model was implemented. This model was trained with the available data, excluding variables directly related to final performance (such as accuracy, loss, and epochs) to focus on the main variables of interest. The model performance was evaluated using two main metrics: Mean Squared Error (MSE) and the R^2 Score, obtaining values of 0.0061 and 0.8830, respectively.

Table 4
Importance of the variables in the RF model.

Feature	Importance
Number of Layers	0.5284
Neurons Layer 1	0.1984
Dropout Rate	0.0937
Learning Rate	0.0821
Neurons Layer 2	0.0432
Batch Size	0.0273
Neurons Layer 3	0.0194
Neurons Layer 4	0.0076

These results indicate that the RF model was able to capture a large portion of the variation in CNN model accuracy, with an R^2 of 0.8830, suggesting that 88.3% of the accuracy variation can be explained by the considered hyper-parameter. The low MSE reflects the model's precision in predictions.

Subsequently, to identify which hyper-parameter has most significant impact on model performance, the importance of variables in the RF was calculated as shown in Table 4. The table shows that the number of layers was identified as the most influential hyper-parameter, with an importance of 52.84%, validating earlier tendencies. Secondly, the number of neurons in the first layer, with an importance of 19.84%, indicates that the configuration of the first layer, responsible for initial feature extraction, is particularly critical for the model's success. In this case, the dropout rate and learning rate also showed significant importance (9.37% and 8.21%, respectively), confirming their essential role in regularization and fine-tuning of the model. Finally, the subsequent

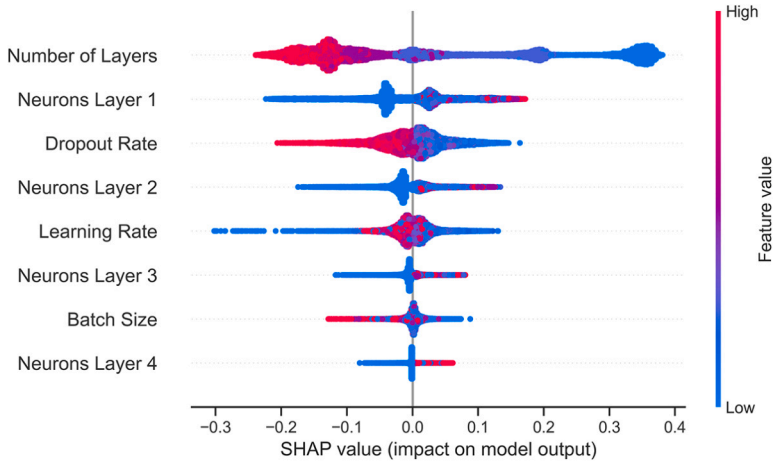

Fig. 7. SHAP graph of hyper-parameter interactions.

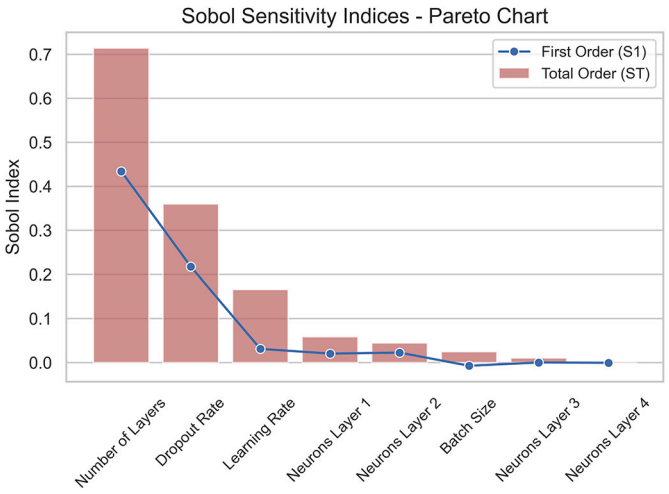

Fig. 8. Sobol analysis of hyper-parameter interactions.

layers (Layers 2, 3, and 4) and batch size show a lower importance, suggesting that while relevant, these parameters have a more subtle impact or are dependent on other configurations.

