Optimizing Transfer Learning and Fine-Tuning Hyperparameters in Image Classification Problems with Firefly Algorithm

Vinícius T. M. G. da Silva[1], **Gustavo F. V. de Oliveira**[1], **Fabrício A. Silva**[1], **Marcus H. S. Mendes**[1]

[1]Instituto de Ciências Exatas e Tecnológicas – Universidade Federal de Viçosa (UFV), Campus Florestal – Florestal – MG – Brazil

{vinicius.tome,gustavo.viegas,fabricio.asilva,marcus.mendes}@ufv.br

Abstract. *Image classification is crucial in computer vision, mainly with Convolutional Neural Networks (CNNs). This paper optimizes transfer learning and fine-tuning hyperparameters of CNNs pre-trained on ImageNet for still image classification. Hyperparameter tuning is a complex task that impacts the classification results. The Firefly Algorithm (FA) was used to optimize these hyperparameters across four datasets with Xception and ResNet-152 architectures. Experiments show that FA enhances model performance, achieving state-of-the-art accuracy on three datasets: Multi-Class Weather (99.11%), Pistachio (100%), and D0 (99.89%). Despite being time-consuming, this approach offers a viable method for improving image classification, mainly with smaller datasets.*

1. Introduction

In recent years, image classification has become a significant task in computer vision [Mohamed et al. 2022], and the ability to accurately categorize and interpret images is critical in numerous domains. One of the most widely used solutions is Convolutional Neural Networks (CNNs), typically pre-trained on a general dataset and then retrained on a specific dataset. This approach retains part of the previous knowledge to increase accuracy, employing Transfer Learning and Fine-Tuning techniques.

Transfer learning offers a significant advantage in image classification tasks [Bloedorn and Webber 2023], taking advantage of pre-trained models on large datasets, such as ImageNet [Deng et al. 2009]. This approach allows for the reuse of learned features, which reduces the need for extensive labeled data and computational resources. It is particularly useful for small datasets, allowing models to achieve high accuracy even with limited training data by fine-tuning pre-trained weights to new tasks. Using transfer learning and fine-tuning methods can sometimes be arduous, requiring considerable time and experimentation with different hyperparameter configurations. Finding the optimal solution often presents an optimization challenge. As a result, many techniques originally developed for optimization problems have been adapted for hyperparameter tuning, such as Bayesian Optimization [Snoek et al. 2012] and Grid Search [Belete and D H 2021]. Metaheuristics are also widely employed in this field due to their adaptability to non-linear optimization problems [Aszemi and Dominic 2019]. However, these methods often face challenges when dealing with large search spaces.

The Firefly Algorithm (FA) [Yang 2010] offers a promising alternative due to its simplicity and flexibility to explore the search space. This study proposes that applying

FA to the hyperparameter tuning of CNNs can achieve superior classification performance compared to other methods, particularly in transfer learning scenarios.

In this work, we used the Firefly Algorithm (FA) to optimize the transfer learning and fine-tuning hyperparameters of CNNs pre-trained on the ImageNet dataset, specifically for still, colored image classification problems. We tested FA on four different datasets and evaluated it using two different CNN architectures. The main contributions of this paper are to extensively test the performance of the FA in this specific task and to demonstrate its effectiveness in different classification problems, such as weather, pistachio type, insect, and cells infected with malaria. We aim to present a relatively simple and comprehensible solution that can be applied to different image classification problems with small datasets of colored images.

The remainder of this paper is organized as follows: Section 2 reviews related work in the field. Section 3 describes the hyperparameters that will be optimized. Section 4 explains how we applied the Firefly Algorithm in this study. Section 5 details additional aspects, such as the datasets and CNN architectures used. Section 6 presents the results obtained, and Section 7 provides our conclusions.

