Received July 7, 2020, accepted July 19, 2020, date of publication July 23, 2020, date of current version August 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011502

A Deep Learning Model Based on Multi-Objective Particle Swarm Optimization for Scene Classification in Unmanned Aerial Vehicles

AGHILA RAJAGOPAL[1,*], GYANENDRA PRASAD JOSHI[2,*], A. RAMACHANDRAN[3], R. T. SUBHALAKSHMI[4], MANJU KHARI[5], SUDAN JHA[6], K. SHANKAR[7], (Senior Member, IEEE), AND JINSANG YOU[8], (Member, IEEE)

[1]Department of IT, Sethu Institute of Technology, Kariapatti 626115, India
[2]Department of Computer Science and Engineering, Sejong University, Seoul 05006, South Korea
[3]University College of Engineering, Panruti 607106, India
[4]Department of Information Technology, Sethu Institute of Technology, Kariapatti 626115, India
[5]Computer Science Department, Ambedkar Institute of Technology, New Delhi, Delhi 110092, India
[6]School of Computer Science and Engineering, Lovely Professional University, Phagwara 144411, India
[7]Department of Computer Applications, Karaikudi 630003, India
[8]Seculayer Company, Ltd., Seoul 04784, South Korea

Corresponding authors: Gyanendra Prasad Joshi (joshi@sejong.ac.kr) and Jinsang You (js.yu@seculayer.com)

([*]Aghila Rajagopal and Gyanendra Prasad Joshi contributed equally to this work.)

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea Government (MSIP) (No. 2020-0-00107, Development of the technology to automate the recommendations for big data analytic models that define data characteristics and problems).

ABSTRACT Recently, the increase in inexpensive and compact unmanned aerial vehicles (UAVs) and lightweight imaging sensors has led to an interest in using them in various remote sensing applications. The processes of collecting, calibrating, registering, and processing data from miniature UAVs and interpreting the data semantically are time-consuming. In UAV aerial imagery, learning effective image representations is central to the scene classification process. Earlier approaches to the scene classification process depended on feature coding methods with low-level hand-engineered features or unsupervised feature learning. These methods could produce mid-level image features with restricted representational abilities, which generally yielded mediocre results. The development of convolutional neural networks (CNNs) has made image classification more efficient. Due to the limited resources in UAVs, it is hard to fine-tune the hyperparameters and the trade-offs between classifier results and computation complexity. This paper introduces a new multi-objective optimization model for evolving state-of-the-art deep CNNs for scene classification, which generates the non-dominant solutions in an automated way at the Pareto front. We use a set of two benchmark datasets to test the performance of the scene classification model and make a detailed comparative study. The proposed method attains a very low computational time of 80 sec and maximum accuracy of 97.88% compared to all other methods. The proposed method is found to be appropriate for the effective scene classification of images captured by UAVs.

INDEX TERMS Unmanned aerial vehicle, particle swarm optimization, deep learning, convolutional neural networks, machine learning, internet of everything, aerial images, smart environment.

I. INTRODUCTION

Recently there has been a lot of interest in autonomous unmanned aerial vehicles (UAVs) and their applications, which include search-and-rescue, surveillance and reconnaissance, and examination of infrastructure [1], [2]. Landcover classification is a significant element of UAV applications, and it is difficult to build entirely autonomous systems. The object detection task is highly composite, and the demands for reduced costs are also challenging. Owing to the motion of UAV, the images are blurred, with noisy frequencies, as the on-board cameras frequently produce low-resolution images. In numerous applications of UAVs, the detection task is highly complex due to the requirement

The associate editor coordinating the review of this manuscript and approving it for publication was Moayad Aloqaily.

for practical performances. There has been a lot of research on UAVs dealing with attempts to track and find particular objects, kinds as vehicles, landmarks, landing sites, and people (including moving pedestrians) [3]. However, there exist some works which assume multiple object detection [4] because multiple target object detection is important for numerous UAV applications. The gap between the requirements of the application and the technical abilities exists because of two crucial constraints: (i) it is complex to construct and save various models of target objects and (ii) huge computing power is required for practical object detection even for a single object.

