scientific reports

Check for updates

OPEN

Optimizing VGG16 deep learning model with enhanced hunger games search for logo classification

Mohammed Hussain[1], Thaer Thaher[2]✉, Mohamed Basel Almourad[1] & Majdi Mafarja[3]

Accurate classification of logos is a challenging task in image recognition due to variations in logo size, orientation, and background complexity. Deep learning models, such as VGG16, have demonstrated promising results in handling such tasks. However, their performance is highly dependent on optimal hyperparameter settings, whose fine-tuning is both labor-intensive and time-consuming. Swarm intelligence algorithms have been widely adopted to solve many highly nonlinear, multimodal problems and have succeeded significantly. The Hunger Games Search (HGS) is a recent swarm intelligence algorithm that has shown good performance across various applications. However, the standard HGS still faces limitations, such as restricted population diversity and a tendency to get trapped in local optima, which can hinder its effectiveness. In this paper, we propose an optimized deep learning architecture called EHGS-VGG16 designed based on the VGG16 model and boosted by an enhanced Hunger Games Search (EHGS) algorithm for hyperparameter tuning. The proposed enhancement to HGS involves modified search strategies, incorporating the concepts of "local best" and a "local escaping mechanism" to improve its exploration capability. To validate our approach, the evaluation is conducted in three folds. First, the EHGS algorithm is evaluated through 30 real-valued benchmark functions from the IEEE CEC2014 suite. Second, a custom-developed VGG16 model is tested on the Flickr-27 logo classification dataset and compared against state-of-the-art deep learning models such as ResNet50V2, InceptionV3, DenseNet121, EfficientNetB0, and MobileNetV2. Finally, EHGS is integrated into the VGG16 model to optimize its hyperparameters. The experimental results show that VGG16 outperformed the other counterparts with an accuracy of 0.956966, a precision of 0.957137, and a recall of 0.956966. Moreover, the integration of EHGS further improved classification quality by 3%. These findings highlight the potential of combining evolutionary optimization techniques with deep learning for enhanced accuracy in log classification tasks.

Keywords Computer vision, Logo classification, Convolution neural network, Hunger games search, Hyperparameters, Metaheuristics

Computer vision is a cutting-edge technology that aims to enable artificial systems to extract valuable information from images[1]. It is an important part of artificial intelligence (AI), used across various sectors such as agriculture for studying crops and managing pests[2]; business, by enhancing customer experiences through face recognition and emotion analysis[3]; and healthcare, advancing medical diagnostics and improving patient care[4]. Furthermore, computer vision is a game-changer for companies in brand management and marketing. Furthermore, computer vision is a game-changer for companies in brand management and marketing. It excels at identifying images and detecting logos. This gives them powerful tools to effectively assess market trends and monitoring their brand's presence.

Classification of logos is an important task in computer vision. This particular technique entails the process of searching for and identifying logos within images or videos[5,6]. The aim is to automatically detect the presence of logos and, for the purposes of classification, identify which brand or organization each logo belongs to. Logo classification plays a significant role in diverse applications, such as copyright protection (e.g., detecting unauthorized use of logos)[7], placing targeted advertisements (e.g., showing relevant ads based on identified logos), brand information retrieval (e.g., gathering details about brands through their logos)[8], brand monitoring on social media (e.g., tracking the brand presence and interactions on platforms like Facebook and Instagram)[9], augmented reality enhancements (e.g., implementation of logo recognition in augmented reality experiences to

[1]College of Technological Innovation, Zayed University, Dubai, United Arab Emirates. [2]Department of Computer Systems Engineering, Arab American University, Jenin, Palestine. [3]Department of Computer Science, Birzeit University, P.O. Box 14 Birzeit, West Bank, Palestine. ✉email: Thaer.Thaher@aaup.edu

provide more context)[10], and traffic management (e.g., detection of logos on vehicles for traffic management and analysis)[11]. These practical applications ensure the proper recognition of logos will allow businesses to handle their brand effectively, defend their property, and bring forth content that is targeted and meaningful.

The classification of logos is a challenging task due to the wide variety of designs with different colors, sizes, orientations, and complex backgrounds[12]. These challenges are further compounded by occlusion, where parts of the logo are hidden, and distortion due to varying camera angles or compression artifacts. Additionally, the presence of inconsistent illumination conditions can confuse shape and color recognition, making it difficult to accurately identify logos in real-world scenarios. Existing logo classification methods often struggle to generalize well in these conditions, leading to reduced accuracy and performance in diverse environments. Therefore, scholars are actively researching solutions to overcome these obstacles[13]. Earlier methods, based on hand-crafted features like scale-invariant feature transform (SIFT) and speed-up robust features (SURF), were once popular for logo classification. However, these approaches have proven insufficient to address the complexities of modern logo designs, which vary widely in color, shape, and size.

To address these challenges, researchers are now focusing on machine learning (ML) and advanced deep learning (DL) methods[6,14,15]. Utilizing neural networks and large training datasets leads to significant performance enhancements. DL is a subfield of ML in the AI domain. It is concerned with algorithms that mimic the structure and function of the human brain in analyzing data and creating patterns for use in decision-making. For this purpose, DL comprises a multi-layered structure of algorithms called neural networks. Recently, DL has become one of the most promising techniques for solving classification problems. Deep learning-based techniques have become popular in signal processing, computer vision, and computer-aided diagnosis (CAD). The four significant architectures of deep learning networks are Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Generative Adversarial Networks (GANs), and Recursive Neural Networks (RvNNs)[16]. One of the most popular methods for information classification based on DL techniques is CNN.

CNNs are a unique class of neural networks used mainly for analyzing digital images[17]. They are widely known for their ability to automatically extract hierarchical features from images, making them effective in various computer vision tasks, including logo classification. CNNs excel in various visual tasks, including object detection, image classification[18], facial recognition[19], generation of textual image descriptions[20], and image segmentation[21]. The first CNN was introduced by Lecun et al.[22] and was used for recognizing numbers. More advanced CNN models, such as AlexNet[23], VGG[24], GoogleNet[25], ResNet[26], DenseNet[27], and SENet[28], have furthered the evolution of CNN architectures and set new benchmarks in computer vision tasks. Recent innovations in CNN architecture often address specific limitations in older models. For instance, ResNet proposed shortcut hyper-connections to overcome the issue of vanishing gradients that arise with increasing the number of layers in a CNN. However, despite their success, many existing deep learning models, including CNNs, lack robustness in handling variations in logo appearance and struggle to maintain high performance when logos are partially occluded or significantly distorted. Additionally, efficiently optimizing hyperparameters remains a major challenge, as traditional techniques like grid search and random search are often time-consuming and inadequate in finding globally optimal solutions for complex models. These gaps motivate the need for a more robust and adaptable solution, such as the proposed EHGS-VGG16 framework.

However, the effectiveness of any CNN model is greatly influenced by factors like the number of convolutional layers and filters, as well as the filter size, batch size, learning rate, the choice of activation function, and more. These parameters are referred to as the CNN hyperparameters[29]. The values of the hyperparameters determine the success rate of a CNN in solving a particular problem. It is important to note that there is no single CNN architecture that can consistently produce satisfactory results for all problem instances. Particularly, finding the optimal values for hyperparameters in a specific problem is an NP-hard task and a major challenge in the field of CNN[1]. There are two popular methods for hyper-parameter optimization: grid search and random search[30]. The grid search method involves testing all possible hyperparameter combinations, which is efficient for a small set of hyperparameters but can become time-consuming with a larger number. Searching randomly is quicker and more efficient, as it tests various hyperparameter combinations randomly. Nevertheless, it fails to learn from previous attempts and might occasionally duplicate the same search[1]. In specific, conventional hyperparameter tuning techniques, such as grid search and random search, are frequently criticized for their extensive time consumption and substantial domain knowledge requirements. This calls for investigating more effective alternatives that can lessen these limitations. Since the optimization of the CNN hyperparameters falls within the category of NP hard problems, using metaheuristic-based techniques could yield satisfactory results.

Metaheuristics (MHs) are a class of stochastic algorithms that aim to find near-optimal solutions within a reasonable amount of time[31]. Although these methods do not always guarantee the optimal solution, they are particularly useful in dealing with intricate problems where an exhaustive search is impractical. The basic idea behind MHs is that they generate a random solution or a set of random solutions (known as a population) and then iteratively improve these solutions by applying a set of mathematical operators until a stopping criterion is satisfied. Population-based techniques are a popular type of MHs that have gained significant attention in recent years due to their successful performance[32]. The key advantage of population-based MHs is that they balance the exploration of the search space to find promising areas (global search) with the exploitation of the nearby regions around the solutions found during the exploration phase (local search). MHs are general-purpose algorithms that can be adapted to handle a wide range of optimization problems. Over the years, MHs have been extensively used in various domains, including scheduling problems, image processing, data mining, engineering design, and more[33]. According to Mirjalili and Lewis[34], population-based MHs can be classified into four main types according to their source of inspiration: physics-based, human-based, evolutionary-based, and swarm intelligence (SI) algorithms. Examples of these categories include the gravitational search algorithm (GSA), teaching-learning-based optimization (TLBO), genetic algorithms (GA), and particle swarm optimization (PSO).

Swarm intelligence (SI) algorithms are well-known for their effective performance in handling optimization problems. These algorithms take inspiration from the collective behavior of various species like bees, wasps, ants, fish, and birds[35,36]. The social behaviors of various species, particularly in activities like prey hunting, cooperative food-finding, and swarm leadership, serve as inspiration for SI algorithms. Novel nature-inspired MHs have recently emerged, integrating familiar natural processes into numerical models. The Hunger Games Search (HGS) is an SI-based algorithm that draws inspiration from the actions and choices made by animals in times of food scarcity[37]. HGS possesses several distinctive traits. It's easy to use and runs quickly[38]. This system includes adaptive and time-varying mechanisms, which help it effectively address challenges related to multi-modality and local optima. Moreover, HGS implements a "hunger ratio" that impacts the search range, enhancing adaptability and enabling adjustments according to the algorithm's performance. These characteristics contribute to HGS being a versatile and robust search algorithm. However, the standard HGS still has its limitations, such as restricted population diversity and a tendency to become trapped in local optima[38]. According to[39], these difficulties are particularly evident in multi-dimensional optimization problems. To address these constraints, researchers have enhanced the HGS by integrating new modified search strategies and pairing it with other optimization techniques to boost its performance[38,40]. As per the No Free Lunch (NFL) principle[41], no one optimization strategy can be universally the most effective for all optimization problems. Therefore, there is still room for additional research and analysis of incorporating effective MS-based methods.

This study aims to propose an optimized deep learning framework, called EHGS-VGG16, which integrates the enhanced Hunger Games Search (EHGS) algorithm with the VGG16 model for accurate and effective logo classification. The objective is to develop a robust and adaptable pipeline capable of identifying and classifying logos from diverse sources, even in challenging scenarios such as occlusion, varying orientations, and complex backgrounds.

To achieve this objective, the study seeks to answer the following research question: Can the integration of the EHGS algorithm with the VGG16 deep learning model improve logo classification performance, particularly under difficult conditions such as occlusion, size variability, and complex backgrounds?

The classification pipeline of the EHGS-VGG16 model involves multiple stages: preprocessing, object proposal generation via selective search, and hyperparameter tuning using the EHGS algorithm. We make use of CNNs, specifically the VGG-16 model, which is well-known for its profound capacity to recognize hierarchical structures and spatial properties in images. Through the use of pretrained weights and a deep architecture, our method improves the accuracy and resilience of the VGG-16 model. The EHGS-VGG16 model was validated using the Flickr-27 dataset and compared against state-of-the-art deep learning models, including ResNet50V2, InceptionV3, DenseNet121, EfficientNetB0, and MobileNetV2. The results demonstrate that EHGS-VGG16 achieved an impressive 98% accuracy in logo classification, surpassing previous models and setting a new benchmark in the field

- Introduction of an optimized deep learning framework for efficient and accurate logo classification using the VGG-16 model.
- Development of a comprehensive classification process that involves data preparation, object proposal via selective search, and optimizing key parameters.
- Implementation of an enhanced version of the HGS (EHGS) for hyperparameter optimization, combining the concepts of "local best" and "local escaping mechanism", to improve exploration and diversity in the search process.
- Validation of the proposed model's performance on the Flickr-27 dataset demonstrated superior accuracy and efficiency compared to other recent deep learning models and state-of-the-art methods, with a remarkable accuracy of 98%.The EHGS-VGG16 framework goes beyond state-of-the-art by addressing key challenges in logo classification, particularly robustness in varying conditions like occlusion and distortion, while efficiently optimizing hyperparameters. The proposed method leverages the advanced search capabilities of the EHGS algorithm. The EHGS improves both the exploration and exploitation phases of the search process, enabling the model to escape local optima and achieve superior classification accuracy in challenging scenarios.

The structure of the paper is organized as follows: "Review of related works" section explores recent research efforts on DL-based logo classification and the advancements in the HGS. The explanation of the techniques being employed is provided in "Research background" section. "Proposed methodology for enhanced logo detection" section offers an in-depth explanation of the proposed methodology. Section "Experimental results and simulations" section delves into the experimental results of the proposed approach. The work is summarized, and future research prospects are discussed in "Impact of EHGS on hyperparameter tuning" section.

Review of related works

This section examines research related to our study, organized into three main parts. "Advances in logo classification with deep learning" section explores advancements in DL-based logo classification, focusing on the difficulties associated with this area. "Metaheuristic-based optimization for deep learning and machine learning models" section investigates the use of MHs for fine-tuning hyperparameters in DL models and the recent hybrid approaches that combine MHs with machine learning across diverse applications. Finally, we explore innovative developments in HGS as a standalone contribution to optimization techniques in "Developed variants of the hunger games search" section.

Advances in logo classification with deep learning

Significant progress has been made in the field of logo classification with the use of DL methods, specifically CNNs. These developments have fundamentally changed computational models' capacity to identify and classify

logos, which is a challenging task because of the vast range of logo designs and the different contexts in which they can be found.

Bianco et al.[14] developed a novel logo recognition technique leveraging DL. Their method employs a two-step pipeline: first, a logo region is proposed, and then a specialized CNN classifies the logo, even in cases where logos are not precisely localized. To assess the effectiveness of this approach, they conducted experiments using the FlickrLogos-32 database. Their study examined the influence of different data augmentation strategies (synthetic vs. real) and image preprocessing techniques on recognition accuracy. Furthermore, they systematically evaluated various training choices, including class balancing, sample weighting, and explicitly modeling non-logo areas (background class). The experimental results of the study prove that the suggested method works well and is better than other state-of-the art methods. The authors' work contributes to advancing logo recognition by leveraging DL techniques and optimizing training choices to enhance performance. Another notable contribution is proposed by Oliveira et al.[42]. In their study, an automatic system for robust logo detection under unconstrained imaging conditions was introduced. The authors utilized Fast Region-based CNN (FRCN) and pretrained CNN models from the ILSVRC ImageNet dataset. They incorporated a selective search for window proposals and employed data augmentation techniques to enhance the logo recognition rate. The novel use of transfer learning and the ability to detect multiple logo instances were key features of their framework. Experimental results on the FlickrLogos-32 dataset demonstrated the promising performance of their models, surpassing state-of-the-art systems with handcrafted-based models.

Sahel et al.[43] employed emerging DL techniques to tackle brand and logo recognition tasks by leveraging multiple modern CNN models. In their research, they used object detection models that had already been trained to make logo detection tasks easier, especially when they only had access to a few labeled training images that had been taken in real life. This way, they did not have to do a lot of work to classify the images by hand. The study utilized the widely used FlickrLogos-32 dataset, which contains real-world product images and serves as a common public dataset for logo detection and brand recognition. The evaluation of the models focused on both their efficiency in model creation and their detection accuracy. Experimental results demonstrated superior logo-detection outcomes.

To overcome the challenge of limited data, Su et al.[44] proposed the Synthetic Context Logo (SCL) technique for training image generation. This method aims to augment limited labeled data and improve the performance of deep learning models for logo recognition. When evaluated against the faster R-CNN model, the SCL method demonstrated superior results. The researchers also introduced TopLogo-10, a novel dataset featuring the 10 most prevalent logos from the clothing industry, which were captured in complex visual environments. This contribution advances the field of logo detection and provides a valuable resource for the future.

Table 1 provide an overview of recent studies in the field of logo classification.

Metaheuristic-based optimization for deep learning and machine learning models

The literature has extensively focused on optimizing the hyperparameters of neural networks[11–24,29,30]. However, the definitions of hyperparameters vary across these works, leading to different optimization objectives. Some studies adopt a narrow definition of hyperparameters, limiting them to parameters within each neural network layer and aiming to fine-tune existing neural networks without altering their overall architecture. In contrast, other studies take a broader approach, encompassing factors like the number and order of layers and more to create entirely new neural network architectures. This paper specifically focuses on the former approach. Fine-tuning occurs after the neural network's architecture is determined or when it is applied to a new dataset. For

References	Year	Method	Dataset	Performance
45	2019	YOLOv3	VLD-3	76.6–98.8
46	2017	Faster R-CNN	FlickerLogo	0.52–0.67
47	2019	OSLD - SDML	BLAC and Flickr-32	84.5–90.9
48	2018	RetinaNet	INbreast - GURO	86–1.00
49	2017	Faster R-CNN	FlickrLogos	mAP 51–66%
50	2020	Faster R-CNN and YOLOv2	WebLogo-2M	mAP 36.8–46.9
51	2017	DenseNet16 -, ResNet101 -, VGG16	FlickrLogos-32 SportsLogos	0.37–0.46
52	2017	Scalable Logo Self Training (SLST)	WebLogo-2M	mAP 34.37%
53	2007	ANN and Support Vector Machine (SVM), Fisher classifier	Tobacco-800	39.3–84.2%
54	2015	RCNN - FRCN - SPPnet	Logos-18 test set Logos-160 test set	81.3–95.2%
5	2019	Faster R-CNN, SSD, YOLOv3	FlickrLogos-32, PL2K	0.565 mAP
14	2017	CNN	FlickrLogos-32 Logos-32plus	0.1–95.8%
42	2016	Fast Region-based Convolutional Networks (FRCN)	ILSVRC, FlickrLogos-32	mAP 54.5–73.74%
55	2018	Retina U-Net, Mask R-CNN, Faster R-CNN +, U-Faster RCNN, + DetU-Net	LIDC-IDRI	mAP 29–50.5%
44	2017	Faster R-CNN	FlickrLogo-32 TopLogo-10	mAP 20.5–81.1%
12	2020	DenseNet-CNN	FlickrLogo-32	92.80%

Table 1. Overview of recent studies on logo classification, summarizing authors, publication year, methods, datasets, and key performance outcomes.

CNNs, relevant hyperparameters include those for convolutional layers (kernel size, number, stride, padding), pooling layers (method, stride, padding), and fully connected layers (number of kernels)[29].