2. Related works

Metaheuristics are a common approach in various studies that require finding an optimal set of hyperparameters in deep learning. One of the studies closest to our approach is [Ayan 2023], which uses a Genetic Algorithm (GA) to optimize transfer learning and fine-tuning hyperparameters. The chromosome of the GA is composed of 4 genes: freezing ratio, dropout rate, optimizer, and the width and depth of fully connected layers. The approach was tested with 3 insect datasets for the classification of crop pests and achieved state-of-the-art accuracy in two datasets: Deng (97.58%) and D0 (99.89%), as well as results close to the state-of-the-art with the IP102 dataset (71.84%).

In [Bacanin et al. 2021], the authors tested the Firefly Algorithm (FA) and also proposed a modified FA to optimize the hyperparameters of CNNs for classifying glioma brain tumors. They optimized hyperparameters such as the number of convolutional layers, number of dense layers, number of filters, dropout rate, optimizer, learning rate, and others. However, they did not use pre-trained models and transfer learning. They tested it in two datasets, the first being a combination of three datasets and the second being the axial brain tumor images dataset. With both datasets, they achieved a state-of-the-art accuracy, being 93.3% with the first dataset and 96.5% with the second.

In [Bacanin et al. 2023], various metaheuristics were tested to optimize the hyperparameters of deep learning models for energy load forecasting, and one of the metaheuristics tested was the FA. However, they did not apply it to an image classification problem. The firefly representation comprised the number of neurons, learning rate, training epochs, and dropout.

In [Golnoori et al. 2023], they tested the Genetic Algorithm, Particle Swarm Optimization, and Differential Evolution Algorithm to optimize the hyperparameters of both pre-trained models and models trained from scratch. They optimized parameters such as optimization function, number of neurons, drop rate, and others, and tested it on the ISIC-2018 and ISIC-2017 datasets.

In [Li et al. 2022] the authors tested GA to optimize especificaly the trainability of layers of the transfer model. In [Lai et al. 2023] the Gray Wolf Optimizer is used to optimize the number of neurons in the hidden layers of the neural network and the weight values between neurons and biases, for skin cancer diagnosis. They achieved 97.09% and 95.17% accuracy with the ISIC-2016 and ISIC-2017 datasets, respectively.

In contrast to previous studies, our objective is to utilize the FA to optimize the number of trainable layers in a pre-trained model, as well as other parameters for further fine-tuning, and evaluate its effectiveness as a potential solution for image classification tasks.

3. Transfer learning and Fine Tunning hyperparameters

One of the main techniques for transferring the knowledge of a CNN that has learned to solve one problem is to freeze some of the layers from the model. By doing so, these layers are not updated during training, allowing the original knowledge to be retained while the model adapts to a new problem. Other hyperparameters can also be adjusted to fine-tune the model and improve its performance. This section focuses on the specific hyperparameters selected for optimization in our study.

The first hyperparameter is the **freezing ratio**, which represents the percentage of layers of the model that will be frozen during training. It is one of the most impactful parameters in the transfer learning technique, affecting both the accuracy of the model and the training time. The freezing ratio ranges from 0 to 1, with a minimum variation of 0.01 in this work.

The second is the **dropout rate**, which determines the proportion of neurons in a layer that will be randomly dropped during each training iteration. This technique helps prevent overfitting by ensuring the model does not become overly reliant on specific neurons, promoting more robust learning. The dropout rate ranges from 0 to 1, with a minimum variation of 0.01 used in this study.

The third hyperparameter is the **optimizer**, which influences how the model weights are updated during training. Optimizers determine the strategy for minimizing the loss function by adjusting the learning rate and updating weights based on gradients computed from the training data. The choice of optimizer can significantly impact the convergence rate and overall performance of the model. This study evaluates several optimizers, including Adam, SGD, RMSprop, Adadelta, Adagrad, and Adamax.

The fourth hyperparameter is the **number of fully connected layers** (depth), which is responsible for integrating features extracted by previous layers and making final predictions. The depth and size of these layers can influence the capacity of the network to learn complex patterns and generalize from the data. In this study, the depth will range from 1 to 5.