Aerial imagery classification of scenes categorizes the derived aerial images into subregions by masking numerous ground objects and kinds of landcover into various semantic types. And, for numerous real-time applications of remote sensing, like urban planning, computer cartography, and management of resources, aerial image classification is highly significant [5]. Commonly, a few similar object classes or kinds of land cover are shared between various types of scenes. For instance, residential and commercial are two major types of scenes that might include trees, buildings, and roads; however, there are differences in the spatial distribution and density of the three classes. Therefore, in aerial scenes, classification based on structural and spatial pattern complexity is a difficult problem. The usual approach is to build a holistic scene representation for the classification of the scene. In the remote sensing community, one well-known method for scene classification problem solving is bag-of-visual-words (BoVW). It was created for analysis of the text that designs a document through the frequency of words. To recognize images through the number of occurrences of ''visual words'' that were generated through local feature quantizing, the bag of words (BOW) model is used with a clustering technique. BoVW model is a version of BoW model to image analysis, where every image is defined as an orderless et of visual words from a visual dictionary by a histogram of the visual words [6], [7].

Spatial information is disregarded by assuming the representation of the BOW, using numerous variant techniques [7]–[9], depending on the BOW model that had been built for enhancing the capability to demonstrate the local feature spatial relationships. The performance of BOW-based techniques depends on the handcrafted local feature extraction, for instance, the color histogram [10], texture features, and local structural points [11]. The process of unsupervised feature learning (UFL) is used by a few researchers [12]. It is used to identify appropriate internal features in a huge amount of unlabeled data through certain unsupervised learning techniques instead of engineered features, achieving superior aerial scene classification. Though it seems that the aerial scene classification performance has only been slightly enhanced through an effective model of minor variants, that is mainly due to the inefficient traditional techniques of producing powerful feature representations sufficient for aerial scenes. The UFL and BOW techniques produce feature representations in the middle format for a definite limit. Hence, highly representative abstractions are needed for classifying the scene, which would also maximize low-level features, and which might demonstrate high discrimination.

Deep learning (DL) techniques [13], [14] have been very useful for solving conventional problems like object detection and recognition, natural language processing, and speech recognition, and also in numerous real-time applications. In many fields, this technique is much more effective than standard procedures, and it has attracted much interest in industrial and educational groups. The method of deep learning tries to derive common hierarchical feature learning with respect to various abstraction levels. Deep convolutional neural networks (CNNs) [7], [15] are the most common method of deep learning. Today, this technique is well-known and very successful in many detection and recognition tasks, yielding better results than a count of standard datasets.

In image classification, CNNs have been popular in recent years because they allow superior classification accuracy [34]–[38]. For industrial use, it is hard to adapt the standard deep CNNs because of the complexity of manually fine-tuning the hyperparameters and trade-offs among computational cost and classification accuracy. There has been researched on minimizing the computational cost [16]. In UAV aerial scene classification, complexity of standard CNNs has been reduced [17]. A specific kind of CNN structure is selected to reduce the search space, and a small search space is formulated using special expert domain knowledge.

Keeping in mind the issues that exist in the earlier models, the objective of this paper is to derive an effective scene classification model based on the optimized DL model. This work introduces a new multi-objective particle swarm optimization (MOPSO) model for evolving state-of-the-art deep CNNs in scene classification that generates the non-dominant solutions in an automated way at the Pareto front. This method helps to achieve a trade-off between the inference latency and classification accuracy, known as multiobjective CNN (MOCNN). For superior performance in UAVs, the CNN hyperparameters are adjusted automatically by MOCNN and placed into a trained model. The MOPSO technique would be built to find Pareto front model specifics. The validation of the presented model takes place in two open access datasets, the UC Merced (UCM) Land Use Dataset and the WHU-RS Dataset.

The paper contributions are listed as follows.