To complement the previous analyses and provide a more granular view of the effects of hyper-parameter on model performance, the SHAP tool was used. The application of SHAP not only confirmed the importance of the number of layers and the configuration of the first layer but also facilitate to analyze how these hyper-parameters influence accuracy in a non-linear manner.

The SHAP graph presented in Fig. 7 reveals that intermediate values of neurons in the first layer and a moderate configuration of the number of layers tend to maximize accuracy, while extreme values may harm performance. The analysis also highlights that dropout rate and learning rate play an important role, but their impact varies depending on interactions with other hyper-parameter. In contrast, batch size and subsequent layer configurations show a smaller impact, aligning with previous observations that their effect is more context-dependent on the overall model architecture. Thus, the use of SHAP validates the previous conclusions, evidencing the complexity of interactions between hyper-parameters and reinforcing the need for a careful approach in tuning these parameters to optimize model performance.

Complementing these findings, a Sobol analysis was carried out to further quantify the model sensitivity to each hyper-parameter. The results presented in Fig. 8 illustrate the earlier findings on the critical

Table 5
Morris analysis of hyper-parameter interactions.

Variable	μ	μ^*	σ
Learning Rate	0.0680	0.1698	0.2722
Batch Size	-0.0046	0.0348	0.0632
Dropout Rate	-0.0791	0.1118	0.1578
Number of Layers	-0.3406	0.4197	0.3667
Neurons Layer 1	0.1144	0.1212	0.2302
Neurons Layer 2	0.0376	0.0445	0.1180
Neurons Layer 3	0.0319	0.0363	0.0992
Neurons Layer 4	0.0131	0.0151	0.0622

importance of the number of layers. The large impact is revealed in terms of direct effect, reflected by its first-order Sobol index value of S1 = 0.454, and by its interactions with other hyper-parameter, as evidenced by the total-order Sobol index of ST = 0.706. These values indicate that the number of layers not only dominates the direct effect on accuracy but also plays a central role in variable interactions. Similarly, the dropout rate has a significant S1 (0.212) and a considerable ST (0.350), highlighting both its direct influence and interactions with other variables. The learning rate, despite having a relatively low S1 (0.011), shows a higher ST (0.141), suggesting that its relevance increases when considering its complex interactions with other hyper-parameters.

In contrast to this impact, variables such as batch size and subsequent layer configurations obtain very low S1 and ST index values, suggesting a limited impact both directly and indirectly. This corroborates the previous observations made with SHAP. Thus, the Sobol analysis complements and validates the obtained conclusions, highlighting the need to optimize the number of layers and dropout rate while exploring the subtler interactions involving the learning rate to maximize model performance.

In addition to the analyses conducted with SHAP and Sobol, a Morris analysis with results in Table 5 was applied to assess the model hyper-parameter sensitivity. The Morris analysis results reveal important nuances about the impact of hyper-parameters. The “Number of Layers” stood out as the most influential hyper-parameter, with the highest μ^* value of 0.42 and a significant negative value $\mu = -0.34$, indicating that, on average, increasing the number of layers tends to decrease model accuracy. However, the high σ value (0.37) suggests that this relationship is highly variable, likely due to complex interactions with other hyper-parameters.

The “Learning Rate” showed a high mean $\mu^* = 0.17$ and the standard deviation $\sigma = 0.27$, indicating a significant influence, mainly through

non-linear effects and interactions with other variables. The "Dropout Rate" also showed notable influence, with a moderate mean $\mu^* = 0.11$ and standard deviation $\sigma = 0.16$, indicating some variability in its effects, although less pronounced than observed for the number of layers.

Hyper-parameters such as "Batch Size" and the configurations of subsequent layers (Layer 2, Layer 3, and Layer 4) presented relatively low mean μ, μ^* and standard deviation σ, suggesting a smaller direct and indirect impact on model performance. These results align with previous findings that these hyper-parameters play a more limited role and are dependent on the overall context of the architecture.

These results validate the critical role of the "Number of Layers" and the "Learning Rate" in the model and emphasize the need for careful analysis of interactions and non-linear effects associated with these hyper-parameters.