The fifth hyperparameter is the **number of neurons** (width) in each fully connected layer, which determines the number of units available to process and integrate features extracted by previous layers. A larger number of neurons can increase the capacity of the model to learn more intricate features but may also lead to overfitting and higher computational costs. However, too few neurons might limit the performance of the network. In this study, the width of the fully connected layers is evaluated with values of 64, 128, 256,

512, 1024, and 2048.

The sixth and final hyperparameter to be optimized is the **L2 regularization parameter**, which controls the strength of the L2 regularization applied to the model. L2 regularization, also known as weight decay, adds a penalty to the loss function proportional to the square of the magnitude of the weights. This technique helps prevent overfitting by discouraging the model from assigning excessive importance to any single weight, thus promoting simpler and more generalizable models. This study will evaluate the L2 regularization parameter with values ranging from 0.001 to 0.1, with a minimum variation of 0.001.

4. Firefly Algorithm

To optimize these parameters, we chose the Firefly Algorithm (FA), a well-known metaheuristic that is easy to adapt to various optimization problems. Developed by Xin-She Yang in 2008, FA is inspired by the behavior of fireflies in nature, which emit light to attract mates. In the context of optimization, a firefly represents a potential solution, and its "brightness"indicates the quality or fitness of that solution. The algorithm iteratively updates the positions of fireflies, with each firefly moving according to the brightness of other fireflies, being attracted to brighter ones. The brightness level and the distance from the target firefly influence the movement of each firefly.

Since this is a maximization problem, the brightness $I_\mathbf{i}$ of a $\text{firefly}_\mathbf{i}$ will be equal to the objective function $f(x_\mathbf{i})$, which we will call fitness. The fitness, as shown by Equation (1), in this problem will be how the model performs with the current set of hyperparameters, and inspired by [Ayan 2023] this will be measured by the mean value of four metrics: accuracy, precision, recall and F1 score.

$$fitness = \frac{accuracy + precision + recall + F1\ score}{4} \tag{1}$$

Since training a CNN is a task that requires time and typically increases proportionally with the dataset size, running the FA in this scenario is very time-consuming. In [Ayan 2023], they limited the Genetic Algorithm to a population of 10 through 10 iterations, which limits the algorithm to 100 fitness evaluations. Similarly, we restrict the population size n and the number of generations. In this case, $n = 5$ and the number of generations is 10, limiting the number of fitness evaluations to approximately 100.

In our implementation, each variable of the firefly is represented by a number between 0 and 100. To convert these variables into the equivalent hyperparameters, whenever a solution is to be evaluated, each variable is modified to be used in the model. The freezing ratio and dropout rate are multiplied by 10^{-2}, the L2 regularization parameter is multiplied by 10^{-3}, and for the discrete variables (optimizer, number of fully connected layers, and neuron number), the firefly variable is divided into equal parts, each part representing an option of the hyperparameter.

Due to the limitations in population size and the number of generations, as well as the characteristics of the fireflies, we made a few adaptations to the FA parameters. The most significant change was to $\beta_\mathbf{0}$, which is typically set to 1 [Yang 2010]. In our implementation, we initially set $\beta_\mathbf{0}$ to 60 and decreased it by 7 with each iteration. This

adjustment was necessary because the large variable range and limited iterations would otherwise restrict the fireflies movements, reducing search area exploration. The final value was determined through experimentation. Consequently, firefly attractiveness was higher in the initial iterations, enhancing early exploration. Other parameters were set to standard values: α was set to 0.5 and decreased by 0.02 each iteration, while the light absorption coefficient γ was set to 0.1.

5. Materials and Methods

To test the proposed method, we needed to choose a CNN model and datasets where it could be applied. First, we selected two models pre-trained with the ImageNet dataset, and then we chose four different datasets to test with each model.

5.1. CNN models

We decided to use the models provided by the Keras library for training and chose models with good top-1 and top-5 accuracy and relatively low model complexity [Bianco et al. 2018].