- Develop an effective scene classification model based on the optimized DL model
- Design a new multi-objective particle swarm optimization (MOPSO) model for scene classification
- Perform CNN hyperparameters are adjusted automatically by MOCNN algorithm
- The MOPSO technique would be built to find Pareto front model specifics
- Validate the experimental results against two open access datasets namely UCM and WHU-RS dataset.

The paper is structured as follows. A few standard techniques of scene classification are reviewed in Section 2. The proposed model is presented in Section 3. Section 4 summarizes the experimental outcomes. Finally, Section 5 presents the conclusions.

II. RELATED WORK

Research has been done on classifying the scenes captured by UAV. BOW is one model that is frequently used for scene classification. Initially, the local features of an image, such as scale-invariant feature transform (SIFT) parameters, are extracted and encoded toward the adjacent visible word. Hence, the last image representation is considered to be a histogram in which all bins count the incidence frequency for local features. Numerous studies have proposed improvised versions of BOW based on specific features of aerial scenes. Al-Turjman *et al.* [18] introduced the spatial co-occurrence kernel (SCK), which explains the spatial distribution of visual words. Ye *et al.* [19] introduced a conversion, as well as a rotation-invariant pyramid-of-spatial-relations (PSR) method to explain relative and absolute spatial relationships of local features. Zhao *et al.* [8] devised a concentric-circle-structured multi-scale BOW (CCM-BOW) technique, which is used to attain rotation invariance. Next, Cheng *et al.* [20] introduced a rotation-invariant model by applying collections of part detectors (COPD), which acquire the visual portions of images. Though they achieve superior performance, they are generally the extended versions of the traditional BOW model, and it is hard to improve the results because of the restricted representative's ability of the low-and mid-level features.

Unsupervised feature learning (UFL) is popular among the researchers and machine learning (ML) communities. The UFL method learns the features from a massive number of unlabeled samples in an automated way. It can identify the most relevant data captured by the corresponding information. Cheriyadat [21] used a sparse coding method to understand the sparse local features from image scenes and to pool local features to produce the image representation. Zhang *et al.* [22] make use of a traditional neural network (NN) known as a sparse autoencoder, which trains a set of selected image patches and which is tested using the saliency degree to filter local features. Coates *et al.* [12] enhanced the traditional UFL pipeline using the learned features. Though the UFL methods are handcrafted features, there is room for improvement because of the shallow learning architectures.

The familiarity of CNN is useful in diverse application areas. LeCun *et al.* [14] started a model for training the CNN structure using the backpropagation technique and attained satisfactory performance in character recognition. Recently, CNN has been frequently used in the computer vision community. At the same time, it is difficult to provide training for a deep CNN that has a large number of attributes, such as is often used for specific tasks, using less amount of training samples. Research has been done to extract the intermittent features from deep CNNs that underwent training on adequately large-scale datasets, like ImageNet, which is used for a broader view of visual recognition processes, namely scene classification [13], object identification, and image retrieval.

Roughly, all experiments are done with CNN activations from fully connected layers and with features from the convolutional layer, which is often overlooked. Cimpoi *et al.* [23] reported better results with examining texture through pooling CNN attributes derived from convolutional layers and the Fisher coding process. The use of CNNs for scene classification for UAVs is yet to be investigated in detail. In [24], a pre-trained CNN is applied and tuned thoroughly on a scene dataset exhibiting superior classifier results, whereas the pretrained CNN model is transferred to scene datasets lacking training modalities. In [25], the generalized potential of CNN features derived from fully connected layers underwent testing while classifying remote sensing images, and offered better results than the comparable methods in the open access scene dataset.

Though several methods for UAV image classification have been available in the literature, there is still a need to improvise the classification performance. At the same time, some of the methods have offered better results on a particular dataset and have not been applied on a large dataset. Therefore, in this study, a new UAV image classification model using MOPSO has been developed.