According to the literature, both evolutionary and swarm intelligence algorithms have numerous applications and implementations in the field of CNNs and computer vision. These methods were investigated and validated using a range of benchmark and real-world datasets. A group of researchers implemented fundamental metaheuristic algorithms and achieved promising outcomes. For example, The authors in[56] proposed a hybrid model combining deep learning and metaheuristics for classifying strokes using EEG signals. They developed a CNN-BiGRU model for effective time series analysis and employed the Harmony Search (HS) algorithm for feature selection and Multiverse Optimization (MVO) for hyperparameter tuning. This approach addresses the limitations of CT and MRI in stroke diagnosis by offering a faster, cost-effective alternative. The model achieved a 99.991% prediction accuracy and demonstrated significant improvement over previous methods. Additionally, a cloud-based decision support system was developed for real-time diagnosis, providing timely and accessible results via SMS notifications. Another study proposed by Sawan et al.[57] implemented three metaheuristic approaches-simulated annealing, differential evolution, and harmony search-to optimize CNNs for improved accuracy. The authors focused on enhancing CNN performance for classifying the MNIST and CIFAR datasets. Despite increased computation time, the metaheuristic-optimized CNNs demonstrated improved accuracy, with gains up to 7.14% compared to the original CNNs. This study addressed the limited exploration of metaheuristic strategies in optimizing CNNs and highlighted their potential for accuracy enhancement in deep learning tasks. Yamasaki et al.[58] presented methods to automatically find optimal parameter settings for CNNs) using the evolutionary algorithm called Particle Swarm Optimization (PSO). Despite the vast parameter space ($> 10^20$), the authors experimentally demonstrated that better parameter settings could be found for the AlexNet configuration across five different image datasets. They also developed two candidate pruning algorithms to enhance the efficiency of the evolutionary process. Their experiments achieved 0.7–5.7% improvements over the original parameter sets in Caffe, while requiring only 2–4% of the processing cost compared to the naive PSO-based approach. Another novel PSO-based optimized CNN approach for image classification is proposed by Sinha et al.[59]. Given the challenge of parameter selection, the authors applied PSO to determine the best parameter combinations. They evaluated their approach on the CIFAR-10 benchmark dataset and a real-world roadside vegetation dataset. The results showed that their method efficiently explored the solution space and identified optimal parameters.

Recent advancements in deep learning have significantly impacted the diagnostic capabilities for COVID-19, particularly through the analysis of chest X-ray images. In this context, Hu et al.[60] proposed a novel two-phase deep learning approach for real-time COVID-19 detection from chest X-ray images. The authors leveraged a deep CNN as a feature extractor and an Extreme Learning Machine (ELM) for classification. To enhance the stability and reliability of the ELM, the Chimp Optimization Algorithm (ChOA) was employed to optimize the input weights and biases. The effectiveness of the proposed approach was validated using the COVID-Xray-5k and COVIDetectioNet datasets, yielding accuracy rates of 98.25% and 99.11%, respectively. Notably, the training and testing times were significantly reduced, which supports the feasibility of real-time applications in clinical settings. Similarly, Cai et al.[61] proposed a novel deep CNN model called DCNN-ChOA that utilized the ChOA to fine-tune the fully connected layers of the DCNN for COVID-19 diagnosis from chest X-ray images. Tested on COVID-Xray-5k and COVIDetectioNet, this model achieved 99.11% accuracy and demonstrated reduced false positive and negative rates. Additionally, Class Activation Mapping (CAM) was employed to highlight COVID-19-infected regions in X-rays to boost interpretability. In a related study, Wang et al.[62] employed a similar methodology but utilized the WOA to train the DCNN's fully connected layers. This approach, when tested on the COVID-Xray-5k dataset, achieved an accuracy of 99.06%. In another notable study, Khishe et al.[63] developed an adaptive CNN framework named OptiDCNN for early detection of COVID-19 from chest X-ray images. This model was intended to maximize diagnostic accuracy while maintaining a manageable model size suitable for real-time applications. OptiDCNN evolves the CNN architecture iteratively, optimizing hyperparameters and layer configurations through a nature-inspired optimization technique called biogeography-based optimization (BBO). The model begins with a basic CNN structure, expanding depth and complexity until optimal accuracy is achieved. This adaptive, phased approach leverages BBO to tune CNN hyperparameters like filter size, activation type, and pooling method at each stage. Using the COVID-Xray-5k dataset, OptiDCNN achieved a 99.11% accuracy rate, demonstrating high diagnostic performance while maintaining a compact network structure, thus supporting fast and reliable real-time applications in clinical settings.

Based on the reviewed studies, we can see that the direct application of basic metaheuristics has significantly improved the performance of CNN-based networks by finding the optimal set of parameters. However, basic metaheuristics often suffer from premature convergence and may fail with large, complex, high-dimensional problems, such as those involving very large neural networks. To address this issue, researchers have attempted to enhance the exploration and exploitation behaviors of basic metaheuristics by either modifying their mechanisms or hybridizing them with other techniques.

Some researchers have suggested hybrid or adaptive approaches for optimizing CNN. Leung et al.[64] proposed a novel PSO model for radial basis function neural networks (RBFNNs) to address classification problems. The model used a linearly decreased inertia weight for each particle (ALPSO), automatically calculated based on the fitness value. The ALPSO algorithm was compared with various well-known PSO algorithms on benchmark test functions. Additionally, the orthogonal least squares algorithm (OLSA) combined with ALPSO was used to further optimize the RBFNN structure, including weights and controlling parameters. The integrated optimization model (MOA-RBFNN) was validated on various benchmark classification problems, demonstrating superior performance compared to conventional methods and recent approaches. Another interesting study in[65] addressed the challenge of plant disease classification using digital images. The authors developed an ensemble model of two pre-trained CNNs, VGG16 and VGG19, to diagnose plant diseases by classifying images of healthy

and unhealthy leaves. To overcome the difficulty of manually identifying optimal hyperparameters for CNNs, the orthogonal learning PSO (OLPSO) algorithm was used. Additionally, an exponentially decaying learning rate (EDLR) schema was applied to prevent CNNs from falling into local minima and to train efficiently. The study also addressed dataset imbalance using random minority oversampling and random majority undersampling methods. The results showed that the proposed model achieved competitive accuracy, outperforming other pre-trained CNN models like InceptionV3 and Xception, whose hyperparameters were selected using non-evolutionary methods.

Another state-of-the-art method was presented in[29]. This study proposed a novel PSO variant, cPSO-CNN, for optimizing the hyperparameter configuration of architecture-determined CNNs. The cPSO-CNN utilized a confidence function defined by a compound normal distribution to model experts' knowledge on CNN hyperparameter fine-tuning, enhancing the exploration capability of the canonical PSO. Additionally, the scalar acceleration coefficients of PSO were redefined as vectors to better adapt to the varying ranges of CNN hyperparameters. A linear prediction model was also adopted for fast ranking of the PSO particles, reducing the cost of fitness function evaluation. Experimental results demonstrated that cPSO-CNN performed competitively compared to several reported algorithms in terms of both CNN hyperparameter optimization and overall computation cost. Recently, the authors in[66] introduced the Adaptive Habitat Biogeography-Based Optimizer (AHBBO) to tune hyperparameters of Deep CNNs (DCNNs) for image classification tasks. Tested on 53 benchmark optimization functions, AHBBO demonstrated improved performance and faster convergence. When applied to DCNNs (DCNN-AHBBO), it outperformed 23 well-known image classifiers on nine challenging classification tasks, reducing error rates by up to 5.14%. This algorithm outperformed 13 benchmark classifiers in 87 out of 95 evaluations, offering a high-performance and reliable solution for optimizing DCNNs in image classification.

In efforts to advance automated COVID-19 detection, Saffari et al.[67] proposed a DCNN-FuzzyWOA model utilizing a deep CNN with the WOA enhanced by fuzzy logic to optimize training. The key innovation is the use of a fuzzy system to tune the parameters of the WOA used to train the DCNN, which allows for better balancing of the exploration and exploitation phases in the optimization. FuzzyWOA achieved a 100% accuracy rate with a processing time of 880.44 s on the COVID-Xray-5k dataset. It proved better performance compared to other DCNN training methods like DCNN-PSO, DCNN-GA, and LeNet-5.

In the context of real-time sonar image recognition, Yutong et al.[68] proposed a study similar to Hu et al.[60] work but employed a fuzzy map-enhanced Slime Mould Algorithm (FSMA) instead. The fuzzy mapping are used to balance exploration and exploitation when tuning the weights and biases of an ELM classifier. This modification targeted the efficient detection of underwater anomalies, achieving a 2.11% improvement over the best benchmark models. Khishe et al.[69] proposed a deep neural network fine-tuned using the Moth Flame Optimization (MFO) algorithm. Their approach customized MFO with different spiral motions to improve classification accuracy, addressing challenges like premature convergence and local minima entrapment. The modified model was validated on three sonar datasets, including the Sejnowski & Gorman, passive sonar, and active sonar datasets. The results confirmed a classification improvement of 1.6 to 2.1% over standard algorithms like ChOA, PSO,ALO, and HBO. The study underscores the value of integrating customized MFO with DCNNs for high-dimensional, real-time sonar classification tasks. To improve classification accuracy and address the challenges of fluctuating underwater sound propagation and ambient noise, Khishe et al.[70] proposed a Variable-Length Habitat Biogeography-Based Optimizer (VLHBBO) for tuning a DCNN in underwater sonar wave recognition tasks. This approach allows the model depth-and consequently the hyperparameter space-to adapt dynamically based on environmental conditions. Tested on experimental sonar data collected in the Persian Gulf and the Oman Sea, as well as benchmark datasets from the New Array Technology III program, the VLBBO-DCNN achieved superior classification performance across eight evaluation metrics.

To enhance the classification accuracy and adaptability of deep learning models, Khishe[71] developed a variable-length DCNN optimized by the WOA algorithm for complex image classification tasks across diverse contexts. This innovative approach introduced an Internet Protocol Address (IPA)-based encoding technique, allowing WOA to dynamically adjust the DCNN layer configurations to fit varying classification challenges. Tested on nine standard datasets, the IPA-based WOA (IPWOA) method outperformed twenty-three other classifiers in 81 out of 95 evaluations, achieving efficient network tuning with higher classification accuracy. In the context of electronic intelligence (ELINT) and electronic support measure (ESM) systems, Azhdari et al.[72] addressed the complexities of Pulse Repetition Interval Modulation (PRIM) recognition by developing a four-phase model (DCNN-VBBO-ELM) to handle noisy PRI patterns. This approach utilizes a deep convolutional neural network (DCNN) for feature extraction, extreme learning machines (ELMs) for real-time pattern recognition, and a variable-length biogeography-based optimizer (VBBO) to optimize the network's parameters. Tested on a custom dataset of five PRI patterns, the model achieved a 97.05% accuracy, outperforming traditional ELM-based models with a swift training time of 27 s for 50,000 images. In another interesting study, Bacanin et al.[73] proposed a chaotic firefly algorithm with enhanced exploration to address global optimization problems by focusing on dropout regularization in DCNN to mitigate the overfitting problem. This novel approach was validated on benchmark datasets, including MNIST and CIFAR-10, and showed significant improvements over traditional methods. Table 2 provides a summary of reviewed studies on Metaheuristic-Based optimization for DCNN. For more metaheuristic-based optimization approaches for CNN parameters, please refer to this study[74].

The novel research field successfully combines machine learning and swarm intelligence approaches and proved to be able to obtain outstanding results in different areas. For optimizing feature selection in complex image recognition tasks, Malakar et al.[75] introduced a GA-based hierarchical feature selection (HFS) model to enhance handwritten word recognition. Their approach employs GA to reduce feature dimensionality to improve model accuracy and computational efficiency. The HFS model was tested on an in-house dataset of 12,000 Bangla word samples and achieved a 1.28% increase in accuracy with a 28% reduction in feature

References	Objective	Model/approach	Employed metaheuristic	Results
Leung et al.[64]	RBFNN optimization for classification	ALPSO with OLSA	Adaptive Linear PSO (ALPSO)	Superior performance on benchmark classifications
Rere et al.[57]	Improve CNN accuracy on MNIST and CIFAR	Metaheuristic-optimized CNN	SA, DE, HS	Accuracy improvement up to 7.14%
Yamasaki et al.[58]	Optimize CNN parameters for image classification	AlexNet with PSO	PSO	0.7–5.7% improvement over Caffe's original setup
Sinha et al.[59]	Image classification on CIFAR-10 and roadside vegetation	PSO-based optimized CNN	PSO	Efficient parameter exploration
Darwish et al.[65]	Plant disease classification	Ensemble CNNs (VGG16, VGG19)	Orthogonal Learning PSO (OLPSO)	Outperformed InceptionV3, Xception
Wang et al.[29]	Optimize CNN hyperparameters for efficiency	cPSO-CNN	Confidence-based PSO (cPSO)	Competitive performance, lower computational cost
Hu et al.[60]	Real-time COVID-19 detection from chest X-rays	CNN-ELM with ChOA	ChOA	98.25–99.11% accuracy on COVID-Xray-5k
Wang et al. [62]	COVID-19 detection from chest X-rays	DCNN-WOA	WOA	99.06% accuracy
Yutong et al.[68]	Real-time sonar anomaly detection	FSMA-ELM	Fuzzy SMA (FSMA)	2.11% improvement over best benchmark models
Khishe et al.[63]	Early COVID-19 detection with compact model	OptiDCNN	BBO	99.11% accuracy, supports real-time applications
Saffari et al.[67]	COVID-19 detection with enhanced WOA	DCNN-FuzzyWOA	Fuzzy WOA	100% accuracy on COVID-Xray-5k
Khishe et al. [70]	Sonar wave recognition under varying conditions	VLBBO-DCNN	Variable-Length Habitat BBO (VLHBBO)	Superior across eight metrics on sonar datasets
Cai et al.[61]	COVID-19 diagnosis from chest X-rays	DCNN-ChOA	ChOA	99.11% accuracy
Khishe et al.[69]	High-dimensional sonar classification	DNN-MFO	MFO	1.6–2.1% classification improvement
Khishe[71]	Adaptable CNN for complex image classification	Variable-Length WOA (IPWOA)	Internet Protocol-based WOA	Outperformed 23 classifiers in 81 of 95 evaluations
Azhdari et al.[73]	Pulse Repetition Interval Modulation (PRIM) recognition	DCNN-VBBO-ELM	Variable-Length BBO	97.05% accuracy, fast real-time training
Sawan et al.[56]	Classify strokes using EEG signals	CNN-BiGRU with HS & MVO	HS, Multiverse Optimization (MVO)	99.991% accuracy, real-time cloud-based system
Xin et al.[66]	DCNN hyperparameter tuning	DCNN-AHBBO	Adaptive Habitat BBO	Outperformed 23 classifiers, lower error rates

Table 2. Overview of metaheuristic-based optimization approaches for Deep Convolutional Neural Networks (DCNN) Based on objective, approach, employed metaheuristic, and main findings.

dimension. Zivkovic et al.[76] developed a novel hybrid metaheuristic approach that combines an improved firefly algorithm with XGBoost to optimize hyperparameters in network intrusion detection systems (NIDS). The authors enhanced the firefly algorithm with additional exploration mechanisms and hybridized it with the sine-cosine algorithm for more robust tuning. Experimental results on the NSL-KDD and UNSW-NB15 datasets demonstrated improved detection accuracy and precision. Additionally, Zivkovic et al.[77] explored the fusion of simple CNN with ELM to tackle IoT security issues. They applied a modified sine cosine algorithm (SCA) for tuning ELM hyperparameters and weights, using the SHapley Additive exPlanations (SHAP) method to interpret the model's outcomes for security analysis. The experimental results demonstrated improved classification accuracy and robustness in IoT intrusion detection outcomes. Recently, an interesting study by Jovanovic et al.[78] focused on advancing IoT security by employing a novel metaheuristic optimization method for feature selection, hyperparameter tuning, and training of ELMs. This approach addresses key hyperparameter optimization (HPO) tasks, specifically by adjusting the count of neural cells in the ELM's intermediate layer and optimizing weight and bias initialization. The method was tested on a Windows 10 security dataset from the TONIot base, showcasing improvements in IoT security applications. These studies collectively highlight the advantages of combining swarm intelligence with machine learning to address complex optimization challenges, such as cybersecurity and IoT security applications.

In conclusion, the reviewed studies demonstrate the success of several metaheuristic techniques-including PSO, GA, WOA, SA, HS, SMA, BBO, MVO, and ChOA-when employed to optimize a DCNN model. These techniques have been effective in improving classification accuracy in a variety of applications, including medical diagnosis and image classification. However, according to the "no free lunch" theorem, no optimization technique performs better than another in every task. In specialized tasks such as logo classification, where distinctive elements are more complex and highly variable in size, design, and orientation, many existing algorithms struggle to generalize well.

Our study aims to address these gaps by using an improved HGS algorithm to optimize the VGG16 model specifically for logo classification, which, to our knowledge has not been applied in this domain before. By incorporating advanced search strategies into HGS, our approach aims to improve both the exploration and exploitation phases and better adapt to the complex feature spaces characteristic of logo data. This novel application of HGS to logo classification not only advances optimization strategies within deep learning, but also provides a specialized solution designed for the complex challenges of logo recognition tasks.

Developed variants of the hunger games search

The HGS approach that inspires its idea from the "Hunger Games" is notable in the optimization domain for its high robustness, few parameters, and simplicity compared to other optimization techniques. These traits are precisely the reason why HGS is useful and frequently applied for resolving challenging issues in various applications. For instance, Fahim et al.[79] developed an HGS-based model for optimizing proton exchange membrane fuel cell (PEMFC) parameters. The proposed HGS-based PEMFC model was applied to two commercial PEMFCs: the Ballard Mark V and the BCS 500 W. With the suggested HGS-based PEMFC model, a very accurate model could be achieved, which would change how the fuel cells work and are controlled in the simulations (2021). A different study by Nguyen and Bui[80] proposed a soft computing model, referred to as HGS-ANN, that combines HGS with an artificial neural network (ANN) for forecasting the intensity of ground vibrations (BIGV) caused by mine blasting. Experimental results using real data demonstrated that the HGS-ANN model performed better compared to other MH-based models; thus, it is recommended for optimizing blast patterns and reducing environmental effects.

To achieve a 96-h long-term hybrid microgrid (HMG) system scheduling energy management scheme, Shaker et al.[81] proposed a multi-objective optimizer based on the hunger game search (HGSO). The proposed strategy focuses on maintaining uninterrupted power with low operating costs and minimal emissions from the storage system, all while achieving a high renewable factor. When comparing the proposed approach with other advanced optimizers, it demonstrates consistent success in decreasing power loss, reducing emissions, and providing cost savings for customers. AbuShanab et al.[82] developed an AI-based predictive model for manufacturing processes, specifically focusing on friction stir welding of different types of plastic materials. Their model employs a technique called Random Vector Functional Link (RVFL), which is enhanced by the Hunger Games Search (HGS) optimizer to improve accuracy by optimizing the model's settings. They compared their RVFL-HGS model against two other models optimized with different methods and found that RVFL-HGS outperformed the others across several statistical measures. This demonstrates its effectiveness in predicting the qualities of welded joints.

In addition to the above-related works that review the application of the basic HGS in various practical problems, some research efforts aim to overcome the limitations of the basic HGS. These efforts involve incorporating new operators and hybridizing with other MHs to enhance performance and solve more complex problems. For instance, Chakraborty et al.[83] proposed an enhanced HGS that combines the starvation concept of HGS with the search methods of the Whale Optimization Algorithm (WOA), termed HSWOA. This integration aims to balance the exploration and exploitation phases effectively. The HSWOA was evaluated using 30 classical benchmark functions and seven real-world engineering challenges. Kutlu Onay and Aydemir[84] utilized the features of chaotic mappings, ergodic and irreducible, to integrate into HGS to address its limited search capability. The chaotic HGS performance is showcased on 23 classical functions and CEC2017, and this version is utilized in three practical engineering application problems. In their study, Li et al.[85] a hybrid algorithm based on HGS and DE called DEHGS. This method aims to enhance the utilization of local regions within individual neighborhoods. In addition, a novel ranking-based method is added to DE to produce more promising results. Experimental results on the IEEE CEC2017 benchmark functions demonstrate the hybrid algorithm's effectiveness compared to some state-of-the-art algorithms.