The first model we chose for our experiments was the Xception model [Chollet 2017], which is built upon the Inception architecture by replacing standard convolutional with depthwise separable convolutions, which helps to reduce computational cost while improving accuracy. This model has a good top-1 and top-5 accuracy on the ImageNet dataset and can be used in tasks that require a balance between accuracy and computational efficiency.

The second model we chose was the ResNet-152 model [He et al. 2015], which is part of the ResNet family and is known for its deep residual learning framework incorporating residual blocks to address the vanishing gradient problem. With 152 layers, this model achieves high accuracy on image classification tasks while maintaining relatively low computational complexity compared to its depth.

Both models are well-established benchmarks, demonstrating strong accuracy on the ImageNet dataset while maintaining a suitable computational complexity. Although newer models are available, Xception and ResNet-152 offer a practical balance between accuracy and resource demands, while also being stable platforms for evaluating the FA impact on hyperparameter optimization.

5.2. Datasets

FA in this case is very time-consuming, as mentioned. Since we decided to run it multiple times with each CNN model, we are limited to smaller-sized datasets. We chose four datasets of increasing size for testing, with the larger datasets being tested fewer times. In all, the data was initially split randomly into 80% for training and 20% for testing. From the training set, 12.5% of the images were randomly selected for validation, resulting in a final distribution of 70% for training, 10% for validation, and 20% for testing.

The first dataset we used is the Multi-Class Weather dataset [Gbeminiyi 2018], which provides outdoor images totaling 1,125 colored images of different weather conditions. It is divided into four classes: Cloudy (300 images), Rainy (215 images), Shine (233 images), and Sunshine (356 images). We executed the Firefly Algorithm with this dataset 24 times for each CNN model.

The second dataset we used was the Pistachio Image Dataset [Singh et al. 2022, Ozkan et al. 2021], which provides colored images of two types of pistachios: Kirmizi pistachio and Siirt pistachio. The dataset contains 2,148 images divided into two classes: 1,232 images of the Kirmizi type and 916 images of the Siirt type. This dataset was also tested with the Firefly Algorithm 24 times for each model.

The third dataset we chose was the D0 dataset [Xie et al. 2018], which provides colored images of insects and is useful for classifying crop pests. The dataset contains 4,508 images, divided into 40 species of insects. This dataset was tested with the Firefly Algorithm 6 times for each model.

The fourth and final dataset used was the Malaria dataset [Rajaraman et al. 2018], which provides images of cells useful for identifying cells infected with malaria. The dataset contains 27,558 images, divided equally between two classes: infected cells and uninfected cells. We applied horizontal and vertical flip data augmentation to the training and validation subsets. We executed the FA with this dataset only once for each model.

5.3. Training setup

We used the TensorFlow and Keras libraries to execute the algorithm and ran it on an Intel Core i5-8600 with 16 GB of RAM and an NVIDIA GTX 4060 with 8 GB of VRAM, on a Linux Mint 21.3 operational system. Each training was conducted over 32 epochs with a batch size of 32. We applied the early stopping method to stop training after 13 epochs without an improvement in validation accuracy. Source code in footnote.[1]

6. Results

In this section, we present the results of multiple algorithm executions. Multiple executions are important because the initial population of fireflies can limit the search area. We then highlight the best solutions for each case and compare these results with those from other studies that applied different training methods to the same datasets. Notice that each column from the Tables 1, 2, and 3 represents the metrics separately, meaning that each line does not necessarily represent the same execution. In the case of the Malaria dataset, the metrics are in Table 5 since there was only one execution.

From the results of the multiple executions, it is noticeable that the best accuracy is not consistently obtained. This highlights the limitations of many metaheuristics, which may not be able to explore the entire search area of the problem. In the case of the Firefly Algorithm, this limitation is especially pronounced due to the initial positions of the fireflies. The movement of each firefly is directed toward other fireflies, leaving areas outside their pathways unreachable. However, from the multiple executions, we can observe that the mean values obtained are close to the best values, and the standard deviation is relatively low.