To conclude the sensitivity analysis, it is relevant to compare the insights obtained from the different methodologies employed. Across all techniques RF feature importance, SHAP values, Morris and Sobol indices the number of layers consistently emerged as the most influential hyper-parameter. This convergence strengthens the reliability of this finding, especially considering that RF and SHAP are model-specific, whereas Morris and Sobol are model-agnostic sensitivity analysis tools. Furthermore, both SHAP and Sobol revealed that the learning rate and dropout rate, while not the most critical individually, gained importance when interactions with other hyper-parameters were considered, as shown by high total order Sobol indices and SHAP interaction effects. Some minor discrepancies were observed: for instance, SHAP assigned slightly higher importance to Neurons Layer 1 compared to Sobol, possibly due to its focus on local feature attributions in the trained RF model. Meanwhile, Morris analysis captured a greater sensitivity in the learning rate, evidenced by its high standard deviation, suggesting non-linear effects that are less apparent in linear methods such as correlation analysis. These differences are expected given the distinct assumptions and mechanisms underlying each technique. Therefore, the combined use of multiple sensitivity methods allowed not only cross-validation of findings but also the uncovering of nuanced interactions, reinforcing the robustness of the overall analysis.

5. Discussion

When comparing the optimized models with the control model, a substantial improvement in accuracy and loss minimization was observed. Greater variability was also noted in the results of the optimized models, suggesting that despite the observed average improvements, the models became more sensitive to variations in the datasets, which could indicate a potential risk of overfitting, especially in scenarios with limited data. This is always one of the risks associated with optimizing artificial intelligence models, as optimizing for a specific training/validation set can lead to the model overfitting to the specific patterns of that set, making it less capable of generalizing to new data. This happens because the optimization may fine-tune the model so precisely to the peculiarities of the training data that it loses the ability to adapt to examples that do not share those same characteristics. Consequently, the model may perform excellently on the validation set but fail when exposed to new data, demonstrating reduced robustness and a lower capacity for generalization.

The in-depth sensitivity analysis conducted using SHAP, Sobol, Morris, and RF reveals strong and consistent evidence regarding which hyper-parameters most influence model performance. The number of layers emerged as the dominant factor, affecting both accuracy and loss significantly. All methods corroborated this: Morris showed the highest μ^* value for this parameter, Sobol attributed to it the largest total-order index (ST = 0.706), and SHAP placed it as the variable with the greatest average impact. These findings provide a quantitative foundation for model design decisions. In practice, this suggests that controlling the network depth is critical to balance learning capacity and generalization. Too many layers introduce complexity without consistent performance gain, while shallow configurations underperform.

Interestingly, the first fully connected layer (neurons layer 1) also showed substantial influence, indicating that initial post-convolutional transformations are crucial for extracting meaningful representations.

Dropout and learning rates were found to have indirect but significant effects, particularly through interactions. Their influence was not strongly linear, but SHAP and Sobol analyses revealed that intermediate values in both tend to yield better performance. This reinforces the need for careful tuning rather than relying on fixed empirical values or defaults.

On the other hand, hyper-parameters like batch size and the configuration of deeper layers (neurons in layer 3 and 4) showed consistently low influence across all methods. This supports the hypothesis that increasing complexity does not necessarily improve outcomes once a sufficiently expressive initial architecture is in place.

Overall, the convergence of results from multiple sensitivity analysis techniques provides strong guidance for optimizing CNNs in agricultural image classification tasks. By focusing on fewer but more impactful hyper-parameters, researchers can reduce computational cost while improving accuracy and robustness.

From a practical standpoint, these results offer valuable insights for the development of more efficient and tailored CNN architectures in agriculture. By understanding which hyper-parameters most significantly affect model performance, practitioners can reduce the search space in future applications, focusing their efforts on tuning key variables like the number of layers, the structure of the first dense layer, and regularization strategies. This not only improves model accuracy but also reduces training time and resource consumption, crucial factors in field-based or real-time applications. Furthermore, while this study focused on olive cultivar classification, the methodology and findings are readily transferable to other agricultural tasks involving visual pattern recognition, such as disease detection, growth stage monitoring, or species identification.

6. Conclusion and future works

This study aimed to evaluate the influence of hyper-parameter configurations on the performance of CNNs applied to olive cultivar classification. Through a systematic experimental framework, the variations in architecture and training settings were explored to determine how they affect model accuracy and generalization.