As reported in the study by Mahajan et al.[86], the convergence rate can be significantly improved by implementing the AOAHGS technique which combines the arithmetic optimization algorithm (AOA) and HGS. This method was tested with 23 classical functions and exhibited an optimal balance between the exploration and exploitation phases of the original HGS. Ma et al.[87] developed a multistage HGS (MS-HGS) and its binary counterpart (MS-bHGS) as two new methods. The proposed algorithm uses chaos theory, greedy selection, and vertical crossover to attain the optimum balance of exploration and exploitation. The new MS-HGS algorithm is assessed on 23 real-valued benchmark functions, while MS-bHGS is implemented to reduce the dimensionality of 20 datasets from the UCI repository. The experimental results demonstrate that MS-HGS is an advanced optimizer, while MS-bHGS can be regarded as a very effective wrapper feature selection technique.

Xu et al.[88] developed an advanced HGS (IHGS) to handle solar photovoltaic (PV) parameter extraction. The authors employed a combination of a quantum revolving gate strategy and Nelder-Mead simplex methods (NMs) to maintain the population diversity and accelerate the search towards the best solutions in the decision space. Experimental results prove that the IHGS algorithm is a valuable optimization technique for the estimation of solar PV parameters. In another notable contribution, Zhou et al.[38] introduced an enhanced variant of the HGS (OCBHGS)and. It incorporates three primary strategies: the Gaussian barebones scheme, chaotic initialization, and orthogonal training. The goal is to maximize the initial population's quality, increase population diversity, and extend domain explorations to improve solution quality. Experimental evaluation was conducted using the CEC2014 competition benchmark function. Additionally, OCBHGS was applied to resolve three constrained real-world engineering challenges. The findings demonstrate that the OCBHGS significantly outperforms in terms of convergence speed and accuracy.

Table 3 summarizes the reviewed studies on HGS and its enhancements. The examined studies demonstrate notable progress in applying HGS for optimization in a variety of domains. However, there is still a need to address the problem of effectively escaping local optima and utilizing local optimal solutions. To significantly boost optimization efficiency, our proposed work provides an enhanced HGS that creatively combines the idea of local best solutions with a mechanism for escaping local optima. Furthermore, this work is the first use of the HGS specifically to fine-tune the VGG16 model for logo classification. This new methodology not only adds to the body of knowledge about optimization techniques but also pioneers the use of advanced HGS variants in the field of deep learning for image recognition.

References	Year	Contribution/Enhancement	Application	Remarks
[79]	2021	HGS-based model for optimizing PEMFC parameters	PEMFC: Ballard Mark V and BCS 500 W	Demonstrated high accuracy, potentially changing fuel cell simulations
[80]	2021	HGS-ANN for forecasting BIGV caused by mine blasting	Forecasting ground vibrations	Outperformed other MH-based models, recommended for blast optimization
[81]	2021	Multi-objective HGSO for hybrid microgrid system scheduling	Energy management in HMG systems	Consistently decreased power loss, emissions, and cost
[82]	2021	RVFL enhanced by HGS for manufacturing processes	Friction stir welding of plastics	Outperformed other models in accuracy
[83]	2021	Enhanced HGS with WOA (HSWOA)	Classical benchmark functions and engineering challenges	Balanced exploration and exploitation phases
[84]	2022	Chaotic mappings integrated into HGS	Classical functions and CEC2017	Addressed HGS's limited search capability
[85]	2021	Hybrid HGS and DE algorithm (DEHGS)	IEEE CEC2017 benchmark functions	Enhanced local region utilization and provided promising results
[86]	2022	AOAHGS method combining AOA and HGS	Classical functions	Achieved faster convergence, balanced exploration and local search
[87]	2022	Multi-strategy HGS (MS-HGS) and its binary variant (MS-bHGS)	Benchmark functions and UCI feature selection datasets	Superior optimizer, valuable for feature selection
[88]	2022	Improved HGS (IHGS) with quantum and Nelder-Mead methods	Solar PV parameter estimation	Showed valuable optimization for solar PV parameter estimation
[38]	2022	Enhanced HGS (OCBHGS) with three primary strategies	CEC2014 benchmark and real-world engineering challenges	Significantly outperforms in convergence speed and accuracy

Table 3. Summary of reviewed studies on Hunger Games Search (HGS) and its enhancements, highlighting main contributions, application domains, and key remarks.

Research background

This section provides the comprehensive background information required for a self-explanatory understanding of the techniques employed in our research. The fundamental HGS, as introduced by Yang et al.[37], is detailed in "Hunger games search (HGS): inspiration and mathematical model" section. Following this, "VGG16 deep learning model" section explores the VGG16 deep learning model. The selective search approach for object proposals, along with the Intersection over Union (IoU) measure, are presented in "Selective search for object proposal" section.

Hunger games search (HGS): inspiration and mathematical model

Inspiration and motivation behind HGS

Animals make decisions and behave based on computational principles that come from their sensory input and interactions with their surroundings. The cognitive structures of animals are based on principles that develop to improve survival, reproduction, and food acquisition[89]. Hunger significantly influences animals' behaviors, emphasizing the importance of seeking food over other inputs and requirements. Therefore, animals must skillfully manage exploration, defense, and competition, demonstrating adaptable feeding techniques[90].

Animal behavioral decisions are impacted by a combination of factors such as their motivational state and environmental cues. Neuroscientists acknowledge hunger as a major motivator that takes precedence over other drives including thirst, fear, and social demands[91]. It drives animals towards behaviors that secure their existence by maintaining stable life situations. Social behaviors are important for avoiding predators and locating food, but they can be less important when there is an immediate need to eat due to hunger. This natural selection process, driven by the ability to secure food, suggests a "hunger game" in the wild, where survival hinges on making the right decisions. Animals communicate about food sources to reduce uncertainty and enhance their chances of survival. In scenarios where food is scarce, animals engage in a strategic competition to access resources, with survival depending on their ability to make logical choices and movements. Building upon these premises, the Hunger Games Search (HGS) algorithms was proposed[37].

The Hunger Games Search (HGS) is a novel and recent population-based metaheuristic algorithm that mimics the natural instinct of animals to search for food[37]. Hunger is employed as the main motivation in order to design a competitive and computationally effective algorithm. The algorithm simulates competition, selection, and adaptive processes seen in nature as well as a game (Hunger Games) that is fictional. In such a game, agents (individuals) compete for resources or survival in a difficult environment. In the context of optimization, these agents are seen as possible solutions to a problem, and the environment is the area where the problem is being searched for solutions. The algorithm goes through a process of competition, selection, and adaptation that helps in generating the best solution(s) to the problem.

Mathematical model of the HGS algorithm

This section explains the mathematical model of the HGS algorithm. The algorithm relies on two fundamental components, the "Approach Rule" and the "Hunger Rule," to simulate natural hunger-driven behaviors and the adaptive decision-making strategies.

Approavh food

In nature, animals frequently collaborate in searching for food, while at other times they choose to forge independently[92]. Animal hunting techniques serve as inspiration for the mathematical formulas in Eq. (1). They

represent three different patterns in which animals move, reflecting their behavior when approaching a food source. These patterns are fundamental to the HGS algorithm, which mimics both cooperative behavior between animals and individual foraging.

$$\vec{X}(t+1) = \begin{cases} \vec{X}(t) \cdot (1 + \mathrm{randn}(1)), & \text{if } r_1 < l \\ \vec{W}_1 \cdot \vec{X}_b + \vec{R} \cdot \vec{W}_2 \cdot |\vec{X}_b - \vec{X}(t)|, & \text{if } r_1 > l, r_2 > E \\ \vec{W}_1 \cdot \vec{X}_b - \vec{R} \cdot \vec{W}_2 \cdot |\vec{X}_b - \vec{X}(t)|, & \text{if } r_1 > l, r_2 \leq E \end{cases} \tag{1}$$

where $\vec{R}$ varies within $[-a, a]$, r_1 and r_2 are random numbers within the [0, 1] range. The term $\mathrm{randn}(1)$ refers to a normally distributed random number. The variable t represents the current iteration, with $\vec{X}(t)$ and $\vec{X}(t+1)$ indicating the current and subsequent positions of each individual, respectively. $\vec{W}_1$ and $\vec{W}_2$ are weights associated with hunger, which are designed based on hunger-driven signals. $\vec{X}_b$ denotes the position of the best individual in the current iteration. Lastly, l is a significant control parameter of the HGS, which can influence its overall performance.

In the proposed updating rules, $\vec{X}(t) \cdot (1 + \mathrm{randn}(1))$ illustrates how an agent searches for food both hungrily and randomly at its current location. The expression $|\vec{X}_b - \vec{X}(t)|$ depicts the activity range of an individual at the current time, which is then adjusted by $\vec{W}_2$ to reflect hunger's impact on this activity range. The searching ceases once the individual is no longer hungry, with $\vec{R}$ serving as a control to gradually reduce the activity range to zero. The addition or subtraction of the activity range, influenced by $\vec{W}_1 \cdot \vec{X}_b$, mimics how an individual, guided by its peers to a food source, resumes searching at the current location after food is found. Here, $\vec{W}_1$ represents the discrepancy in accurately pinpointing the actual location in reality.

According to the mathematical formula in Eq. (2) E serves as a variation control for all positions.

$$E = \mathrm{sech}(|F(i) - BF|) \tag{2}$$

For each individual i, where i ranges from 1 to n with n being the number of search agents or population size, $F(i)$ denotes the fitness value. BF signifies the best fitness achieved during the current iteration (so far). The hyperbolic function sech is defined as:

$$\mathrm{sech}(x) = \frac{2}{e^x + e^{-x}}. \tag{3}$$

The expression for $\vec{R}$ is outlined as:

$$\vec{R} = 2 \times \mathrm{shrink} \times \mathrm{rand} - \mathrm{shrink} \tag{4}$$

$$\mathrm{shrink} = 2 \times \left(1 - \frac{t}{T}\right), \tag{5}$$

Here, rand represents a random value within the interval [0, 1], and T denotes the total number of iterations. The shrink parameter, calculated based on the current iteration t relative to the total iterations T, ranges from 0 to 2. This range reflects how the influence of the shrink factor diminishes over time, from its maximum at the start of the search process to zero as t approaches T. Consequently, the range of $\vec{R}$, which adjusts the activity range of the search agents based on shrink and rand, also varies from -2 to 2, as demonstrated in Fig. 1. This dynamic range allows for a controlled exploration of the search space, narrowing as the algorithm progresses, to focus on the exploitation of the best solutions found.

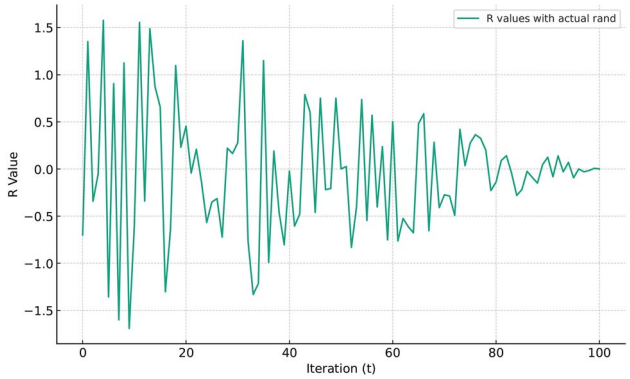


Fig. 1. Dynamic range of $\vec{R}$ in HGS over 100 iterations, demonstrating controlled exploration and focused exploitation.

Hunger rule

This section introduces the mathematical model designed to simulate the starvation traits of individuals, which form the foundational element of the HGS algorithm. The formulas for $\vec{W}_1(i)$ and $\vec{W}_2(i)$ in Eq. (1) are described in Eqs. (6) and (7), respectively.

$$\vec{W}_1(i) = \begin{cases} \text{hungry}(i) \cdot \frac{N}{\text{SHungry}} \times r_4, & \text{if } r_3 < l \\ 1, & \text{if } r_3 > l \end{cases} \tag{6}$$

$$\vec{W}_2(i) = (1 - \exp(-|\text{hungry}(i) - \text{SHungry}|)) \times r_5 \times 2 \tag{7}$$

In this context, hungry quantifies the hunger level of each individual; N denotes the total number of individuals; and *SHungry* represents the aggregate hunger across all individuals, essentially the sum of their hunger levels, that is, $\sum(\text{hungry})$. Random numbers r_3, r_4, and r_5 fall within the range of [0, 1]. The equation for calculating an individual's hunger, $\text{hungry}(i)$, is outlined in Eq. (8).

$$\text{hungry}(i) = \begin{cases} 0, & \text{if AllFitness}(i) = BF \\ \text{hungry}(i) + H, & \text{if AllFitness}(i) \neq BF \end{cases} \tag{8}$$

where $\text{AllFitness}(i)$ stores the fitness value of each individual for the current iteration. At each iteration, the hunger level of the best-performing individual is reset to 0. For the rest of the individuals, their hunger level (H) is updated by adding a certain amount to their existing hunger. This implies that the updated hunger levels, denoted by H, will vary among individuals. The value of H can be calculated using Eq. (9).

$$H = \begin{cases} LH \times (1 + r), & \text{if } TH < LH \\ TH, & \text{if } TH \geq LH \end{cases} \tag{9}$$

$$TH = \frac{F(i) - BF}{WF - BF} \times r_6 \times 2 \times (UB - LB) \tag{10}$$

where r_6 is a random number within the [0,1] range; $F(i)$ denotes the fitness value of each individual; BF represents the highest fitness achieved during the current iteration; WF signifies the lowest fitness obtained in the current iteration; UB and LB refer to the upper and lower bounds of the feature space, respectively. The sensation of hunger, as described in[93], denoted as H, is constrained by a minimum value, LH. To enhance the algorithm's efficiency, we manage hunger's upper and lower thresholds, with LH's value being explored during parameter tuning. As hunger can influence the activity range both positively and negatively, W_1 and W_2 are modeled to reflect this. In Eq. (10), the disparity between UB and LB illustrates the maximum hunger level under varying conditions; $F(i) - BF$ quantifies the remaining food required for an individual to satisfy hunger; Each iteration modifies an individual's hunger level. $WF - DF$ calculates an individual's total foraging potential in the current scenario; $\frac{F(i)-DF}{WF-DF}$ determines the hunger ratio; $r_6 \times 2$ assesses the environmental impact, either positive or negative, on hunger.

Algorithm 1 presents the pseudo-code for the HGS, while Fig. 2 illustrates the corresponding flowchart.

Initialize parameters: N, Max_iter, D, SHungry, and l.
Initialize positions of individuals $\vec{X}_i$ for all $i = 1, 2, \ldots, N$.
while $t \leq$ Max_iter **do** ▷ Main loop
    Evaluate fitness for each individual.
    Update best fitness BF, worst fitness WF, best individual position $\vec{X}_b$, and best individual index BI.
    Determine hunger level using Eq. 8.
    Compute weight W_1 using Eq. 6.
    Compute weight W_2 using Eq. 7.
    for each individual in the population **do**
        Calculate exploration factor E using Eq. 2.
        Adjust range of search R using Eq. 4.
        Update individual positions according to Eq. 1.
        Adjust X_i through the limits set for the variable's upper and lower boundaries.
    end for
    Increment iteration counter: $t = t + 1$.
end while
Return BF and $\vec{X}_b$.

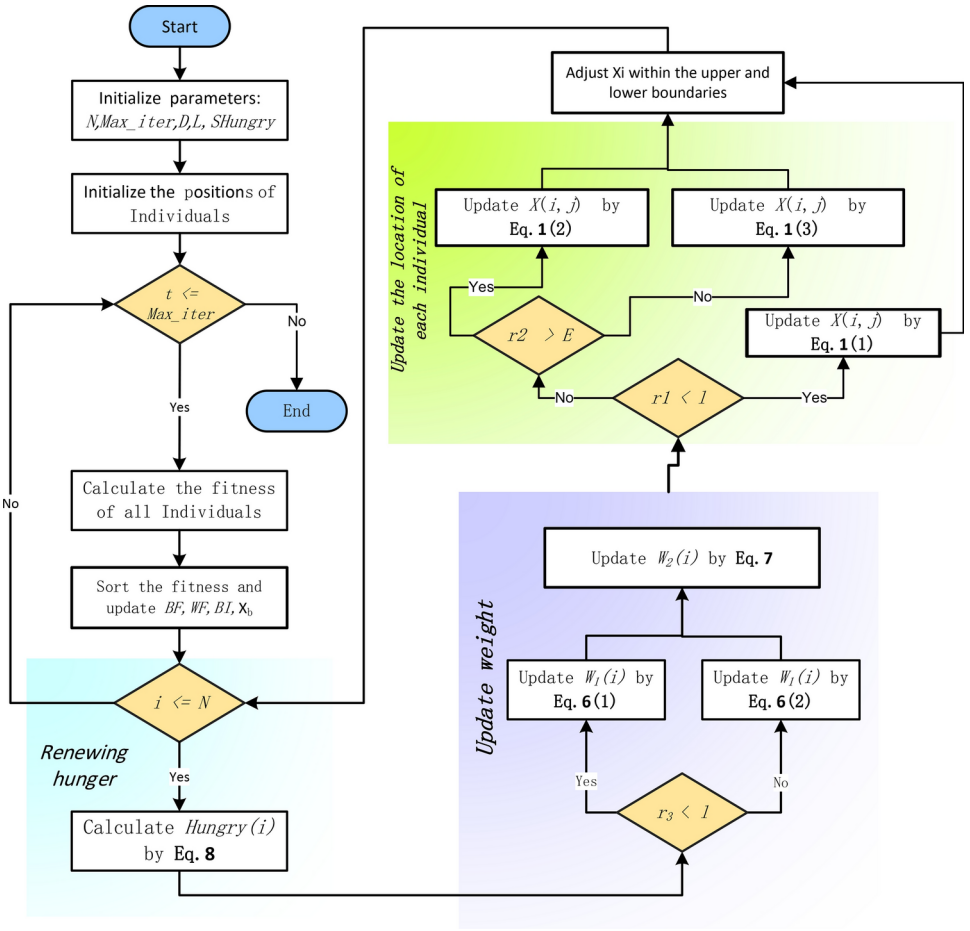


Fig. 2. Flowchart of the original hunger games search algorithm.

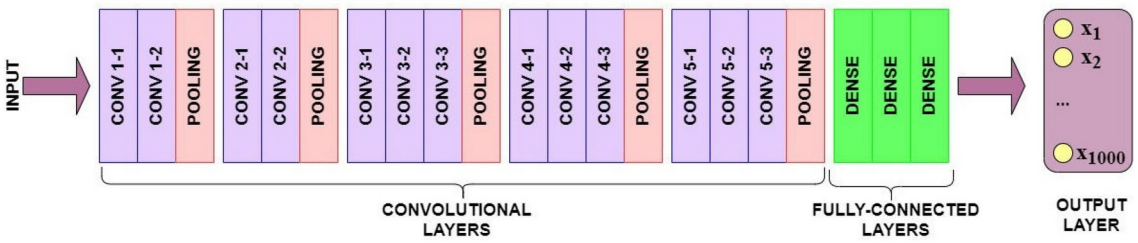


Fig. 3. Architecture of the VGG16 model, comprising 13 convolutional layers and 3 fully connected layers.

Algorithm 1. Pseudo-code of HGS

VGG16 deep learning model

The VGG16 (Visual Geometry Group), referred to as VGGNet, is a CNN deep learning model with 16 layers. Given its impressive performance in image classification tasks, the VGG-16 presented by Simonyan and Zisserma[94] has attracted a lot of attention. There are 16 layers total in the VGG-16 architecture, comprising 13 convolutional layers and 3 fully connected layers (see Fig. 3).