The algorithm performed best with the Pistachio dataset (see Table 2), achieving 100% accuracy with the Xception model in 16 out of 24 executions. In the case of the ResNet-152 model, it achieved the best result in 12 out of 24 executions. With the Multi-Class Weather dataset(see Table 1), the best accuracy was achieved only 3 times out of 24 executions with the Xception model, and the best overall accuracy of 99.11% was

[1] https://github.com/viniciustmgs/POC

Table 1. Results of the firefly algorithm with the Multi-Class Weather dataset

Xception				
	Accuracy	**Precision**	**Recall**	**F1 score**
Best	0.986666	0.986732	0.991031	0.986666
Worst	0.964444	0.963458	0.968468	0.955555
Mean	0.979259	0.978891	0.980649	0.975925
Standard deviation	0.005689	0.005839	0.005429	0.007361
ResNet-152				
	Accuracy	**Precision**	**Recall**	**F1 score**
Best	0.991111	0.990955	0.991031	0.986666
Worst	0.959999	0.958766	0.959999	0.915555
Mean	0.980370	0.979955	0.983420	0.976666
Standard deviation	0.007017	0.007211	0.005912	0.013653

Table 2. Results of the firefly algorithm with the Pistachio image dataset

Xception				
	Accuracy	**Precision**	**Recall**	**F1 score**
Best	1.000000	1.000000	1.000000	1.000000
Worst	0.993023	0.992849	0.993023	0.993023
Mean	0.998837	0.998810	0.998837	0.998837
Standard deviation	0.001898	0.001943	0.001898	0.001898
ResNet-152				
	Accuracy	**Precision**	**Recall**	**F1 score**
Best	1.000000	1.000000	1.000000	1.000000
Worst	0.995348	0.995250	0.995348	0.995348
Mean	0.998449	0.998416	0.998449	0.998449
Standard deviation	0.001733	0.001770	0.001733	0.001733

Table 3. Results of the firefly algorithm with the D0 dataset

Xception				
	Accuracy	**Precision**	**Recall**	**F1 score**
Best	0.998891	0.999090	0.998886	0.994456
Worst	0.991131	0.989267	0.993311	0.982261
Mean	0.994456	0.993438	0.996465	0.989283
Standard deviation	0.003263	0.004132	0.001973	0.003768
ResNet-152				
	Accuracy	**Precision**	**Recall**	**F1 score**
Best	0.997782	0.998147	0.997772	0.993348
Worst	0.992239	0.990280	0.995531	0.987804
Mean	0.995750	0.995317	0.996841	0.991685
Standard deviation	0.001743	0.002511	0.000769	0.002098

achieved only once with the ResNet-152 model. In that regard, it is noticeable that the Multi-Class Weather dataset had a more limited set of hyperparameters that achieved a

close to optimal result, while the Pistachio dataset had different areas of the search space that achieved an optimal result.

In the case of the D0 dataset (see Table 3), the algorithm achieved the best result with the Xception model twice out of the 6 executions, and with the ResNet-152 model, it achieved the best result once out of the 6 executions. The mean values were relatively close to the best results, with low variation.

From these results, it is possible to infer that, in the case of the Malaria dataset (see Table 5), the optimal set of hyperparameters was possibly not obtained from only one execution. It may also have only achieved a local maximum. To further test this, more executions of the algorithm are needed. However, this approach would be unfeasible due to the significant time requirements. In cases where improvement in accuracy is urgently needed, conducting more executions is a possible solution, but it could very well achieve only small improvements.