The results consistently identified the number of layers and the number of neurons in the first fully connected layer as the most influential hyper-parameters. Optimal performance was observed with intermediate configurations, namely, two to three dense layers and 256 to 512 neurons, while batch size and deeper layers had marginal effects. Dropout and learning rates also showed moderate influence, particularly through interaction effects.

A sensitivity analysis framework, combining RF, SHAP, Morris, and Sobol methods, was applied to identify these relationships. This multi-perspective approach reinforced the robustness of the findings and enabled interpretable insights into hyper-parameter behavior.

Based on these results, is recommended the use of shallower architectures (up to three dense layers), moderate neuron counts (256–512 in the first layer), dropout rates between 0.3 and 0.5, and learning rates in the range of 10^{-5} to 10^{-4}. These configurations offer a balance between representational power and generalization, avoiding the instability and overfitting commonly associated with deeper or overly complex models.

However, this study has some limitations. While effective for initial exploration, the random search method introduces variability in the results due to its stochastic nature. Moreover, the computational cost of exploring a large hyper-parameter space, especially when combined with repeated cross-validation, was significant. These computational

constraints limited the possibility of testing a wider range of architectures and optimization strategies.

Despite these limitations, the findings offer a practical framework for guiding CNN design in agricultural image classification. By narrowing the search space to the most impactful parameters, practitioners can improve model performance while reducing training time, an important step toward scalable and efficient deployment in real-world precision agriculture systems.

Future work will focus on two main directions. First, we aim to expand this analysis to other CNN architectures, such as EfficientNet or MobileNet, and to explore advanced optimization strategies like Bayesian optimization or reinforcement learning for dynamic hyperparameter tuning. Second, the proposed methodology will be tested on well-established benchmark datasets to assess the generalization of the sensitivity patterns beyond the agricultural domain.

CRediT authorship contribution statement

João Mendes: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **José Lima:** Writing – review & editing, Validation, Supervision. **Lino Costa:** Writing – review & editing, Validation, Supervision. **Eligius M.T. Hendrix:** Writing – review & editing, Validation, Supervision, Conceptualization. **Ana I. Pereira:** Writing – review & editing, Validation, Supervision, Conceptualization.

Ethics approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Human participants and/or animals

This research did not involve human participants or animals.

If any of the sections are not relevant to your manuscript, please include the heading and write 'Not applicable' for that section.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work, the author(s) used ChatGPT and Grammarly in order to assist with drafting and refining the text, as well as for grammar and stylistic improvements. After using these tools, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by national funds through FCT/MCTES (PIDDAC): CeDRI, UID/05757 (DOI: https://doi.org/10.54499/UIDB/05757/2020 and DOI: https://doi.org/10.54499/UIDP/05757/2020), SusTEC, LA/P/0007/2020 (DOI: https://doi.org/10.54499/LA/P/0007/2020) and Algoritmi UIDB/00319/2020.

Data availability

The dataset used in this research is publicly available for access and reuse through the following DOI: https://doi.org/10.34620/dadosipb/TVYE8K. This repository ensures transparency and facilitates the replication of the results presented in this work.