Rectified Linear Unit (ReLU) activation functions are put after each convolutional layer in the VGG-16 architecture. This makes it easier for the model to recognize patterns that are irregular. The network uses 3x3 filters with a stride of 1 and padding of 1 to capture fine spatial features while maintaining spatial resolution. By using many layers of convolution, this arrangement enables the model to learn complicated and abstract properties. Following each set of two or three convolutional layers, two max-pooling layers with a 2 × 2 window

and a stride of two are used. These max-pooling methods downsample the feature maps, preserving the most important data while lowering the spatial dimensions. In VGG-16, the fully connected layers are essential for high-level feature fusion and decision-making. The network has three fully connected layers, each with 4096 units, and then a final softmax layer for classification. This approach enables the model to learn intricate correlations between features and generate predictions using the learnt representations. The VGGNet design integrates key elements of CNNs. Here is a brief overview of the VGG architecture[94,95]:

- *Input layer*: VGGNet is fed with images of 224×224 pixel size. Concerning the consistency of inputs, the images in the ImageNet competition were brought to a standard form by cropping a 224×224 section from their center.
- *Convolutional layers*: The model utilizes 3×3 convolutional filters, the smallest possible size, to extract detailed features from the input image. Furthermore, a linear transformation is implemented by 1×1 convolutional filters. The same stride of one pixel is kept across these layers to ensure that the spatial resolution is preserved, and the network is therefore able to thoroughly analyze the image space.
- *ReLU activation*: VGGNet proceeds with convolutional layers, and then includes the ReLU activation function. The innovation, which was first used by the AlexNet approach, shortens the training time. ReLu activates only positive value inputs, outputting zero for negative value inputs, which greatly simplifies the computational process without compromising on effectiveness.
- *Hidden layers*: In the case of AlexNet, there is a use of local response normalization, but VGGNet uses only ReLU for its hidden layers. With such a selection, the model does not require any additional training time or memory utilization, as in the case of normalization. The model's accuracy is not affected, either.
- *Pooling layers*: This architecture consecutively applies convolutional layers and then pooling to decrease the dimensionality as well as the number of parameters in the resulting feature maps. Pooling is essential for managing the exponential increase in filter numbers, which grow from 64 in the initial layers to 512 in the later stages.
- *Fully connected layers*: The VGGNet architecture concludes with a stack of three fully connected layers. The first two layers possess 4096 channels each, preparing the network for the final classification layer. The final layer adapts to the dataset's number of classification categories (e.g., 1000 for ImageNet). By integrating high-level features, these layers facilitate critical decision-making and precision in the network.Finally, VGG-16 is recognized as one of the most notable DL models in terms of its simplicity and depth. In spite of being basic in architecture, it has proved to be successful in image classification cases and is regarded as a benchmark in the area of computer vision. The design principles of this model have influenced subsequent models for deeper layers with smaller filters. So far, these principles have demonstrated the fact that such architecture can substantially contribute to the improvement of performance while avoiding complexities. This methodology is not only a main component of the research on DL models but is also considered to be the standard for feature extraction techniques in various image processing applications[96,97].

Selective search for object proposal

Selective search is an image processing method that aims to find potential objects in an image[14]. It divides the image into different regions based on how they look (color, pattern, size) and then groups these regions to create objects. Figure 4 demonstrates the process of selective search applied to an input image. Uijlings et al.[98] was the first to develop this technique, which has since gained widespread use in the literature. According to[99], the algorithm employs segmentation to create potential object regions. Selective search starts with small image areas and progressively combines them to form bigger ones, resulting in a variety of object proposals with varying sizes and resolutions. This merging process simplifies the search process for object identification and recognition tasks, leading to greater efficiency and accuracy in computer vision applications.

Intersection over Union (IoU) for object proposal evaluation

The Intersection over Union (IoU) is frequently used in object detection as a metric for assessing possible target regions[100]. The approach computes the IoU between suggested regions (i.e. predicted bounding box) and true target regions (i.e. the ground truth bounding box). It gives higher ratings to areas with a higher IoU, suggesting

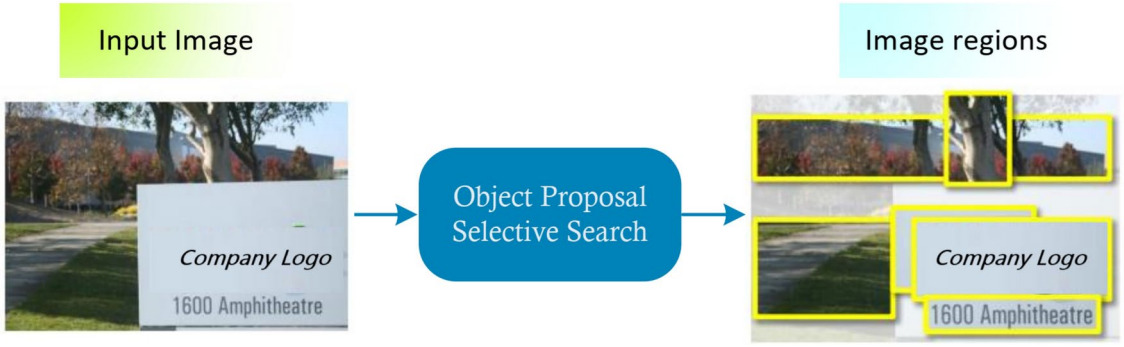


Fig. 4. The process of generating object proposals using selective search.

a more accurate alignment with the target items. This process assists in prioritizing the most favorable regions and reduces the search area, focusing computational resources on regions more likely to contain the target object. Thus, IoU is essential for enhancing the precision of object detection. It helps identify candidate regions for further analysis by CNNs or other classifiers. IoU serves as a benchmark for evaluating the performance of object detection algorithms. By defining a threshold, predicted bounding boxes can be deemed successful if their IoU with the actual bounding box (ground truth) meets or exceeds the threshold.

As shown in Fig. 5, the IoU calculates the intersection of the predicted bounding box with the ground-truth bounding box. A score of 0 on the IoU indicates no overlap, which is considered poor detection. Scores of around 0.7 or higher suggest good to excellent detection, with significant overlap. The IoU is calculated using Eq. (11).

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{11}$$

Proposed methodology for enhanced logo detection

The methodology adopted in our research was chosen to address the complexities involved in logo detection by using a series of interconnected components to improve accuracy and effectiveness. This methodology combines a powerful deep learning (DL) model with an innovative, nature-inspired optimization algorithm. Its functionality was evaluated based on a well-selected dataset. The pipeline consists of several phases: dataset selection and preprocessing, generation of object proposals, model configuration and optimization, hyperparameter tuning, and finally, the training and validation phases. The schematic diagram shown in Fig. 1 demonstrates the methodology used to develop the framework for detecting and classifying logos. While the basic stages are explained in the subsequent sections.

Dataset selection and preprocessing

Data selection

We performed an extensive search for already available datasets that may be used for logo classification. In this sense, we used the well-known FlickrLogos-27 dataset as a baseline for our analyses. The Flicker dataset contains a wide variety of annotated logo pictures, allowing us to thoroughly test and compare the performance of our model.

The FlickrLogos-27 dataset[101] is commonly used in logo recognition tasks. It consists of approximately 8000 images divided into 27 distinct logo categories, such as Adidas, Apple, BMW, Coca-Cola, Nike, and Starbucks. Samples of logos within the FlickrLogos collection can be found on the dataset's official website [https://www.uni-augsburg.de/en/fakultaet/fai/informatik/prof/mmc/research/datensatze/flickrlogos/]. The dataset includes numerous variations of each logo showcasing varying colors, sizes, orientations, and backgrounds. These variations are crucial for training and evaluating algorithms designed to identify logos in real-world scenarios.

The FlickrLogos-27 dataset consists of a diverse collection of logo images collected from various sources, ensuring a comprehensive representation of real-world logo variations. Each image has bounding box annotations pinpointing the logo's location and a class label specifying the represented logo. These annotations facilitate logo detection, identification, localization, and retrieval. The dataset poses challenges due to significant variations in logo appearance, including variations in size, orientation, lighting, obstructions, and background noise. To tackle these challenges, advanced computer vision algorithms are needed. The FlickrLogos-27 dataset is valuable because it showcases real-world logo variations. This allows us to test our models under realistic conditions and assess their accuracy. Additionally, the dataset helps us refine our models and handle specific issues related to logo identification. Overall, the diversity of FlickrLogos-27 ensures that the proposed model can adapt to different logo scenarios and thus contributing to developments in logo-related applications.

Preprocessing

The first major step in our logo detection pipeline is pre-processing the incoming images. This phase includes several basic techniques aimed at preparing and improving the quality of images and ensuring their suitability for optimal performance with the VGG-16 model.

- *Image Loading and region of interest extraction*: First of all, the images and their annotations are loaded from the dataset. Every record in the annotation file consists of three parts: the filename, the class label, and coordinates of the logo within the image (x1, y1, x2, y2). These annotations are applied to recognize the region of

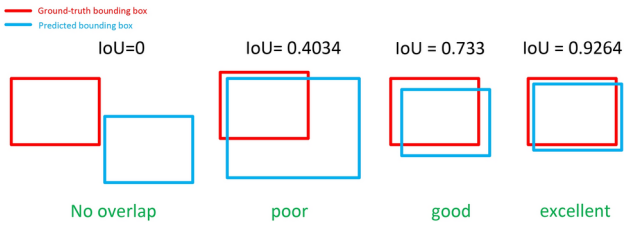


Fig. 5. Examples of IoU scores in object detection, illustrating various cases: no overlap, poor overlap, good overlap, and excellent overlap.

interest (ROI) for each image containing the logo and hence concentrate on the relevant portion of the image containing the logo for further analysis.
- *Image resizing*: In this step, each ROI is resized to 224 × 224 pixels to fit the input size requirements of the VGG-16 model. This resizing is performed using TensorFlow's resize function, which seamlessly scales images to the desired dimensions while maintaining the aspect ratio.
- *Preprocessing for VGG-16*: Following the resizing step, the images go through one more preprocessing step to prepare them for the input format expected by the VGG-16 model. The $'preprocess\_input'$ function from the Keras applications is utilized to prepare each image by applying several preprocessing adjustments, including channel reordering and pixel value scaling. In particular, it ensures that the channels are in the BGR order because VGG-16 was trained on images in this format, and it normalizes the pixel values by subtracting the mean of each color channel (B, G, R) across the entire ImageNet dataset from the corresponding channel in the input image (i.e. each color channel is zero-centered with respect to the ImageNet dataset). Through efficient pixel-by-pixel color normalization and channel modification, this technique sets up the images for precise analysis and categorization by the model.

Object proposal generation and data augmentation

In this study, we employed the selective search algorithm and the IoU metric described in "Selective search for object proposal" section for the generation and selection of object proposals. This approach was applied for two purposes: to extract regions containing logos within images and to enhance the dataset through augmentation, which will improve the model's training and recognition capabilities.

Using the selective search algorithm, we generated a wide range of potential object locations in images. The algorithm focuses on identifying regions that have a high probability of containing logos. This process is also useful for locating logos that may differ in size and appearance across different images. To evaluate the quality of the proposals, we employed the IoU score against the known logo annotations (the actual logos). We have set a threshold of 0.9 to assess the precision of the predictions. A predicted bounding box is considered a successful match if its IoU with the ground truth bounding box exceeds the selected threshold. After that, the selected regions are resized to the necessary size (224 × 224 pixels), and preprocessed to match VGG-16 model specifications. This process therefore augments the original dataset with high-quality logo instances and also enriches the training data by providing the model with diverse displays of logos to learn from.

Deep learning model configuration

This part aims to explore the detailed process of setting up the DL model for our logo detection framework. This involves making modifications and adding extra layers to customize the model according to our unique requirements. Following the fundamentals outlined in "VGG16 deep learning model" section, our practical approach incorporates the powerful features of the VGG-16 and carefully adjusted modifications to improve the accuracy of logo recognition.

Model construction

Our model relies on the VGG16 architecture. It has been preloaded with initial weights trained on the ImageNet dataset[102] to utilize the strong feature extraction abilities of this DL model. The VGG16 model is modified by removing the original top layer to enable the addition of a new top layer that's designed for logo classification. During the initial training phase, the layers of the base model are kept frozen to maintain the pre-trained features. Several layers are added to the base VGG16 model:

- A **flatten** layer is used to transform the 2D feature maps into a 1D vector.
- Two **dense** layers, each consisting of 4096 units and ReLU activation, are utilized to incorporate more learnable parameters for interpreting high-level features.
- A final **dense** prediction layer with several units matching the distinct number of logos in the dataset, is added. A softmax activation function is applied for multi-class classification.

Training procedure

For training the custom VGG16 model, the data set is first divided into two subsets: 75% for training and validation, and 25% for testing. The 75% portion designated for training and validation is divided further using an 80–20 split. This division results in 80% of the data being used for training purposes, with the remaining 20% allocated to validation. By following this systematic approach, we ensure comprehensive training of our model as well as a way to evaluate its effectiveness and extend its utility to unseen data.

To boost the training efficiency of our model and avoid overfitting, we have incorporated an early stopping mechanism. This approach monitors validation accuracy during training. If there is no progress for a specified number of epochs, training will stop automatically. Furthermore, this approach ensures that the model returns to the best-performing iteration weights, ensuring that the final model reflects the most effective version encountered during training. This method helps save computational resources by skipping unnecessary training cycles and keeps the model's performance generalizable without overfitting the training data.

Table 4 summarizes the configuration and training details of our customized VGG16 model.

Enhancement of the HGS algorithm (EHGS)

Before discussing the enhancements to the HGS algorithm, it's crucial to acknowledge the limitations of its standard version. Despite the effectiveness of HGS at handling global optimization problems, it has certain flaws, including low search veracity, limited population diversity, and a high potential for getting into a local optimum. These issues become more apparent when the algorithm is applied to intricate optimization problems. Moreover,

Parameter/component	Description/setting
Base model	VGG16, pre-trained on ImageNet, modified for logo detection
Optimizer	General gradient-based optimization method (e.g., Adam)
Loss function	Categorical crossentropy
Performance metrics	Precision, Recall, Accuracy
Input shape	$224 \times 224 \times 3$
Final layer activation	Softmax
Number of classes	Corresponding to the dataset
Dataset split	75% for training/validation; 25% for testing
Training/validation split	Within the 75%, an 80–20 split for training-validation
Early stopping	Monitors validation accuracy; stop after 5 epochs with no improvement;
	reverts to best-performing weights
Max epochs	15
Batch size	32

Table 4. Configuration and training details of the customized VGG16 model.

when inspecting the mathematical model of the typical HGS, it does not consider information from the optimal solution for every search agent (i.e., local best).

To enrich the search strategies of MHs, it is useful to consider, maintain, and track the best solutions obtained so far for each search agent. The success of this strategy in several algorithms such as PSO, CSA, and CapSA draws attention to this principle. This motivated our attempts to introduce an improved version of the HGS called the EHGS, which leverages the benefits of the local best optimum to dynamically change the behavior and search strategy of HGS. EHGS incorporates two separate update strategies for each individual in the population, based on whether an abandonment limit criterion is satisfied.

When the *i*th best local optimum ($pbest_i$) shows improvement, the basic rules of the HGS are applied. If there is no progress seen in the best local position ($pbest_i$) after *K* iterations, the update process for the corresponding individual x_i is stopped. Alternatively, a different update process inspired by the differential evolution (DE) algorithm[103] is used. This method involves applying a perturbation to the local optimum using Eq. (12).

$$X_i(t+1) = pbest_i + r\left(x_{rand} - x_i(t)\right) \tag{12}$$

where $pbest_i$ represents the best local position of the *i*th individual, x_{rand} refers to a position selected at random from a set of *N* solutions, and *r*, denotes a uniformly generated random number in the range [0, 1].

Algorithm 2 presents the pseudo-code of the proposed EHGS. Leveraging local optima and the abandonment limit strategy enables the search agent to investigate the surrounding area of its current best location. By incorporating a step size based on the difference between the randomly selected position and the current position, a more optimal solution can be identified. This process enhances the utilization of the best local optima and enhances the quality of the population.

Initialize parameters: N, Max_iter, D, SHungry, and l.
Initialize positions of individuals $\vec{X}_i$ for all $i = 1, 2, \ldots, N$.
Set the parameter for the abandonment limit (k)
define $count_i = 0$ (i=1,2,...,N)
Calculate the fitness of each search agent
Initialize local best memory $pbest_i$ of individuals
Update BF, WF, $\vec{X}_b$, and BI.
while ($t < T$) **do**
    for each individual i in the population **do**
        if ($count_i \leq K$) **then**
            Update X_i following the rules of the original HGS through Eqs. (1) to (8).
        else
            Update X_i using Eq. (12)
            $count_i = 0$
        end if
        Modify X_i within the established upper and lower boundary limits.
        Evaluate the fitness of each new position $f(xi)$
        if $f(x_i) < f(pbest_i)$ **then**
            $pbest_i = x_i$
            $f(pbest_i) = f(x_i)$
        else
            $count_i = count_i + 1$
        end if
            Update BF, WF, $\vec{X}_b$, and BI.
        end for
        Increment iteration counter: $t = t + 1$.
    end whileReturn BF and $\vec{X}_b$.

Algorithm 2. Pseudo-code of the enhanced HGS (EHGS)

Adapting HGS for hyperparameter tuning of VGG16 model

This section outlines the process of adapting the HGS and its enhanced version to optimize the hyperparameters of the VGG16 model for improved logo classification. The approach consists of two primary stages: first, formulating the hyperparameter tuning task as an optimization problem, and second, detailing the step-by-step integration of EHGS to search for the optimal hyperparameter configuration

Problem formulation

The objective of adapting the HGS algorithm for hyperparameter tuning is to identify the optimal set of hyperparameters that maximizes the classification performance of the VGG16 model on the logo classification task. When adapting any metaheuristic algorithm to handle an optimization problem, two critical components must be formulated: the solution representation and the fitness function. In this context:

- *Solution Representation*: In the context of HGS, a candidate solution is represented as a vector of hyperparameters specific to the VGG16 model and its training process. In this study, the solution vector includes the following hyperparameters: learning rate, number of neurons for dense layers, activation function for additional layers, optimizer, patience, and batch size. Table 5 provides the lower and upper bounds for each parameter. For categorical parameters, such as activation function and optimizer, indices are used for representation. For instance, activation function index (0: 'relu', 1: 'sigmoid', 2: 'tanh') and optimizer index (0: 'adam', 1: 'sgd', 2: 'rmsprop').

Hyperparameter	Lower bound	Upper bound
Learning rate	0.0001	0.01
Number of neurons	128	4096
Activation function index	0	2
Optimizer index	0	2
Patience	3	10
Batch size	16	128

Table 5. Hyperparameter search space and bounds for HGS optimization.

- *Fitness Function*: The fitness function, also known as the objective function, evaluates the classification performance of each candidate solution. For hyperparameter tuning, the fitness function is designed to maximize the validation accuracy of the VGG16 model. The fitness function can be formulated as follows:

$$\text{Fitness}(x) = -\text{val\_accuracy(x)}$$

where x is the candidate solution representing a set of hyperparameters. The accuracy is measured on the validation set of the logo classification task.