Table 4. Best set of hyperparameters achieved for each dataset

Multi-Class Weather dataset						
Model	**Freezing ratio**	**Dropout**	**Optimizer**	**FC layers**	**Neurons**	**L2**
XC	0.04	0.21	Adamax	5	512	0.023
RN-152	0.73	0.46	SGD	2	64	0.075
Pistachio dataset						
XC	0.31	0.39	RMSprop	3	128	0.031
RN-152	0.57	0.45	Adam	2	512	0.012
D0 dataset						
XC	0.41	0.27	Adamax	1	1024	0.037
RN-152	0.57	0.14	SGD	1	512	0.036
Malaria dataset						
XC	0.32	0.56	RMSprop	2	256	0.003
RN-152	0.78	0.37	SGD	2	1024	0.015

Table 5. Metrics of the best solution achieved for each dataset

Multi-Class Weather dataset				
	Accuracy	**Precision**	**Recall**	**F1 score**
Xception	0.986666	0.986732	0.986666	0.986666
ResNet-152	0.991111	0.990955	0.991031	0.982222
Pistachio dataset				
Xception	1.000000	1.000000	1.000000	1.000000
ResNet-152	1.000000	1.000000	1.000000	1.000000
D0 dataset				
Xception	0.998891	0.999090	0.998880	0.988913
ResNet-152	0.997782	0.998147	0.997772	0.993348
Malaria dataset				
Xception	0.971516	0.971516	0.971516	0.971516
ResNet-152	0.971153	0.971152	0.971153	0.971153

Table 4 shows the values of hyperparameter for the best solutions, and Table 5 shows the subsequent results from these solutions.

Table 6. Comparison with other works

Multi-Class Weather dataset				
research method	**Accuracy**	**Precision**	**Recall**	**F1 score**
[Tian et al. 2021]	93.50%	93.42%	92.9%	93.15%
[Al-Haija et al. 2020]	98.22%	96.50%	-	-
XC (this work)	98.66%	98.67%	98.66%	**98.66%**
RN-152 (this work)	**99.11%**	**99.09%**	**99.10%**	98.22%
Pistachio dataset				
[Avuçlu 2023]	88.57%	88.44%	-	88.28%
[Ozkan et al. 2021]	94.18%	95.13%	-	94.94%
[Lisda et al. 2023]	96.00%	96.00%	96.00%	96.00%
[Singh et al. 2022]	98.84%	98.28%	-	98.84%
XC (this work)	**100.00%**	**100.00%**	**100.00%**	**100.00%**
RN-152 (this work)	**100.00%**	**100.00%**	**100.00%**	**100.00%**
D0 dataset				
[Saranya et al. 2024]	98.02%	98.92%	97.88%	98.88%
[Ayan et al. 2020]	98.81%	98.88%	98.81%	98.81%
[Ung et al. 2021]	99.78%	99.66%	99.71%	99.68%
RN-152 (this work)	99.78%	99.81%	99.77%	99.33%
[Nanni et al. 2022]	99.81%	99.83%	99.77%	99.71%
[Ayan 2023]	99.89%	99.82%	**99.91%**	**99.86%**
XC (this work)	**99.89%**	**99.91%**	99.89%	98.89%
Malaria dataset				
[Sibinraj et al. 2024]	88.00%	88.00%	88.00%	88.00%
[Behera et al. 2024]	96.24%	-	-	95.5%
[Narayanan et al. 2023]	97.10%	-	-	-
RN-152 (this work)	97.11%	97.11%	97.11%	97.11%
XC (this work)	97.15%	97.15%	97.15%	97.15%
[Gesmundo 2022]	97.46%	-	-	-
[Mujahid et al. 2024]	97.57%	96.59%	98.62%	97.55%
[Schwarz Schuler et al. 2022]	97.61%	-	-	-
[Rajaraman et al. 2018]	98.60%	-	-	98.70%
[Mishra 2021]	**99.37%**	**99.52%**	**99.23%**	**99.37%**

Table 6 shows that, compared with other works that applied different deep learning techniques, our proposed method achieves astounding results with the Weather, Pistachio, and D0 datasets. It improves the current state-of-the-art performance with the Weather and Pistachio datasets, while achieving a result equal to the state-of-the-art with the D0 dataset.