References

[1] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, Y. Miao, Review of image classification algorithms based on convolutional neural networks, Remote Sens. 13 (22) (2021), https://doi.org/10.3390/rs13224712.
[2] J. Mendes, J. Lima, L. Costa, N. Rodrigues, D. Brandão, P. Leitão, A. Pereira, Machine learning to identify olive-tree cultivars, in: International Conference on Optimization, Learning Algorithms and Applications, 2023, pp. 820–835.
[3] J. Mendes, J. Lima, L. Costa, N. Rodrigues, A. Pereira, Deep learning networks for olive cultivar identification: a comprehensive analysis of convolutional neural networks, Smart Agric. Technol. 8 (2024) 100470, https://doi.org/10.1016/j.atech.2024.100470.
[4] I.J. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, MA, USA, 2016.
[5] T.N. Tran, Grid search of convolutional neural network model in the case of load forecasting, Arch. Electr. Eng. 70 (1) (2021) 25–36, https://doi.org/10.24425/aee.2021.136050.
[6] R. Andonie, A.C. Florea, Weighted random search for cnn hyperparameter optimization, Int. J. Comput. Commun. Control 15 (2) (2020), https://doi.org/10.15837/ijccc.2020.2.3868.
[7] Y. Lu, Z. Wang, R. Xie, S. Liang, Bayesian optimized deep convolutional network for electrochemical drilling process, J. Manuf. Mater. Proc. 3 (3) (2019), https://doi.org/10.3390/jmmp3030057.
[8] N.M. Aszemi, P.D.D. Dominic, Hyperparameter optimization in convolutional neural network using genetic algorithms, Int. J. Adv. Comput. Sci. Appl. 10 (6) (2019) 269–278.
[9] S. Kiziloluk, E. Sert, Covid-ccd-net: Covid-19 and colon cancer diagnosis system with optimized cnn hyperparameters using gradient-based optimizer, Med. Biol. Eng. Comput. 60 (6) (2022) 1595–1612, https://doi.org/10.1007/s11517-022-02553-9.
[10] A. Iranfar, M. Zapater, D. Atienza, Multiagent reinforcement learning for hyperparameter optimization of convolutional neural networks, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 41 (4) (2022) 1034–1047, https://doi.org/10.1109/TCAD.2021.3077193.
[11] J. Wu, S. Chen, X. Liu, Efficient hyperparameter optimization through model-based reinforcement learning, Neurocomputing 409 (2020) 381–393, https://doi.org/10.1016/j.neucom.2020.06.064.
[12] X. Qi, B. Xu, Hyperparameter optimization of neural networks based on q-learning, Signal Image Video Process. 17 (4) (2023) 1669–1676, https://doi.org/10.1007/s11760-022-02377-y.
[13] R. Taylor, V. Ojha, I. Martino, G. Nicosia, Sensitivity analysis for deep learning: ranking hyper-parameter influence, in: 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), 2021, pp. 512–516.
[14] M. Wojciuk, Z. Swiderska, K. Siwek, A. Gertych, Improving classification accuracy of fine-tuned cnn models: impact of hyperparameter optimization, Heliyon 10 (2024) 26586, https://doi.org/10.1016/j.heliyon.2024.e26586.
[15] N. Parashar, P. Johri, Enhancing apple leaf disease detection: a cnn-based model integrated with image segmentation techniques for precision agriculture, Int. J. Math. Eng. Manag. Sci. 9 (4) (2024) 943–964, https://doi.org/10.33889/IJMEMS.2024.9.4.050.
[16] R. Kyamelia, R. Subharthy, P. Tapan Kumar, S.C. Sheli, Potato plant leaf disease detection using custom cnn deep net: a step towards sustainable agriculture, in: 2024 ITU Kaleidoscope: Innovation and Digital Transformation for a Sustainable World (ITU K), New Delhi, India, Oct. 21–23, 2024, Istanbul Teknik Universitesi, 2024, pp. 9–15.
[17] J. Mendes, J. Lima, L. Costa, N. Rodrigues, A.I. Pereira, Olive cultivar classification based on leaf images, https://doi.org/10.34620/dadosipb/TVYE8K.
[18] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, Ieee, 2009, pp. 248–255.
[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, https://arxiv.org/abs/1409.1556, 2015.
[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2818–2826.
[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.
[22] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32, https://doi.org/10.1023/A:1010933404324.
[23] D. Mienye, Y. Sun, A survey of ensemble learning: concepts, algorithms, applications, and prospects, IEEE Access 10 (2022) 99129–99149, https://doi.org/10.1109/ACCESS.2022.3207287.

[24] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, https://arxiv.org/abs/1705.07874, 2017.
[25] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 30, Curran Associates, Inc., 2017.
[26] M.D. Morris, Factorial sampling plans for preliminary computational experiments, Technometrics 33 (2) (1991) 161–174, https://doi.org/10.1080/00401706.1991.10484804.
[27] X. Zhang, M. Trame, L. Lesko, S. Schmidt, Sobol sensitivity analysis: a tool to guide the development and evaluation of systems pharmacology models, CPT: Pharmacometr. Syst. Pharmacol. 4 (2015), https://doi.org/10.1002/psp4.6.