Procedure for integrating EHGS with VGG16
The integration of HGS and EHGS with the VGG16 model enables an automatic selection of the best hyperparameters to maximize VGG16's accuracy. The main steps of this process are outlined below:

- *Step 1: Initialization* This stage involves initializing the problem parameters, EHGS parameters, and the VGG16 hyperparameter search space. First, the objective function $\text{Fitness}(x) = -\text{val\_accuracy(x)}$, solution encoding ($x = [x_1, x_2, \ldots, x_n]$ where each x_i represents a hyperparameter), and search space dimensions are defind. The solution representation and fitness function, formulated in the previous subsection, serve as the foundation for the optimization process. The adjustable parameters of EHGS must also be set up at this stage. These include the population size, maximum number of generations, and any tuning parameters specific to the EHGS operators (e.g., hunger ratio and adaptation rate). Additionally, the search space bounds for each hyperparameter, such as learning rate, number of neurons, activation function, optimizer, patience, and batch size, must be established, as described in Table 5. Finally, the VGG16 model architecture is constructed with the necessary layers and adjustments for the logo classification task, as detailed in "Model construction" section.
- *Step 2: Generate Initial Population* In this step, EHGS follows the standard initialization process of the HGS algorithm. A population of candidate solutions $X = (x_1, x_2, x_3, \ldots, x_N)$ is initially positioned randomly within the defined hyperparameter search space. These candidates represent possible configurations of VGG16 hyperparameters for the optimization task. Each candidate solution x_j (where $j \in \{1, 2, 3, \ldots, N\}$) is determined using Eq. (13).

$$\vec{x}_j = \vec{x}_L + r(\vec{x}_U - \vec{x}_L) \tag{13}$$

where $\vec{x}_L$ and $\vec{x}_U$ represent the lower and upper bounds for each hyperparameter, respectively, and r is a random number uniformly distributed in the range $[0, 1]$. This initialization ensures a diverse set of starting solutions, enabling EHGS to explore a wide range of hyperparameter configurations.

- *Step 3: Fitness Evaluation* For each candidate solution in the population, EHGS evaluates the VGG16 model's classification performance using the hyperparameters represented by that solution. The fitness function is calculated based on validation accuracy, where higher validation accuracy indicates a better solution. During this evaluation, the VGG16 model is trained on the logo classification dataset using the candidate hyperparameter configuration
- *Step 4: Evolutionary Optimization through EHGS Operators* In this step, EHGS refines the population by using its mathematical operators, as described in Algorithm 2. These operators are designed to balance exploration (discovering new hyperparameter configurations) and exploitation (enhancing promising configurations) within the search space.
- *Step 5: Iteration and Stopping Criterion Check* The EHGS process iterates across multiple generations, continuously updating the population of candidate solutions. This iterative loop continues until the stopping criterion-such as a predefined number of generations-is satisfied. If the stopping condition is not met, the process returns to Step 3 to further optimize the candidate solutions.
- *Final Hyperparameter Selection* Once the EHGS algorithm meets the stopping criterion, it selects the top-performing set of hyperparameters, based on the highest validation accuracy, as the final configuration for training the VGG16 model on the logo classification task.

Code availability

The custom code and algorithms used in this study to generate results are available upon request. The code is hosted in a GitHub repository and can be accessed at the following link: [https://github.com/Thaer83/EHGS-Logo-Classification.git]. The repository includes the code for all experiments, including the implementation of swarm intelligence algorithms for the CEC2014 benchmark functions, the deep learning models for logo classification, and the hyperparameter tuning using the EHGS algorithm. Access to the code is available for public use with proper citation. If any access restrictions apply, they will be clearly described on the repository page.

Performance evaluation

Logo classification performance metrics

To thoroughly evaluate the performance of our logo detection model, we used a series of common metrics for assessing multi-task classification models. We considered three primary metrics: accuracy, recall, and precision.

- *Accuracy*: this measure determines the proportion of correct predictions to the total number of observations, indicating the model's overall correctness. It is calculated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP denotes true positives, TN true negatives, FP false positives, and FN false negatives.

- *Recall* (sensitivity): This metric quantifies the model's ability to accurately identify all true positives. It is crucial when missing a positive instance is significant. It is calculated as:

$$Recall = \frac{TP}{TP + FN}$$

- *Precision*: is a measure of the model's accuracy in making positive predictions, demonstrating its ability to distinguish between different logos. The formula for precision is:

$$Precision = \frac{TP}{TP + FP}$$

It is important to note that the Precision and Recall metrics are calculated as an average across all classes in a multi-class classification context.

Evaluation of HGS and EHGS algorithms

To evaluate the performance of the HGS and its enhanced version EHGS, we focus on two specific metrics:

- Fitness values: This metric assesses the optimization capability of the algorithms by measuring the quality of solutions found. A lower fitness value indicates a better solution.
- Diversity Measure: This metric evaluates the algorithms' ability to explore a broad range of solutions by analyzing the variability among the solutions produced. A higher diversity score implies that the algorithms can explore various areas of the search space, crucial for avoiding early convergence to suboptimal solutions. The calculation of this measure refers to[33].

Experimental results and simulations

This section evaluates the Enhanced Hunger Games Search (EHGS) algorithm and its application in logo classification through a custom-developed VGG16 deep learning model. The assessment includes three parts. Firstly, we test EHGS against benchmark functions from the IEEE CEC2014 to show its effectiveness. These benchmark functions are particularly suited for evaluating the behavior and properties of optimization algorithms, including their robustness, convergence rate, and overall performance factors[33]. These problems are framed as minimization problems with a known global minimum, allowing for precise evaluation. CEC2014 benchmark functions are modified versions of other challenging mathematical optimization problems that have been rotated, shifted, extended, and combined[104]. They can be grouped into four categories based on their characteristics: unimodal, multimodal, hybrid, and composition functions. Accordingly, they offer a comprehensive testing environment that simulates a range of optimization challenges. In specific, they are useful to evaluate the search and optimization capabilities of EHGS in a controlled environment. By performing well on these benchmarks, EHGS demonstrates its potential for effective hyperparameter tuning in more complex, real-world tasks like logo classification. In the second phase, we examine the custom VGG16 model's performance for logo classification and compare it to other deep-learning models. Lastly, we integrate EHGS into the VGG16 model for hyperparameter optimization and show how it improves accuracy in logo classification. Overall, this section shows the benefits of combining evolutionary optimization techniques with deep learning for better accuracy and efficiency in logo classification tasks. Due to the stochastic nature of the algorithms under examination, computational intelligence research has increasingly focused on non-parametric statistical analysis[105]. In this context, two non-parametric tests-the Friedman test and the Wilcoxon signed-rank test-were applied at a 5% significance level to calculate the overall ranking of each algorithm and to perform specific pairwise comparisons.

All experiments were conducted in the same environment to ensure fair comparisons. The first part of the experiments, where HGS, EHGS, and the compared algorithms were tested, was conducted on a machine running Ubuntu 20.04 LTS, equipped with an Intel(R) Core(TM) i7-1165G7 CPU at 2.80 GHz (8 CPUs) and 16 GB of RAM. The second and third experimental tasks, which involved deep learning for logo classification and the integration of EHGS with VGG16, were performed in the Google Colab environment using the n1-highmem-4 machine type with NVIDIA Tesla T4 GPU acceleration and 100 GB Standard Disk (pd-standard)[106].

Google Colab provides powerful computational resources, facilitates Python library integration, and enables efficient model training and testing, making it an ideal platform for extensive deep learning experiments.

Validation of EHGS on benchmark functions

In this section, a comprehensive evaluation is provided to compare the basic HGS algorithm with its advanced version, EHGS. "Comparison between standard HGS and EHGS" section focuses on evaluating the improvements in EHGS over the standard HGS, while "Comparison of EHGS with established metaheuristic methods" section extends the analysis by comparing EHGS with other well-known metaheuristic algorithms. The comparison is conducted on a series of IEEE CEC 2014 benchmark functions through rigorous analysis. The study includes 30 independent runs for each algorithm to determine the statistical and practical significance of the enhancements integrated into EHGS. All optimizers are evaluated utilizing the same standard configurations and settings. The detailed parameters of the *HGS* and the competitor algorithms are reported in Table 6

Comparison between standard HGS and EHGS

In this subsection, we compare the performance of the basic HGS algorithm with its advanced variant, EHGS. The comparison aims to validate how the modifications in EHGS, such as enhanced exploration and exploitation mechanisms, lead to better overall performance, particularly in terms of solution quality, convergence rate, and population diversity. Table 7 reports the results of a comparative study between HGS and EHGS. In this comparison, the symbols "*W*", "*L*", and "*T*", representing Win, Loss, and Tie, respectively, are used to indicate whether EHGS outperforms, underperforms, or matches the performance of the compared algorithm. As depicted in Table 7, The results indicate that EHGS outperforms the basic HGS algorithm in 22 out of 30 cases, which is approximately 73% of the test functions. This suggests that the enhancements made to EHGS have significantly improved its optimization capabilities, making it more effective in finding better solutions more consistently than the basic version. Interestingly, both EHGS and basic HGS perform equally well in 7 test functions, which is around 23% of the total test cases. Notably, for test functions F23, F24, F25, F27, F28, F29, and F30, both algorithms are able to find the optimal solutions, indicating that both versions of HGS are well-suited for these particular optimization landscapes.

A nonparametric Wilcoxon Ranksum test was performed to establish a reliable statistical foundation for EHGS results and introduce meaningful comparisons. The outcomes are presented in Table 7, where the symbols '+,' '−,' and '≈' are used to indicate the statistical superiority, inferiority, or similarity of EHGS with its counterpart. Additionally, instances marked with "NAN" suggest that there is no significant difference between HGS and alternate EHGS. A 'NaN' *p*-value indicates that the data from both groups being compared are practically equivalent. We can observe that EHGS exhibits superior performance compared to the basic HGS

Common parameters		
Population size *N*	30	
Maximum No. of iterations	1000	
No. of runs	30	
Significance level α (Friedman test)	0.05	
Internal parameters		
Algorithm	**Parameter**	**Value**
SCA	*r*1	Decreased linearly [2 0]
	*r*2	Random values inside [0 2π]
	*r*3	Random values inside [0 2]
	*r*4	Random values inside [0 1]
HGS	Hunger threshold (*LH*)	1000
	Probability of updating position *l*	0.08
HHO	Convergence constant (*E*)	Decreased linearly [2 0]

Table 6. Settings for common and algorithm-specific parameters for HGS and comparison methods (BAT, HHO, and SCA).

Function	HGS		EHGS		Wilcoxon Rank Sun		Diversity	
	Avg	Std	Avg	Std	*P*-value	Sig.	HGS	EHGS
F1	4.1633E+08	1.49E+08	**3.3732E+08**	1.67E+08	**0.0489**	+	1.58E+05	**2.08E+05**
F2	2.9822E+10	1.02E+10	**2.3447E+10**	7.85E+09	**0.0117**	+	1.22E+05	**1.42E+05**
F3	8.6504E+04	7.54E+03	**8.2105E+04**	7.80E+03	**0.0087**	+	1.71E+04	**1.90E+04**
F4	3.7944E+03	1.84E+03	**2.5521E+03**	9.59E+02	**0.0037**	+	1.09E+05	**1.14E+05**
F5	5.2086E+02	1.33E−01	**5.2083E+02**	1.09E-01	0.5106	=	**3.23E+05**	2.87E+05
F6	**6.4209E+02**	2.36E+00	6.4260E+02	2.59E+00	0.4035	=	**2.30E+05**	1.25E+05
F7	8.9618E+02	7.23E+01	**8.2818E+02**	5.20E+01	**0.0004**	+	**1.60E+05**	1.58E+05
F8	1.0783E+03	2.20E+01	**1.0607E+03**	3.11E+01	**0.0207**	+	3.12E+04	**3.96E+04**
F9	1.2230E+03	3.57E+01	**1.2107E+03**	2.93E+01	0.1907	=	3.06E+04	**3.52E+04**
F10	7.4926E+03	6.28E+02	**7.4912E+03**	7.76E+02	0.9823	=	**2.44E+05**	1.85E+05
F11	7.9613E+03	8.04E+02	**7.9018E+03**	5.61E+02	0.4918	=	**3.08E+05**	2.50E+05
F12	1.2029E+03	5.94E−01	**1.2023E+03**	4.69E-01	**0.0012**	+	**3.29E+05**	2.57E+05
F13	1.3045E+03	7.45E−01	**1.3039E+03**	9.31E-01	**0.0117**	+	1.07E+05	**1.41E+05**
F14	1.5006E+03	2.24E+01	**1.4672E+03**	2.26E+01	**0.0000**	+	**1.36E+05**	1.35E+05
F15	2.0431E+04	1.46E+04	**1.6567E+04**	1.03E+04	0.4035	=	7.34E+04	**7.60E+04**
F16	1.6132E+03	4.25E−01	**1.6131E+03**	3.45E-01	0.5395	=	1.41E+05	**2.37E+05**
F17	4.7280E+07	4.11E+07	**2.3024E+07**	2.45E+07	**0.0112**	+	1.94E+05	**2.00E+05**
F18	8.8264E+07	1.11E+08	**3.5572E+07**	5.25E+07	**0.0033**	+	**1.75E+05**	1.71E+05
F19	2.1559E+03	1.06E+02	**2.1314E+03**	8.65E+01	0.3042	=	1.07E+05	**1.49E+05**
F20	3.1814E+05	2.27E+05	**1.7785E+05**	1.34E+05	**0.0127**	+	1.15E+05	**1.24E+05**
F21	2.4255E+07	2.16E+07	**1.5993E+07**	1.42E+07	0.1537	=	2.13E+05	**2.68E+05**
F22	3.7424E+03	8.56E+02	**3.6339E+03**	5.04E+02	0.9587	=	2.30E+05	**2.85E+05**
F23	**2.5000E+03**	0.00E+00	**2.5000E+03**	0.00E+00	NaN	=	0.00E+00	**5.86E-29**
F24	**2.6000E+03**	0.00E+00	**2.6000E+03**	0.00E+00	NaN	=	0.00E+00	**2.43E-27**
F25	**2.7000E+03**	0.00E+00	**2.7000E+03**	0.00E+00	NaN	=	0.00E+00	**1.78E-25**
F26	2.7684E+03	4.55E+01	**2.7649E+03**	4.69E+01	0.6961	=	4.16E+04	**8.53E+04**
F27	**2.9000E+03**	0.00E+00	**2.9000E+03**	0.00E+00	NaN	=	0.00E+00	**2.15E-27**
F28	**3.0000E+03**	0.00E+00	**3.0000E+03**	0.00E+00	NaN	=	0.00E+00	**6.12E-28**
F29	**3.1000E+03**	0.00E+00	**3.1000E+03**	0.00E+00	NaN	=	0.00E+00	**6.66E-41**
F30	**3.2000E+03**	0.00E+00	**3.2000E+03**	0.00E+00	NaN	=	0.00E+00	**9.50E-39**
W|T|L	1|7|22		22|7|1		12|18|0			

Table 7. Comparative analysis of basic HGS and EHGS on IEEE CEC 2014 benchmark functions over 30 independent runs, showing average, standard deviation, Wilcoxon test *p*-values, and significance (Sig.), indicating whether EHGS is superior, inferior, or equivalent to basic HGS. Significant values are in [bold].

in a significant number of cases. The validity of these findings is reinforced by p-values ≤ 0.05, indicating that these improvements are statistically significant.

The Diversity results in Table 7 offer valuable information about the population diversity maintained by the HGS and the EHGS algorithms upon completion of the optimization process. A higher diversity means that the algorithm is exploring a wider range of the search space, which can prevent convergence and help to find better solutions. It is clear that the EHGS algorithm generally maintains a higher diversity compared to the basic HGS. The enhancements made to the algorithm help it to explore the search space better, which is important for solving complex optimization problems. The balance between exploration and exploitation in EHGS helps it to find better solutions while considering a broader set of potential solutions, making it more robust and reliable for the search process.

The comparative analysis of the convergence and diversity curves of EHGS and HGS across selected benchmark functions is demonstrated in Fig. 6. It was found that EHGS outperforms basic HGS with its superior performance. EHGS is more efficient in converging to lower fitness values, indicating a more effective search strategy for optimal solutions. Moreover, EHGS maintains higher solution diversity throughout the optimization process, ensuring a broader exploration of the search space. This minimizes the risk of premature convergence to local optima and highlights the algorithm's enhanced robustness in complex optimization landscapes.

Comparison of EHGS with established metaheuristic methods

To further validate the effectiveness of EHGS, we compare it with three well-known metaheuristic algorithms, including Bat Algorithm (BA)[107], Harris Hawks Optimization (HHO)[36], and Sine Cosine Algorithm (SCA)[108]. The comparison between EHGS and these algorithms across F1-F30 functions is presented in Table 8. The average

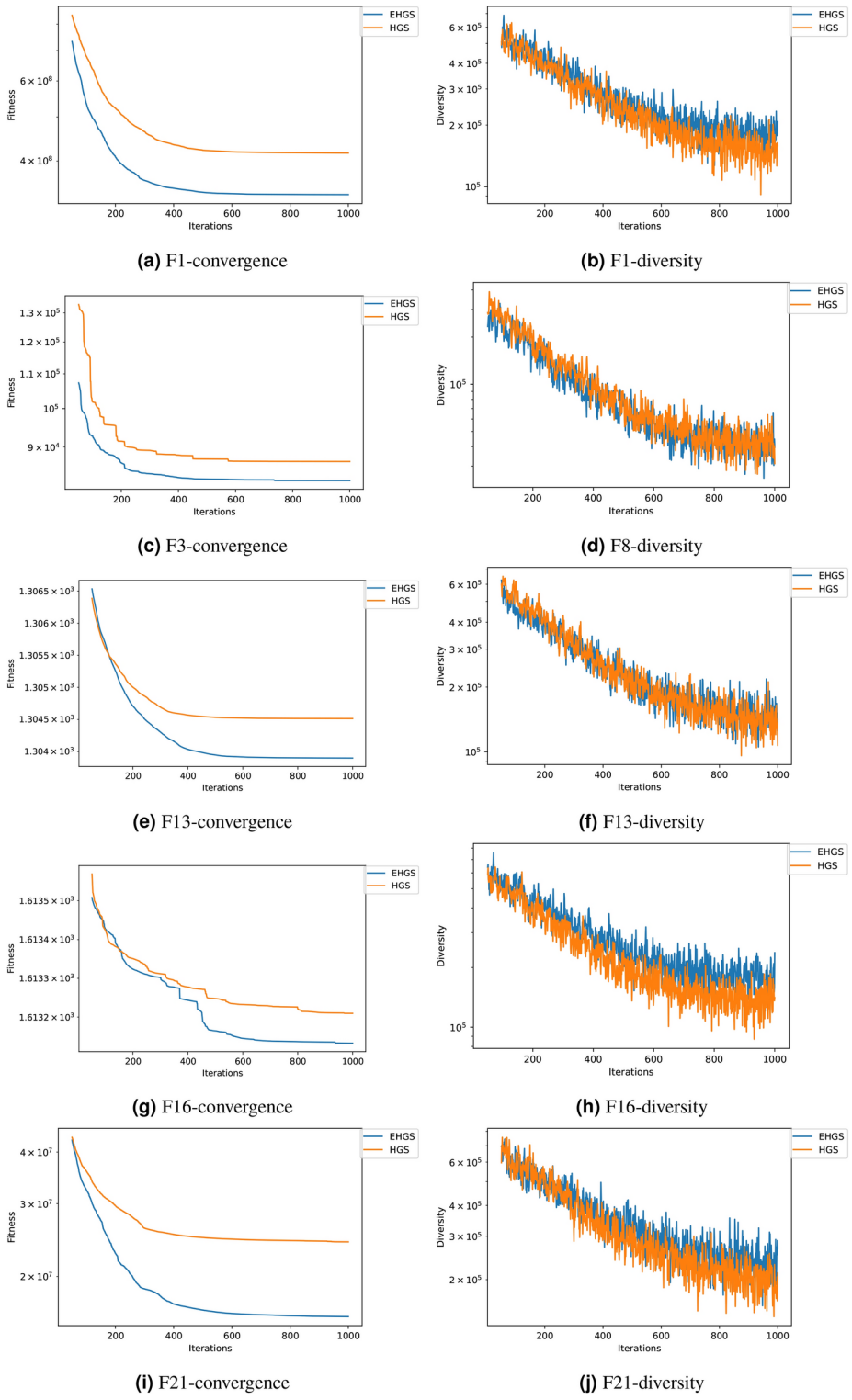


Fig. 6. Convergence and diversity curves of HGS variants on sample of CEC2014 benchmarks.

and standard deviation of the best-obtained fitness values for 30 independent runs are reported. The mean rank, which represents the algorithm's average ranking, is derived from the Friedman test. Referring to Table 8, EHGS performs well across a variety of benchmark functions. When comparing EHGS to each algorithm individually, EHGS demonstrated superior performance compared to BAT on 29 out of 30 functions. In comparison with SCA, EHGS performed better on 19 functions. Against HHO, EHGS outperformed HHO on 16 functions, while HHO achieved better results on 9 functions, indicating competitive performance between the two algorithms.