On the other hand, with the Malaria dataset, the algorithm could have improved the accuracy compared to other works that achieved better results with the same dataset. In a medical image classification problem, such as the classification of malaria-infected cells in this case, a model with higher accuracy, such as the one from [Mishra 2021], is

preferable. However, the proposed method still achieved good results compared to other works implementing more complex methods.

7. Conclusion

In this paper, we applied the Firefly Algorithm to optimize the percentage of trainable layers and other fine-tuning hyperparameters in pre-trained models for image classification problems. Based on our experiments with different datasets, the Firefly Algorithm can successfully optimize the set of parameters for this specific task. We achieved state-of-the-art performance with three of the four datasets tested, obtaining 99.11% accuracy with the Multi-Class Weather dataset, 100% accuracy with the Pistachio dataset, and 99.89% accuracy with the D0 dataset.

The method also achieved good results with the Malaria dataset, with an accuracy of 97.15%. Although these results are comparable to other studies, they did not reach the state-of-the-art.

From this, we can determine that the proposed Firefly Algorithm method is generally effective for this specific task. However, based on the results and the time limitations previously mentioned, it is best suited for cases with small-sized datasets, since in this case it is also possible to increase the population of fireflies.

Future work can test different metaheuristics with the same set of hyperparameters. Different combinations of parameters and other CNN architectures can also be explored. Parallelization strategies or surrogate models could make the computational cost manageable for larger datasets.

References

Al-Haija, Q. A., Smadi, M. A., and Zein-Sabatto, S. (2020). Multi-class weather classification using resnet-18 cnn for autonomous iot and cps applications. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1586–1591.

Aszemi, N. M. and Dominic, P. (2019). Hyperparameter optimization in convolutional neural network using genetic algorithms. *International Journal of Advanced Computer Science and Applications*, 10(6).

Avuçlu, E. (2023). Classification of pistachio images with the resnet deep learning model. *Selcuk Journal of Agriculture and Food Sciences*, 37(2):291–300.

Ayan, E. (2023). Genetic algorithm-based hyperparameter optimization for convolutional neural networks in the classification of crop pests. *Arabian Journal for Science and Engineering*, 49(3):3079–3093.

Ayan, E., Erbay, H., and Varçın, F. (2020). Crop pest classification with a genetic algorithm-based weighted ensemble of deep convolutional neural networks. *Computers and Electronics in Agriculture*, 179:105809.

Bacanin, N., Bezdan, T., Venkatachalam, K., and Al-Turjman, F. (2021). Optimized convolutional neural network by firefly algorithm for magnetic resonance image classification of glioma brain tumor grade. *Journal of Real-Time Image Processing*, 18(4):1085–1098.

Bacanin, N., Stoean, C., Zivkovic, M., Rakic, M., Strulak-Wójcikiewicz, R., and Stoean, R. (2023). On the benefits of using metaheuristics in the hyperparameter tuning of deep learning models for energy load forecasting. *Energies*, 16(3).

Behera, N., Das, S., Singh, A. P., Swain, A. K., and Rout, M. (2024). Cnn - based medical image classification models for the identification of pneumonia and malaria. In *2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC)*, pages 01–06.

Belete, D. and D H, M. (2021). Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results. *International Journal of Computers and Applications*, 44:1–12.

Bianco, S., Cadene, R., Celona, L., and Napoletano, P. (2018). Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277.

Bloedorn, F. G. and Webber, C. G. (2023). Transfer learning applied to a classification task: a case study in the footwear industry. *IEEE Latin America Transactions*, 21(3):427–433.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Gbeminiyi, A. (2018). Multi-class weather dataset for image classification. *Mendeley Data*, 6:15–23.

Gesmundo, A. (2022). A continual development methodology for large-scale multitask dynamic ml systems.