Function	Measure	BA	HHO	SCA	EHGS
F1	AVG	1.985E+09	7.644E+08	4.295E+08	**3.373E+08**
	STD	8.72E+08	2.37E+08	1.14E+08	1.67E+08
F2	AVG	9.109E+10	5.656E+10	2.515E+10	**2.345E+10**
	STD	1.97E+10	8.55E+09	4.05E+09	7.85E+09
F3	AVG	3.475E+05	8.122E+04	**6.293E+04**	8.211E+04
	STD	1.54E+05	6.22E+03	1.25E+04	7.80E+03
F4	AVG	2.058E+04	9.623E+03	**2.311E+03**	2.552E+03
	STD	6.90E+03	2.93E+03	7.10E+02	9.59E+02
F5	AVG	**5.203E+02**	5.204E+02	5.210E+02	5.208E+02
	STD	2.76E−01	2.16E−01	4.63E−02	1.09E−01
F6	AVG	6.471E+02	6.410E+02	**6.362E+02**	6.426E+02
	STD	2.19E+00	2.77E+00	2.71E+00	2.59E+00
F7	AVG	1.642E+03	1.113E+03	9.214E+02	**8.282E+02**
	STD	1.28E+02	8.29E+01	3.91E+01	5.20E+01
F8	AVG	1.168E+03	**1.036E+03**	1.075E+03	1.061E+03
	STD	3.79E+01	2.01E+01	1.98E+01	3.11E+01
F9	AVG	1.314E+03	**1.166E+03**	1.204E+03	1.211E+03
	STD	5.06E+01	3.09E+01	2.36E+01	2.93E+01
F10	AVG	7.939E+03	**6.724E+03**	7.894E+03	7.491E+03
	STD	8.55E+02	7.16E+02	3.74E+02	7.76E+02
F11	AVG	8.198E+03	**7.537E+03**	8.563E+03	7.902E+03
	STD	1.06E+03	7.87E+02	3.24E+02	5.61E+02
F12	AVG	1.204E+03	**1.202E+03**	1.203E+03	1.202E+03
	STD	9.23E−01	6.03E−01	3.54E−01	4.69E−01
F13	AVG	1.309E+03	1.307E+03	**1.304E+03**	1.304E+03
	STD	1.39E+00	8.95E−01	4.29E−01	9.31E−01
F14	AVG	1.723E+03	1.579E+03	1.472E+03	**1.467E+03**
	STD	7.61E+01	2.87E+01	1.27E+01	2.26E+01
F15	AVG	5.287E+06	4.910E+04	1.926E+04	**1.657E+04**
	STD	6.34E+06	2.68E+04	1.96E+04	1.03E+04
F16	AVG	1.614E+03	**1.613E+03**	1.613E+03	1.613E+03
	STD	5.96E−01	3.32E−01	2.20E−01	3.45E−01
F17	AVG	1.192E+08	7.413E+07	**1.560E+07**	2.302E+07
	STD	7.85E+07	5.46E+07	9.68E+06	2.45E+07
F18	AVG	4.933E+09	2.330E+09	3.359E+08	**3.557E+07**
	STD	2.19E+09	1.59E+09	1.81E+08	5.25E+07
F19	AVG	2.642E+03	2.226E+03	**2.021E+03**	2.131E+03
	STD	2.52E+02	9.58E+01	2.25E+01	8.65E+01
F20	AVG	5.509E+06	3.033E+05	**3.835E+04**	1.778E+05
	STD	9.76E+06	3.34E+05	1.73E+04	1.34E+05
F21	AVG	6.249E+07	3.026E+07	**3.549E+06**	1.599E+07
	STD	5.31E+07	3.09E+07	1.46E+06	1.42E+07
F22	AVG	6.009E+03	5.511E+03	**3.260E+03**	3.634E+03
	STD	3.35E+03	7.92E+03	1.79E+02	5.04E+02
F23	AVG	3.705E+03	**2.500E+03**	2.721E+03	**2.500E+03**
	STD	4.19E+02	0.00E+00	2.62E+01	0.00E+00
F24	AVG	2.792E+03	2.600E+03	2.606E+03	**2.600E+03**
	STD	4.47E+01	2.54E−04	5.87E+00	0.00E+00
F25	AVG	2.820E+03	**2.700E+03**	2.736E+03	**2.700E+03**
	STD	4.71E+01	0.00E+00	7.50E+00	0.00E+00
F26	AVG	2.799E+03	2.779E+03	**2.704E+03**	2.765E+03
	STD	1.01E+02	3.93E+01	3.33E−01	4.69E+01
F27	AVG	4.278E+03	**2.900E+03**	3.763E+03	**2.900E+03**
	STD	1.54E+02	0.00E+00	2.98E+02	0.00E+00
F28	AVG	6.989E+03	**3.000E+03**	5.391E+03	**3.000E+03**

Continued

Function	Measure	BA	HHO	SCA	EHGS
	STD	1.07E+03	0.00E+00	4.51E+02	0.00E+00
F29	AVG	1.066E+08	**3.100E+03**	2.497E+07	**3.100E+03**
	STD	6.82E+07	0.00E+00	1.00E+07	0.00E+00
F30	AVG	3.696E+06	2.011E+05	5.780E+05	**3.200E+03**
	STD	3.28E+06	5.56E+05	2.09E+05	0.00E+00
Mean rank		3.87	2.22	2.15	**1.77**

Table 8. Comparison between EHGS and other metaheuristics on IEEE CEC 2014 functions over 30 independent runs i terms of average and standard deviation. Significant values are in [bold].

Model	Measure	Classification quality			Inference tme
		Accuracy	Precision	Recall	
VGG16	AVG	**0.956966**	**0.957137**	**0.956966**	77.431
	STD	0.008421	0.008646	0.008421	4.313
DensNet121	AVG	0.394004	0.699926	0.275132	38.010
	STD	0.105727	0.122067	0.143514	3.087
EfficientNetV0	AVG	0.944974	0.949677	0.942328	22.042
	STD	0.013089	0.010853	0.014605	0.386
InceptionV3	AVG	0.23545	0.783533	0.040741	18.541
	STD	0.023539	0.143987	0.021039	0.255
MobileNetV2	AVG	0.943386	0.945192	0.943034	**13.685**
	STD	0.014428	0.013762	0.014632	0.671
ResNET	AVG	0.223457	0.634886	0.015344	33.608
	STD	0.045445	0.197309	0.008586	4.785
VGG16-NP	AVG	0.413051	0.630518	0.288183	77.045
	STD	0.229115	0.228787	0.245976	0.568

Table 9. Comparative evaluation of custom VGG16 and other DL models in terms of averaged classification metrics and inference timing over five independent runs. Significant values are in [bold].

In terms of mean rank, EHGS achieved the first rank with a mean rank of 1.77, outperforming BAT (3.87), HHO (2.22), and SCA (2.15).

Evaluation of the proposed approach for logo classification

Following the successful validation of the proposed improvements to the HGS algorithm, this section assesses the application of the EHGS-optimized VGG16 framework for logo detection. Our investigation involves two main aspects: evaluating the performance of the customized VGG16 model and establishing a benchmark for accuracy in logo classification. Next, we will examine the effects of hyperparameter optimization using EHGS on this model.

Comparison of deep learning models

In this section, we have conducted a comprehensive analysis of various DL models, including VGG16, VGG16-NP, DenseNet121, EfficientNetV0, InceptionV3, MobileNetV2, and ResNet. All these models utilize pre-trained weights on the ImageNet dataset, except for one variant of VGG16, which we refer to as VGG16-NP (not pretrained). Our analysis is based on the evaluation of classification quality metrics, such as accuracy, precision, and recall, as well as the inference time (testing time) of each model. We conducted five independent runs for each experiment and reported the results in terms of average and standard deviation. Moreover, we have visualized the training and validation accuracy curves over epochs, along with the training and validation loss.

Based on the outcomes reported in Table 9 and visualized in Figs. 7 and 8, it can be observed that VGG16 is the best-performing model in terms of average classification metrics, as indicated by its consistent performance across the box plots for accuracy, precision, and recall. EfficientNetV0 and MobileNetV2 are closely following VGG16 and offer competitive outcomes, as shown by the narrow interquartile ranges in their respective box plots. These models also exhibit consistency in their performance, as indicated by lower standard deviations. However, VGG16's high accuracy comes with the trade-off of longer inference time. MobileNetV2 stands out for its exceptional balance between high classification accuracy and the lowest inference time, making it suitable for real-time applications that require both speed and precision.

Conversely, DenseNet121, InceptionV3, and ResNet display comparatively poor performance with significantly lower average scores in accuracy and recall. InceptionV3, despite having reasonable precision, exhibits low recall, indicating limitations in its ability to identify all relevant instances. Similarly, ResNet shows

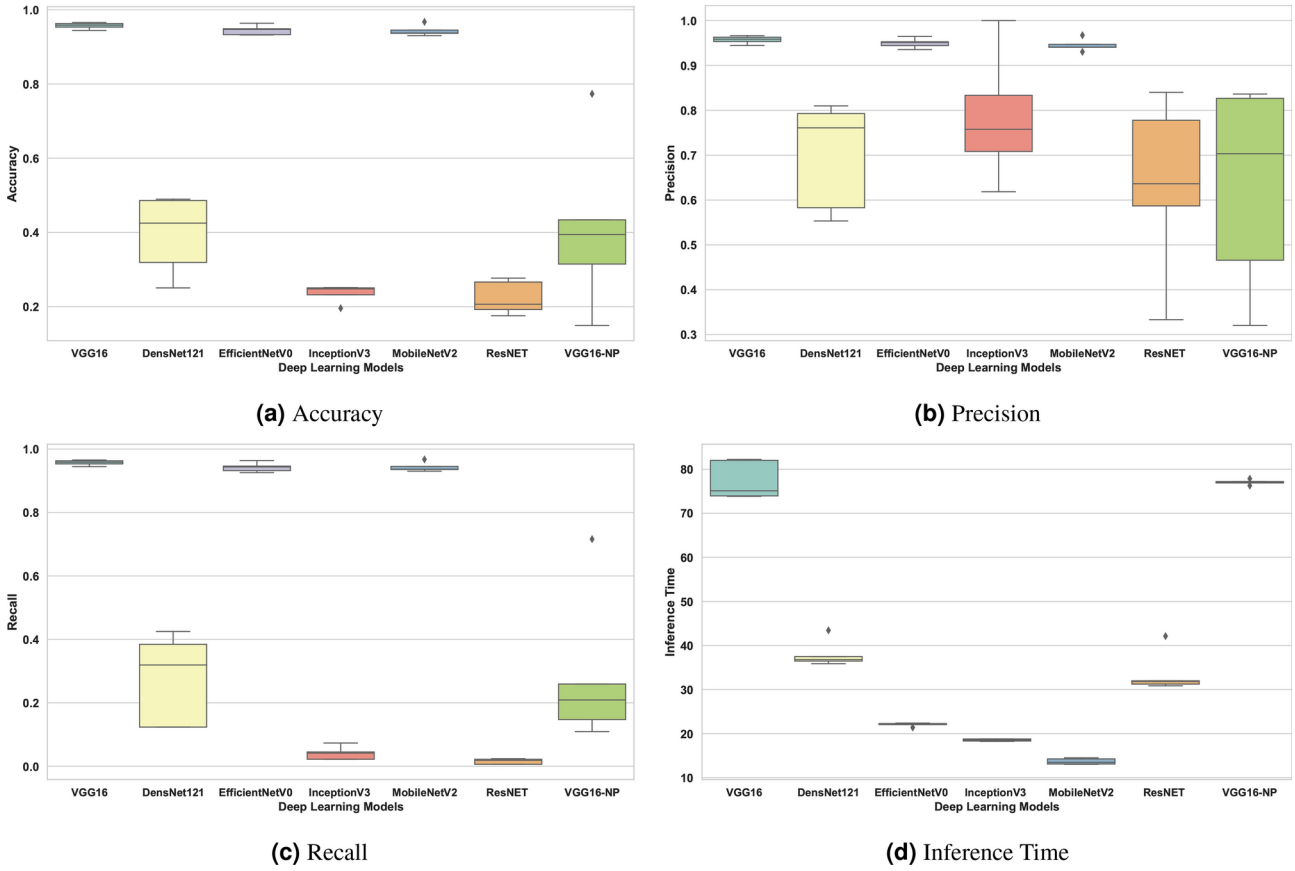


Fig. 7. Box-and-whiskers plot of deep learning model performance in terms of classification quality and inference time.

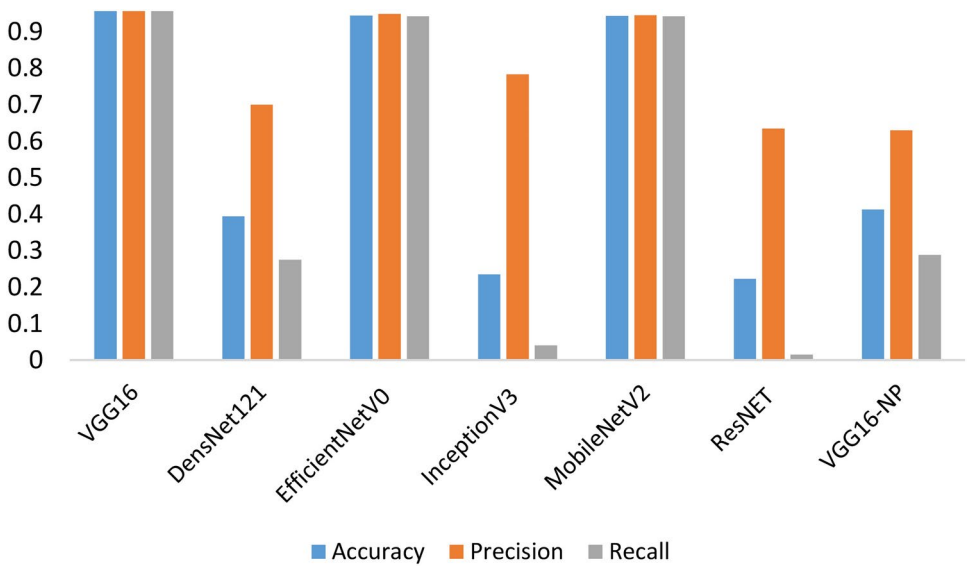


Fig. 8. Bar chart of classification metrics reflecting results in Table 9.

moderate precision but suffers from very low recall, which can be critical depending on the application. Lastly, DenseNet121 has the lowest accuracy and recall, which could be a concern for tasks requiring high reliability. Our study focuses on identifying the model that demonstrates the best classification quality overall. As a result, VGG16 is recognized as the best-performing model in this study, prioritizing comprehensive classification quality.

Model	Measure	Classification Quality			Inference Time
		Accuracy	Precision	Recall	
DensNet121	*P*-value	**4.311E−02**	**4.311E−02**	**4.311E−02**	**4.311E−02**
	Sig.	+	+	+	-
EfficientNetV0	*P*-value	7.962E−02	2.249E−01	7.962E−02	**4.311E−02**
	Sig.	=	=	=	-
InceptionV3	*P*-value	**4.311E−02**	7.962E−02	**4.311E−02**	**4.311E−02**
	Sig.	+	=	+	-
MobileNetV2	*P*-value	2.249E−01	2.249E−01	2.249E−01	**4.311E−02**
	Sig.	=	=	=	-
ResNET	*P*-value	**4.311E−02**	**4.311E−02**	**4.311E−02**	**4.311E−02**
	Sig.	+	+	+	-
VGG16-NP	*P*-value	**4.311E−02**	**4.311E−02**	**4.311E−02**	6.858E−01
	Sig.	+	+	+	=

Table 10. Wilcoxon Signed-Rank Test results for pairwise comparison of VGG16 with other tested models across accuracy, precision, recall, and inference time. Significant values are in [bold].

Model	Run 1	Run 2	Run 3	Run 4	Run 5	Average
VGG16	14	**15**	10	14	15	13.6
DensNet121	6	8	13	10	7	8.8
EfficientNetV0	**15**	13	**15**	11	10	12.8
InceptionV3	7	6	6	10	6	7
MobileNetV2	**15**	**15**	13	13	15	14.2
ResNET	6	6	7	6	6	6.2
VGG16-NP	**15**	**15**	9	15	15	13.8

Table 11. Comparison of DL models on early stopping epoch counts across five training runs. Significant values are in [bold].

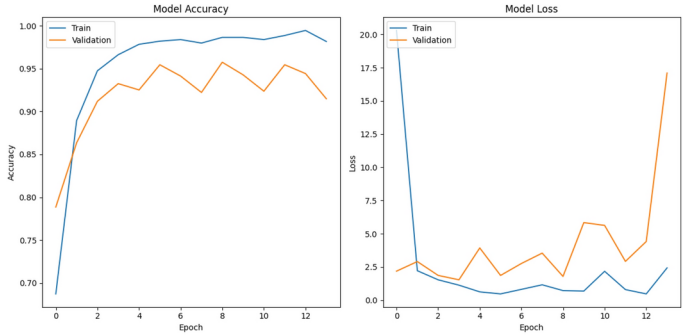
The Wilcoxon Signed-Rank Test results, shown in Table 10, present the comparative performance of the VGG16 model with other tested models across accuracy, precision, recall, and inference time. The *p*-values indicate statistical significance, with VGG16 showing superior performance (+) in accuracy, precision, and recall compared to DenseNet121, InceptionV3, ResNet, and VGG16-NP, as denoted by *p*-values below the significance level of 0.05. However, VGG16 demonstrates equivalent performance (=) with EfficientNetV0 and MobileNetV2 across classification quality measures, as indicated by the non-significant *p*-values. In terms of inference time, VGG16 exhibited a statistically higher time compared to all other models except VGG16-NP.

Table 11 presents the number of epochs required for different DL models to reach an early stopping point over 5 independent runs, with the last column displaying the average number of epochs across these runs. Early stopping is a form of regularization used to prevent overfitting by terminating the training process if the model's performance on a validation set does not improve for a specified number of epochs. From the table, we can observe that VGG16 and its non-pretrained version, VGG16-NP, require more epochs to train, averaging 13.6 and 13.8 epochs, respectively. This suggests a slower convergence to the early stopping threshold. MobileNetV2 outperforms both, with the highest average of 14.2 epochs. This could indicate a stronger resistance to overfitting or simply a learning pattern that favors longer training periods before meeting the early stopping criteria. On the otherhand, ResNET reaches the early stopping point significantly faster, with an average of only 6.2 epochs. This suggests that ResNET either achieves optimal validation performance quickly or overfits to the training data at an early stage.

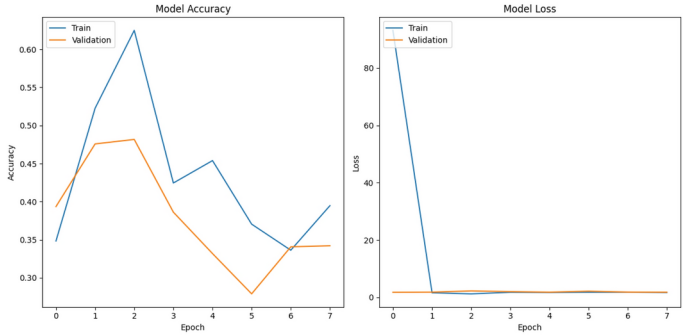
The visualizations in the form of accuracy and loss curves for various DL models provide insights into their learning behavior over epochs. behaviors are illustrated in Figs. 9 and 10. It is clear that VGG16, EfficientNetV0, and MobileNetV2 show higher and more stable accuracy over epochs, indicating robust learning and generalization. In contrast, DenseNet121, InceptionV3, ResNet, and the VGG16-NP variant exhibit less stable learning, with either declining accuracy, indications of overfitting, or high variability in loss.

Impact of EHGS on hyperparameter tuning

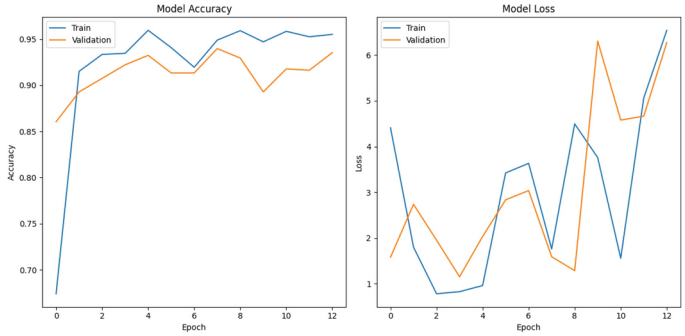
This section explores the impact of the basic HGS and EHGS algorithms on the hyperparameter tuning of the VGG16 model. The goal is to demonstrate the improvements in classification quality achieved through these optimization methods. The performance of HGS and EHGS is also compared with the traditional grid search method and Multi-Scale Delaunay Triangulation (msDT)[101], as previously applied in a published study using the same dataset. Experiments involving the HGS and EHGS algorithms were performed using consistent parameters: 20 iterations, a population size of 10, and 5 independent runs. These experimental parameters were



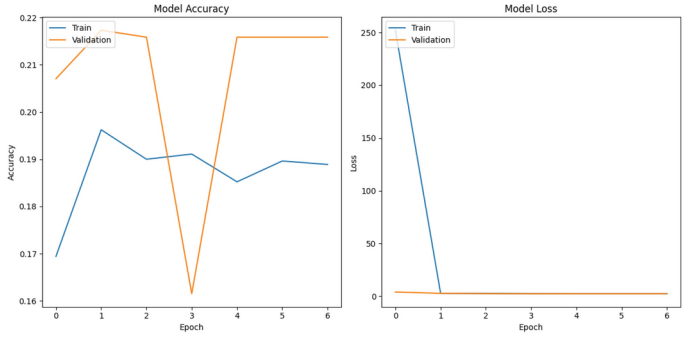
(a) VGG16



(b) DensNet121



(c) EfficientNetV0

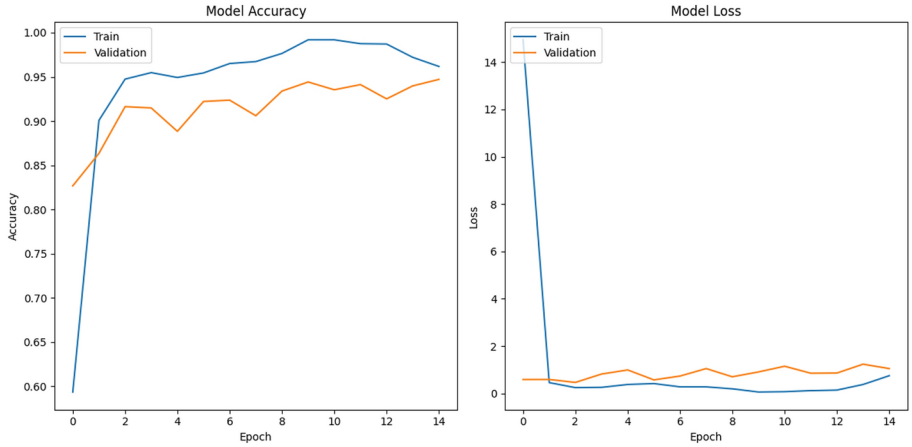


(d) InceptionV3

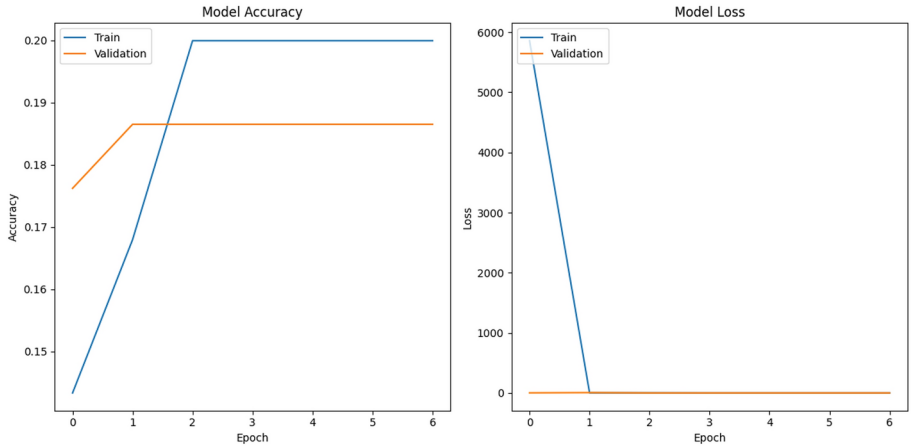
Fig. 9. Learning and validation performance trends for VGG16, DensNet121, EfficientNetV0, and InceptionV3.

carefully chosen to ensure a thorough exploration of optimal hyperparameters while balancing computational cost and overall performance.

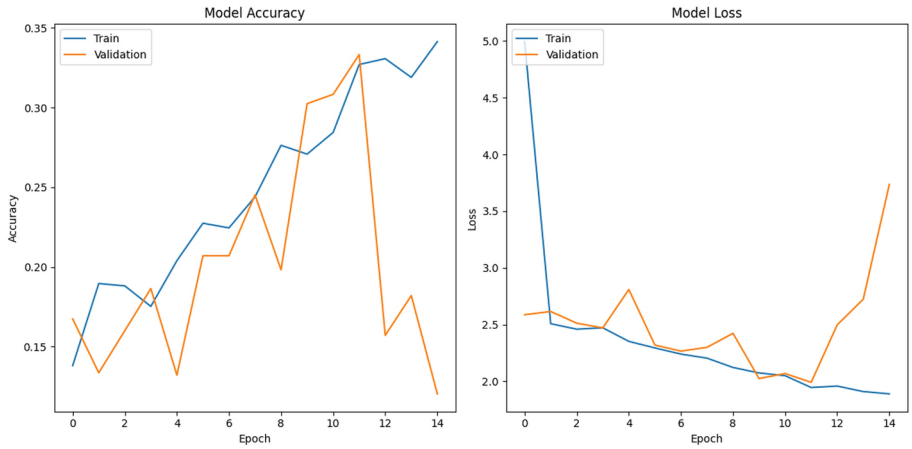
As shown in Table 12 and Fig. 11, the VGG16 model with predefined parameters achieved an accuracy of 95.70%, precision of 95.71%, and recall of 95.70%. Applying the traditional grid search for hyperparameter tuning yielded a slight improvement, with accuracy, precision, and recall increasing to 96.00%, 96.10%, and 96.00%, respectively. In contrast, the Multi-Scale Delaunay Triangulation approach resulted in significantly



(a) MobileNetV2



(b) ResNet



(c) VGG16-NP

Fig. 10. Learning and validation performance trends for MobileNetV2, ResNet, and VGG16-NP.

lower accuracy, reaching only 55%. By utilizing evolutionary optimization for hyperparameter tuning, the HGS algorithm offered further enhancements over both the predefined VGG16 model and traditional grid search, achieving accuracy, precision, and recall values of 96.50%, 96.60%, and 96.50%, respectively. EHGS scored the highest performance metrics, with an accuracy of 98.00%, precision of 98.10%, and recall of 98.00%. This superior performance is likely due to its advanced mechanisms, such as the ”local best” and ”local escaping mechanism,” which enhance both exploration and diversity, leading to better hyperparameter configurations.

Model	Accuracy	Precision	Recall
Predefined VGG16	0.9570	0.9571	0.9570
HGS_VGG16	0.9650	0.9660	0.9650
EHGS_VGG16	**0.9800**	**0.9810**	**0.9800**
Traditional Grid Search	0.960	0.961	0.960
multi-scale Delaunay triangulation[101]	0.55	-	-

Table 12. Performance metrics of VGG16 model with predefined parameters, VGG16 with traditional grid search, HGS-VGG16, EHGS-VGG16, and multi-scale delaunay triangulation. Significant values are in [bold].

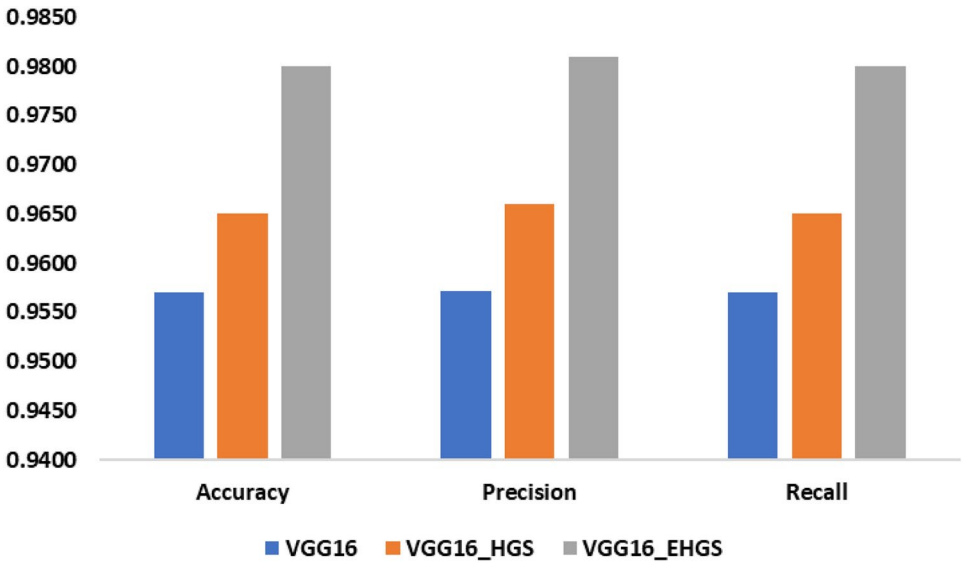


Fig. 11. Bar chart comparing the classification metrics for predefined parameters, Basic HGS, and EHGS as detailed in Table 12.

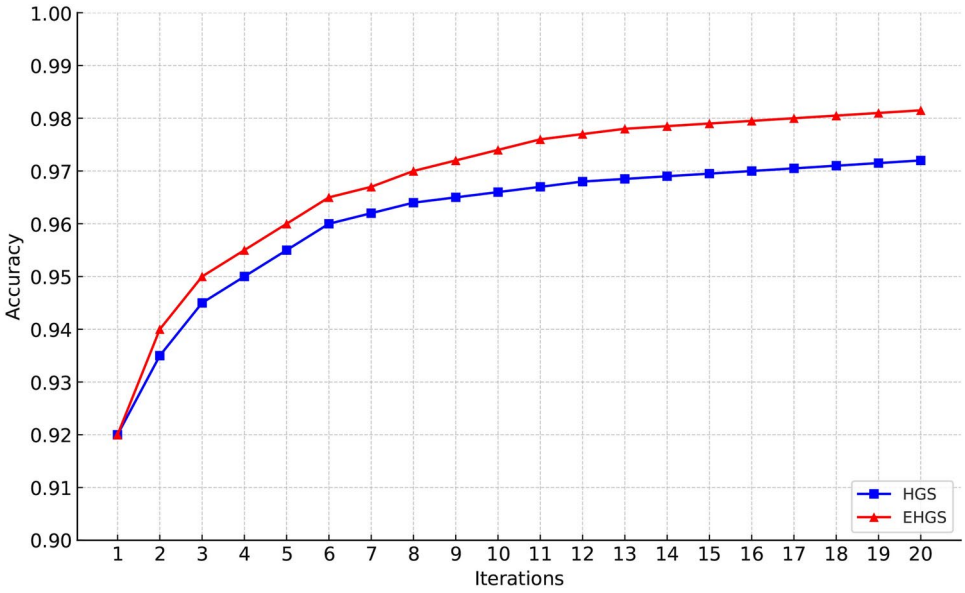


Fig. 12. Convergence curves for HGS and EHGS for hyperparameter tuning of the VGG16 model over 20 iterations, averaged across 5 runs.

These findings establish a new performance benchmark for the VGG16 model on this dataset and confirm the effectiveness of EHGS in achieving high classification quality.

The convergence curves for HGS and EHGS are depicted in Fig. 12. The curves indicate that both algorithms achieve rapid initial improvements in accuracy within the first few iterations. However, as the iterations progress, EHGS shows a slightly faster rate of convergence. This trend highlights the superior ability of the EHGS algorithm to avoid premature convergence and continue improving effectively. Through the final iterations, EHGS achieves a higher overall accuracy compared to HGS, confirming the benefits of its improved mechanisms in improving the hyperparameter tuning of the VGG16 model.

In conclusion, these results highlight the significant enhancement in performance of the VGG16 model on logo classification tasks by incorporating evolutionary optimization techniques, in particular Enhanced HGS. This hybrid approach effectively exploits the strengths of evolutionary and deep learning algorithms.

Conclusion and future works

This study presented an optimized deep learning framework, EHGS-VGG16, which leverages an enhanced variant of the Hunger Games Search (EHGS) to tune hyperparameters of the VGG16 model for logo classification tasks. Incorporating EHGS demonstrated significant improvements in model accuracy, precision, and recall. The evaluation process included three phases: testing the EHGS through the IEEE CEC2014 benchmark functions, evaluating the performance of the custom-developed VGG16 model against state-of-the-art deep learning models, and integrating the EHGS for hyperparameter optimization. The results showed that the EHGS-VGG16 model achieved a remarkable accuracy of 98%, with an increase of 3% compared to the basic predefined VGG16 model. Even a small increase in classification accuracy can lead to meaningful benefits in practical applications, such as more reliable brand monitoring, better copyright protection, and improved user engagement through targeted advertising.

Despite these promising results, the study has several limitations. Theoretically, while the EHGS algorithm performed well in benchmark tests, it still faces challenges like local optima entrapment and finding an optimal trade-off between exploration and exploitation, which can affect its efficiency in certain complex optimization tasks. From a practical perspective, the experiments were conducted using the Flickr-27 dataset, which is relatively small compared to other available datasets. As such, the results may not fully generalize to large-scale, real-world logo classification tasks with higher variability and noise. Furthermore, while EHGS enhances model accuracy, its iterative optimization process increases computational time compared to standard training procedures, which may limit its feasibility for time-sensitive applications.

Future work will focus on addressing these limitations. We plan to evaluate the EHGS-VGG16 framework on larger and more diverse datasets to ensure its robustness and generalizability. Some of these datasets include the Logo-2K+, FlickrLogos-32, and FlickrLogos-47. In addition, we will explore the application of the EHGS-VGG16 model to other image classification tasks, such as object detection and facial recognition, to extend its utility beyond logo classification. Moreover, we aim to investigate the integration of other advanced evolutionary algorithms, such as the Crow Search Algorithm (CSA) and Capuchin Search Algorithm (CapSA), to further optimize the hyperparameter tuning process and enhance the model's performance. Given the excellent performance of EHGS, it will be valuable to validate it to handle other challenging practical applications such as image segmentation and feature selection problems.

Data availability

The data involved in this study is public data, which can be downloaded through http://image.ntua.gr/iva/datasets/flickr_logos/

Received: 26 May 2024; Accepted: 2 December 2024
Published online: 30 December 2024

References

1. Bacanin, N., Bezdan, T., Tuba, E., Strumberger, I. & Tuba, M. Optimizing convolutional neural network hyperparameters by enhanced swarm intelligence metaheuristics. *Algorithms*[SPACE]https://doi.org/10.3390/a13030067 (2020).
2. Attri, I., Awasthi, L. K., Sharma, T. P. & Rathee, P. A review of deep learning techniques used in agriculture. *Eco. Inform.* **77**, 102217. https://doi.org/10.1016/j.ecoinf.2023.102217 (2023).
3. Jain, A., Ross, A. & Prabhakar, S. An introduction to biometric recognition. *IEEE Trans. Circuits Syst. Video Technol.* **14**, 4–20. https://doi.org/10.1109/TCSVT.2003.818349 (2004).
4. Litjens, G. et al. A survey on deep learning in medical image analysis. *Med. Image Anal.* **42**, 60–88. https://doi.org/10.1016/j.media.2017.07.005 (2017).
5. Fehérvái, I. & Appalaraju, S. Scalable logo recognition using proxies. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 715–725 (2019). https://doi.org/10.1109/WACV.2019.00081
6. Bianco, S., Buzzelli, M., Mazzini, D. & Schettini, R. Logo recognition using cnn features. In *In International Conference on Image Analysis and Processing* vol. 9280, 438–448. https://doi.org/10.1007/978-3-319-23234-8_41 (2015).
7. Gupta, M., Singhal, S., Nyamati, V. & Ramasamy, S. Logo infringement detection using machine learning. *Int. J. Sci. Res. Rev.* (2021).
8. Joly, A. & Buisson, O. Logo retrieval with a contrario visual query expansion. In *Proceedings of the 17th ACM International Conference on Multimedia*, MM '09, 581-584 (Association for Computing Machinery, 2009). https://doi.org/10.1145/1631272.1631361
9. Liu, L., Dzyabura, D. & Mizik, N. Visual Listening In: *Extracting Brand Image Portrayed on Social Media. Working Papers numberw0258, New Economic School (NES)* (2017). https://doi.org/10.1287/mksc.2020.1226.
10. Hagbi, N., Bergig, O., El-Sana, J. & Billinghurst, M. Shape recognition and pose estimation for mobile augmented reality. *IEEE Trans. Visual Comput. Graphics* **17**, 1369–1379. https://doi.org/10.1109/TVCG.2010.241 (2011).
11. Psyllos, A. P., Anagnostopoulos, C.-N.E. & Kayafas, E. Vehicle logo recognition using a sift-based enhanced matching scheme. *IEEE Trans. Intell. Transp. Syst.* **11**, 322–328. https://doi.org/10.1109/TITS.2010.2042714 (2010).