Golnoori, F., Boroujeni, F. Z., and Monadjemi, A. (2023). Metaheuristic algorithm based hyper-parameters optimization for skin lesion classification. *Multimedia Tools and Applications*, 82(17):25677–25709.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

Lai, W., Kuang, M., Wang, X., Ghafariasl, P., Sabzalian, M. H., and Lee, S. (2023). Skin cancer diagnosis (scd) using artificial neural network (ann) and improved gray wolf optimization (igwo). *Scientific Reports*, 13(1):19377.

Li, C., Jiang, J., Zhao, Y., Li, R., Wang, E., Zhang, X., and Zhao, K. (2022). Genetic algorithm based hyper-parameters optimization for transfer convolutional neural network. In *International Conference on Advanced Algorithms and Neural Networks (AANN 2022)*, volume 12285, pages 232–241. SPIE.

Lisda, L., Kusrini, K., and Ariatmanto, D. (2023). Classification of pistachio nut using convolutional neural network. *Inform : Jurnal Ilmiah Bidang Teknologi Informasi dan Komunikasi*, 8(1):71–77.

Mishra, S. (2021). Malaria parasite detection using efficient neural ensembles. *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, 3(3):119–133.

Mohamed, A., Hemeida, A., and Hassan, M. (2022). Image classification based deep learning: A review. *Aswan University Journal of Sciences and Technology*, 2.

Mujahid, M., Rustam, F., Shafique, R., Montero, E. C., Alvarado, E. S., de la Torre Diez, I., and Ashraf, I. (2024). Efficient deep learning-based approach for malaria detection using red blood cell smears. *Scientific Reports*, 14(1):13249.

Nanni, L., Manfè, A., Maguolo, G., Lumini, A., and Brahnam, S. (2022). High performing ensemble of convolutional neural networks for insect pest image detection. *Ecological Informatics*, 67:101515.

Narayanan, B. N., de Silva, M. S., and Hardie, R. C. (2023). A patient specific algorithm for plasmodium malaria detection on cell images. In *NAECON 2023 - IEEE National Aerospace and Electronics Conference*, pages 258–262.

Ozkan, I., Koklu, M., and Saraçoğlu, R. (2021). Classification of pistachio species using improved k-nn classifier. *Health*, 23:e2021044.

Rajaraman, S., Antani, S. K., Poostchi, M., Silamut, K., Hossain, M. A., Maude, R. J., Jaeger, S., and Thoma, G. R. (2018). Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images. *PeerJ*, 6:e4568.

Saranya, T., Deisy, C., and Sridevi, S. (2024). Efficient agricultural pest classification using vision transformer with hybrid pooled multihead attention. *Computers in Biology and Medicine*, 177:108584.

Schwarz Schuler, J. P., Also, S. R., Puig, D., Rashwan, H., and Abdel-Nasser, M. (2022). An enhanced scheme for reducing the complexity of pointwise convolutions in cnns for image classification based on interleaved grouped filters without divisibility constraints. *Entropy*, 24(9).

Sibinraj, V., Gafoor, F., Anandhu, P., and Anoop, V. (2024). Automated malaria cell image classification using convolutional neural networks. *TechRxiv*.

Singh, D., Taspinar, Y. S., Kursun, R., Cinar, I., Koklu, M., Ozkan, I. A., and Lee, H.-N. (2022). Classification and analysis of pistachio species with pre-trained deep learning models. *Electronics*, 11(7).

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms.

Tian, M., Chen, X., Zhang, H., Zhang, P., Cao, K., and Wang, R. (2021). Weather classification method based on spiking neural network. In *2021 International Conference on Digital Society and Intelligent Systems (DSInS)*, pages 134–137.

Ung, H. T., Ung, H. Q., and Nguyen, B. T. (2021). An efficient insect pest classification using multiple convolutional neural network based models.

Xie, C., Wang, R., Zhang, J., Chen, P., Dong, W., Li, R., Chen, T., and Chen, H. (2018). Multi-level learning features for automatic classification of field crop pests. *Computers and Electronics in Agriculture*, 152:233–241.

Yang, X.-S. (2010). *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.