12. Alsheikhy, A., Said, Y. & Barr, M. Logo recognition with the use of deep convolutional neural networks. *Eng. Technol. Appl. Sci. Res.* **10**, 6191–6194. https://doi.org/10.48084/etasr.3734 (2020).
13. Sanghvi, J., Rathod, J., Nemade, S., Panchal, H. & Pavate, A. Logo detection using machine learning algorithm : A survey. In *2023 International Conference on Communication System, Computing and IT Applications (CSCITA)*, 136–141 (2023). https://doi.org/10.1109/CSCITA55725.2023.10105056
14. Bianco, S., Buzzelli, M., Mazzini, D. & Schettini, R. Deep learning for logo recognition. *Neurocomputing* **245**, 23–30. https://doi.org/10.1016/j.neucom.2017.03.051 (2017).
15. Hou, S. et al. Deep learning for logo detection: A survey. **2210**, 04399 (2022).
16. Alzubaidi, L. et al. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *J. Big Data*[SPACE]https://doi.org/10.1186/s40537-021-00444-8 (2021).
17. Alsajri, A. K. S. & Hacimahmud, A. V. Review of deep learning: Convolutional neural network algorithm. *Babylonian J. Mach. Learn.* **2023**, 19–25. https://doi.org/10.58496/BJML/2023/004 (2023).
18. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031 (2017).
19. Taigman, Y., Yang, M., Ranzato, M. & Wolf, L. Deepface: Closing the gap to human-level performance in face verification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1701–1708, https://doi.org/10.1109/CVPR.2014.220 (2014).
20. Karpathy, A. & Li, F. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Patt. Anal. Mach. Intell.*[SPACE]https://doi.org/10.1109/TPAMI.2016.2598339 (2014).
21. Farabet, C., Couprie, C., Najman, L. & LeCun, Y. Learning hierarchical features for scene labeling. *IEEE Trans. Pattern Anal. Mach. Intell.* **35**, 1915–1929. https://doi.org/10.1109/TPAMI.2012.231 (2013).
22. Lecun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324. https://doi.org/10.1109/5.726791 (1998).
23. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **60**, 84–90. https://doi.org/10.1145/3065386 (2017).
24. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations (ICLR 2015)*, 1–14 (Computational and Biological Learning Society, 2015).
25. Szegedy, C. *et al.* Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1–9 (IEEE Computer Society, 2015). https://doi.org/10.1109/CVPR.2015.7298594
26. He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
27. Huang, G., Liu, Z., Maaten, L. V. D. & Weinberger, K. Q. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269, (IEEE Computer Society, Los Alamitos, CA, USA, 2017). https://doi.org/10.1109/CVPR.2017.243
28. Hu, J., Shen, L. & Sun, G. Squeeze-and-excitation networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7132–7141 (2018). https://doi.org/10.1109/CVPR.2018.00745
29. Wang, Y., Zhang, H. & Zhang, G. cpso-cnn: An efficient pso-based algorithm for fine-tuning hyper-parameters of convolutional neural networks. *Swarm Evol. Comput.* **49**, 114–123. https://doi.org/10.1016/j.swevo.2019.06.002 (2019).
30. Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **13**, 281–305 (2012).
31. Talbi, E.-G. *Metaheuristics: from design to implementation* Vol. 74 (Wiley, NY, 2009).
32. Mafarja, M. M. & Mirjalili, S. Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing* **260**, 302–312 (2017).
33. Thaher, T., Sheta, A., Awad, M. & Aldasht, M. Enhanced variants of crow search algorithm boosted with cooperative based island model for global optimization. *Expert Syst. Appl.* **238**, 121712. https://doi.org/10.1016/j.eswa.2023.121712 (2024).
34. Mirjalili, S. & Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008 (2016).
35. Ab Wahab, M. N., Nefti-meziani, S. & Atyabi, A. A comprehensive review of swarm optimization algorithms. *PLoS ONE* **10**, e0122827. https://doi.org/10.1371/journal.pone.0122827 (2015).
36. Heidari, A. A. et al. Harris hawks optimization: Algorithm and applications. *Futur. Gener. Comput. Syst.* **97**, 849–872. https://doi.org/10.1016/j.future.2019.02.028 (2019).
37. Yang, Y., Chen, H., Heidari, A. A. & Gandomi, A. H. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Syst. Appl.* **177**, 114864. https://doi.org/10.1016/j.eswa.2021.114864 (2021).
38. Zhou, X. et al. Advanced orthogonal learning and Gaussian barebone hunger games for engineering design. *J. Comput. Design Eng.* **9**, 1699–1736. https://doi.org/10.1093/jcde/qwac075 (2022).
39. Qu, C. & Fu, Y. Crow search algorithm based on neighborhood search of non-inferior solution set. *IEEE Access* **7**, 52871–52895. https://doi.org/10.1109/ACCESS.2019.2911629 (2019).
40. Hou, L. et al. Image segmentation of intracerebral hemorrhage patients based on enhanced hunger games search optimizer. *Biomed. Signal Process. Control* **82**, 104511. https://doi.org/10.1016/j.bspc.2022.104511 (2023).
41. Wolpert, D. & Macready, W. No free lunch theorems for optimization. *Evolut. Comput. IEEE* **1**, 67–82 (1997).
42. Oliveira, G., FrazÃ£o, X., Pimentel, A. & Ribeiro, B. Automatic graphic logo detection via fast region-based convolutional networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, 985–991 (2016) https://doi.org/10.1109/IJCNN.2016.7727305.
43. Sahel, S., Alsahafi, M., Alghamdi, M. & Alsubait, T. Logo detection using deep learning with pretrained cnn models. *Eng. Technol. Appl. Sci. Res.* **11**, 6724–6729. https://doi.org/10.48084/etasr.3919 (2021).
44. Su, H., Zhu, X. & Gong, S. Deep learning logo detection with data expansion by synthesising context. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 530–539 (2017) https://doi.org/10.1109/WACV.2017.65
45. Yang, S., Zhang, J., Bo, C., Wang, M. & Chen, L. Fast vehicle logo detection in complex scenes. *Optics Laser Technol.* **110**, 196–201. https://doi.org/10.1016/j.optlastec.2018.08.007 (2019).
46. Eggert, C., Zecha, D., Brehm, S. & Lienhart, R. Improving small object proposals for company logo detection. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, ICMR '17, 167-174, https://doi.org/10.1145/3078971.3078990 (Association for Computing Machinery, 2017).
47. Bastan, M., Wu, H.-Y., Cao, T., Kota, B. & Tek, M. *Large scale open-set deep logo detection* Vol. 1911, 07440 (2022).
48. Jung, H. et al. Detection of masses in mammograms using a one-stage object detector based on a deep convolutional neural network. *PLoS ONE* **13**, e0203355 (2018).
49. Eggert, C., Brehm, S., Winschel, A., Zecha, D. & Lienhart, R. A closer look: Small object detection in faster r-cnn. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, 421–426 (2017). https://doi.org/10.1109/ICME.2017.8019550
50. Su, H., Gong, S. & Zhu, X. Scalable logo detection by self co-learning. *Pattern Recogn.* **97**, 107003. https://doi.org/10.1016/j.patcog.2019.107003 (2020).
51. Tüzkö, A., Herrmann, C., Manger, D. & Beyerer, J. *Open set logo detection and retrieval* vol. 1710, 10891 (2017).
52. Su, H., Gong, S. & Zhu, X. Weblogo-2m: Scalable logo detection by deep learning from the web. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 270–279 (2017). https://doi.org/10.1109/ICCVW.2017.41
53. Zhu, G. & Doermann, D. Automatic document logo detection. In *Document Analysis and Recognition, International Conference on* Vol. 2, 864–868. https://doi.org/10.1109/ICDAR.2007.68 (2007).

54. Hoi, S. C. H. et al. Logo-net: Large-scale deep logo detection and brand recognition with deep region-based convolutional networks (2015). 1511.02462.
55. Jaeger, P. F. et al. *Retina u-net: Embarrassingly simple exploitation of segmentation supervision for medical object detection* **1811**, 08661 (2018).
56. Sawan, A., Awad, M., Qasrawi, R. & Sowan, M. Hybrid deep learning and metaheuristic model based stroke diagnosis system using electroencephalogram (eeg). *Biomed. Signal Process. Control* **87**, 105454. https://doi.org/10.1016/j.bspc.2023.105454 (2024).
57. Rere, L. M. R., Fanany, M. I. & Arymurthy, A. M. Metaheuristic algorithms for convolution neural network. *Comput. Intell. Neurosci.* **2016**, 1537325 (2016).
58. Yamasaki, T., Honma, T. & Aizawa, K. Efficient optimization of convolutional neural networks using particle swarm optimization. In *2017 IEEE Third International Conference on Multimedia Big Data (BigMM)*, 70–73 (2017) https://doi.org/10.1109/BigMM.2017.69
59. Sinha, T., Haidar, A. & Verma, B. Particle swarm optimization based approach for finding optimal values of convolutional neural network parameters. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, 1–6 (2018) https://doi.org/10.1109/CEC.2018.8477728
60. Hu, T. et al. Real-time covid-19 diagnosis from x-ray images using deep cnn and extreme learning machines stabilized by chimp optimization algorithm. *Biomed. Signal Process. Control* **68**, 102764. https://doi.org/10.1016/j.bspc.2021.102764 (2021).
61. Cai, C. et al. Improved deep convolutional neural networks using chimp optimization algorithm for covid19 diagnosis from the x-ray images. *Expert Syst. Appl.* **213**, 119206. https://doi.org/10.1016/j.eswa.2022.119206 (2023).
62. Wang, X., Gong, C., Khishe, M., Mohammadi, M. & Rashid, T. A. Pulmonary diffuse airspace opacities diagnosis from chest x-ray images using deep convolutional neural networks fine-tuned by whale optimizer. *Wirel. Pers. Commun.* **124**, 1355–1374. https://doi.org/10.1007/s11277-021-09410-2 (2022).
63. Khishe, M., Caraffini, F. & Kuhn, S. Evolving deep learning convolutional neural networks for early covid-19 detection in chest x-ray images. *Mathematics*[SPACE]https://doi.org/10.3390/math9091002 (2021).
64. Leung, S., Tang, Y. & Wong, W. A hybrid particle swarm optimization and its application in neural networks. *Expert Syst. Appl.* **39**, 395–405. https://doi.org/10.1016/j.eswa.2011.07.028 (2012).
65. Darwish, A., Ezzat, D. & Hassanien, A. E. An optimized model based on convolutional neural networks and orthogonal learning particle swarm optimization algorithm for plant diseases diagnosis. *Swarm Evol. Comput.* **52**, 100616. https://doi.org/10.1016/j.swevo.2019.100616 (2020).
66. Xin, J., Khishe, M., Zeebaree, D. Q., Abualigah, L. & Ghazal, T. M. Adaptive habitat biogeography-based optimizer for optimizing deep cnn hyperparameters in image classification. *Heliyon* **10**, e28147. https://doi.org/10.1016/j.heliyon.2024.e28147 (2024).
67. Saffari, A., Khishe, M., Mohammadi, M., Hussein Mohammed, A. & Rashidi, S. Dcnn-fuzzywoa: Artificial intelligence solution for automatic detection of Covid-19 using x-ray images. *Comput. Intell. Neurosci.* **2022**, 5677961. https://doi.org/10.1155/2022/5677961 (2022).
68. Yutong, G., Khishe, M., Mohammadi, M., Rashidi, S. & Nateri, M. Evolving deep convolutional neural networks by extreme learning machine and fuzzy slime mould optimizer for real-time sonar image recognition. *Int. J. Fuzzy Syst.* **24**, 1371–1389. https://doi.org/10.1007/s40815-021-01195-7 (2021).
69. Khishe, M., Mohammadi, M., Rashid, T. A., Mahmud, H. & Mirjalili, S. *Evolving deep neural network by customized moth flame optimization algorithm for underwater targets recognition* **2303**, 00922 (2023).
70. Mohammad Khishe, M. M. & Mohammed, A. H. Complex active sonar targets recognition using variable length deep convolutional neural network evolved by biogeography-based optimizer. *Waves Random Complex Med.*[SPACE]https://doi.org/10.1080/17455030.2022.2155319 (2022).
71. Khishe, M. Variable-length deep convolutional neural networks by internet protocol addresses whale optimization algorithm for random and complex image classification. *Waves Random Complex Med.*[SPACE]https://doi.org/10.1080/17455030.2022.2164377 (2023).
72. Azhdari, S. M. H., Mahmoodzadeh, A., Khishe, M. & Agahi, H. Pulse repetition interval modulation recognition using deep cnn evolved by extreme learning machines and ip-based bbo algorithm. *Eng. Appl. Artif. Intell.* **123**, 106415. https://doi.org/10.1016/j.engappai.2023.106415 (2023).
73. Bacanin, N. et al. Performance of a novel chaotic firefly algorithm with enhanced exploration for tackling global optimization problems: Application for dropout regularization. *Mathematics*[SPACE]https://doi.org/10.3390/math9212705 (2021).
74. Raiaan, M. A. K. et al. A systematic review of hyperparameter optimization techniques in convolutional neural networks. *Decis. Anal. J.* **11**, 100470. https://doi.org/10.1016/j.dajour.2024.100470 (2024).
75. Malakar, S., Ghosh, M., Bhowmik, S., Sarkar, R. & Nasipuri, M. A ga based hierarchical feature selection approach for handwritten word recognition. *Neural Comput. Appl.* **32**, 2533–2552. https://doi.org/10.1007/s00521-018-3937-8 (2020).
76. Zivkovic, M. et al. Novel hybrid firefly algorithm: an application to enhance xgboost tuning for intrusion detection classification. *PeerJ Comput. Sci.* **8**, e956. https://doi.org/10.7717/peerj-cs.956 (2022).
77. Dobrojevic, M. et al. Addressing internet of things security by enhanced sine cosine metaheuristics tuned hybrid machine learning model and results interpretation based on shap approach. *PeerJ Comput. Sci.* **9**, e1405. https://doi.org/10.7717/peerj-cs.1405 (2023).
78. Jovanovic, L. et al. Tackling iot security challenge by metaheuristics tuned extreme learning machine. In *Intelligent Sustainable Systems* (eds Raj, J. S. et al.) 507–522 (Springer, 2023).
79. Fahim, S. R. et al. Parameter identification of proton exchange membrane fuel cell based on hunger games search algorithm. *Energies*[SPACE]https://doi.org/10.3390/en14165022 (2021).
80. Nguyen, H. & Bui, X.-N. A novel hunger games search optimization-based artificial neural network for predicting ground vibration intensity induced by mine blasting. *Nat. Resour. Res.* **30**, 3865–3880. https://doi.org/10.1007/s11053-021-09903-8 (2021).
81. Shaker, Y. O. et al. Optimal charging/discharging decision of energy storage community in grid-connected microgrid using multi-objective hunger game search optimizer. *IEEE Access* **9**, 120774–120794. https://doi.org/10.1109/ACCESS.2021.3101839 (2021).
82. AbuShanab, W. S., Abd Elaziz, M., Ghandourah, E. I., Moustafa, E. B. & Elsheikh, A. H. A new fine-tuned random vector functional link model using hunger games search optimizer for modeling friction stir welding process of polymeric materials. *J. Mater. Res. Technol.* **14**, 1482–1493. https://doi.org/10.1016/j.jmrt.2021.07.031 (2021).
83. Chakraborty, S., Saha, A. K., Chakraborty, R., Saha, M. & Nama, S. Hswoa: An ensemble of hunger games search and whale optimization algorithm for global optimization. *Int. J. Intell. Syst.*[SPACE]https://doi.org/10.1002/int.22617 (2021).
84. Kutlu Onay, F. & Aydemir, S. Chaotic hunger games search optimization algorithm for global optimization and engineering problems. *Math. Comput. Simul.* **192**, 514–536. https://doi.org/10.1016/j.matcom.2021.09.014 (2021).
85. Li, S. *et al.* Incorporation of improved differential evolution into hunger games search algorithm. In *2021 13th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 39–43, https://doi.org/10.1109/IHMSC52134.2021.00017 (2021).
86. Mahajan, S., Abualigah, L. & Pandit, A. Hybrid arithmetic optimization algorithm with hunger games search for global optimization. *Multimed. Tools Appl.* **81**, 28755–28778. https://doi.org/10.1007/s11042-022-12922-z (2022).
87. Ma, B. J., Liu, S. & Heidari, A. A. Multi-strategy ensemble binary hunger games search for feature selection. *Knowl. Based Syst.* **248**, 108787. https://doi.org/10.1016/j.knosys.2022.108787 (2022).

88. Xu, B. *et al.* Quantum nelder-mead hunger games search for optimizing photovoltaic solar cells. *International Journal of Energy Research***https://aliasgharheidari.com/HGS.html**, https://doi.org/10.1002/er.8011 (2022).
89. Real, L. A. Animal choice behavior and the evolution of cognitive architecture. *Science* **253**, 980–986 (1991).
90. Burnett, C. et al. Hunger-driven motivational state competition. *Neuron* **92**, 187–201. https://doi.org/10.1016/j.neuron.2016.08.032 (2016).
91. Sutton, A. K. & Krashes, M. J. Integrating hunger with rival motivations. *Trends Endocrinol. Metab.* **31**, 495–507. https://doi.org/10.1016/j.tem.2020.04.006 (2020).
92. Clutton-Brock, T. Cooperation between non-kin in animal societies. *Nature* **462**, 51–57. https://doi.org/10.1038/nature08366 (2009).
93. Friedman, M., Ulrich, P. & Mattes, R. A figurative measure of subjective hunger sensations. *Appetite* **32**, 395–404. https://doi.org/10.1006/appe.1999.0230 (1999).
94. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Bengio, Y. & LeCun, Y. (eds.) *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015).
95. Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* Vol. 20 (eds Pereira, F. et al.) (Curran Associates Inc., 2012).
96. Chen, Y. et al. Vgg16-based intelligent image analysis in the pathological diagnosis of IGA nephropathy. *J. Radiat. Res. Appl. Sci.* **16**, 100626. https://doi.org/10.1016/j.jrras.2023.100626 (2023).
97. Sharma, S., Guleria, K., Tiwari, S. & Kumar, S. A deep learning based convolutional neural network model with vgg16 feature extractor for the detection of alzheimer disease using mri scans. *Measure. Sens.* **24**, 100506. https://doi.org/10.1016/j.measen.2022.100506 (2022).
98. Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T. & Smeulders, A. W. M. Selective search for object recognition. *Int. J. Comput. Vision* **104**, 154–171 (2013).
99. van de Sande, K. E. A., Uijlings, J. R. R., Gevers, T. & Smeulders, A. W. M. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, 1879–1886 (2011). https://doi.org/10.1109/ICCV.2011.6126456
100. Rezatofighi, H. *et al.* Generalized intersection over union: A metric and a loss for bounding box regression. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 658–666 (IEEE Computer Society, Los Alamitos, 2019). https://doi.org/10.1109/CVPR.2019.00075
101. Kalantidis, Y., Pueyo, L. G., Trevisiol, M., van Zwol, R. & Avrithis, Y. Scalable triangulation-based logo recognition. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, ICMR '11, (Association for Computing Machinery, 2011). https://doi.org/10.1145/1991996.1992016
102. Deng, J. *et al.* Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255, https://doi.org/10.1109/CVPR.2009.5206848 (2009).
103. Wang, H., Rahnamayan, S., Sun, H. & Omran, M. G. H. Gaussian bare-bones differential evolution. *IEEE Trans. Cybern.* **43**, 634–647. https://doi.org/10.1109/TSMCB.2012.2213808 (2013).
104. Liang, J., Qu, B. & Suganthan, P. Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. Tech. Rep., Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Technical Report, Nanyang Technological University, Singapore (201311).
105. Derrac, J. & García, S., Molina, D. & Herrera, F.,. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **1**, 3–18 (2011).
106. Google. Google colaboratory (2023).
107. Yang, X.-S. & Gandomi, A. Bat algorithm: A novel approach for global engineering optimization. *Eng. Comput.* **29**, 464–483. https://doi.org/10.1108/02644401211235834 (2012).
108. Mirjalili, S. Sca: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **96**, 120–133. https://doi.org/10.1016/j.knosys.2015.12.022 (2016).

Acknowledgments

This research was funded by Zayed University Research Incentive Fund (Grant No. R29103).

Author contributions

T.T.: Methodology, Formal Analysis, Software, Validation, Writing—Original Draft, Writing - Review and Editing. M.H.: Project Investigation, Conceptualization, Supervision, Methodology, Formal Analysis, Validation, Writing—Review and Editing. B.A.: Supervision, Writing—Review and Editing. M.M.: Supervision, Methodology, Validation, Writing—Review and Editing. All authors have read and approved the final manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Ethical statement

The manuscript has not been submitted to more than one journal for simultaneous consideration and has not been published elsewhere in any form or language.

Additional information

Correspondence and requests for materials should be addressed to T.T.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.