III. THE PROPOSED METHOD

This paper proposes a technique of the MOPSO algorithm to manage a trade-off between the inference latency and classification accuracy that is known as MOCNN. For superior performance in UAVs, the CNN hyperparameters are adjusted automatically by MOCNN and placed into a trained model. A MOPSO technique would be created to find Pareto front model specifics. Hence, depending on target devices and the image classification task, a highly appropriate model for a UAV is derived. The entire procedure involved in this work is shown in Figure 1. As shown in the figure, the UAV captures the videos, which are then divided into a set of frames. Then preprocessing of the video frames is done, allowing further processing. Afterward, training takes place using optimal multi-objective particle swarm optimization (OMOPSO) with CNN. Once the training process is completed, a model is derived. During the testing phase, the CNN-based model is executed, and the provided input image undergoes effective scene classification.

A. PREPROCESSING

The scenes captured by the image sensors in a UAV might be not clear in some cases. If the color of an object is variable, image segmentation techniques allow the frame to be segmented into target regions and other objects. Through the subsequent steps, the possible object areas undergo additional processing. This phase allows the frame region to be processed, and, in a few instances, to filter or skip the frames with no possible object areas. Hence, this method offers superior

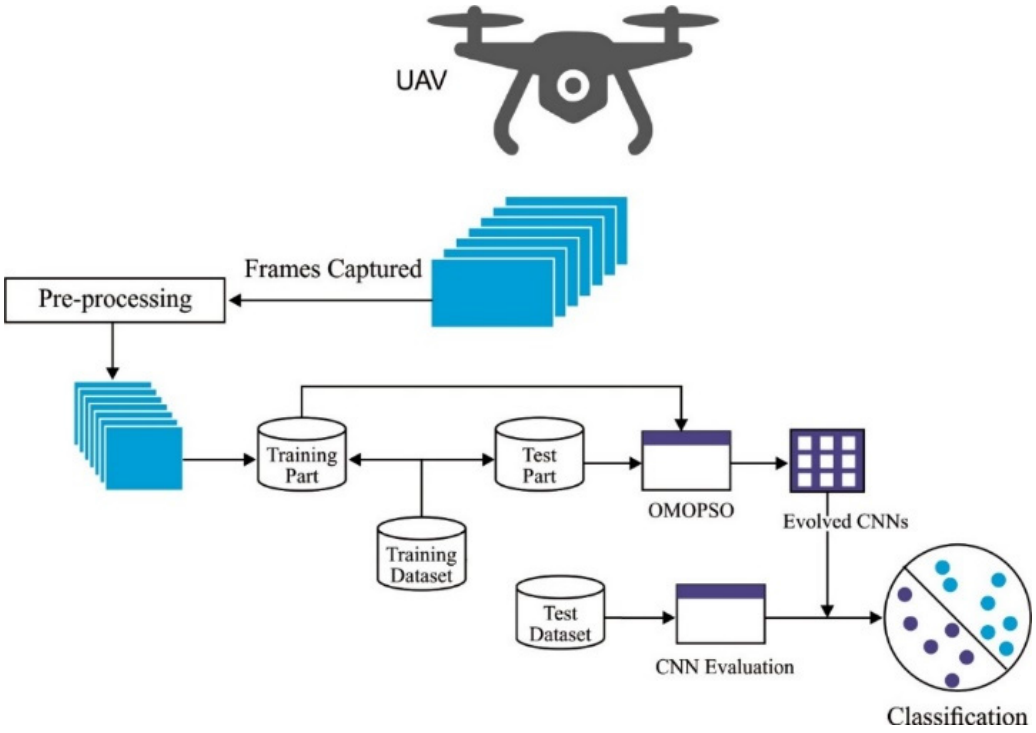

FIGURE 1. The overall process of the proposed method.

object localization and minimizes computation time. With a sliding window, in the HSV color space, a frame might be scanned in the preprocessing phase, and every window would be checked through saturation window component thresholding $th_{sat}\,(V)$. The thresholding is done according to the following equation:

$$th_{sat}\,(V) = 1.0 - \frac{0.8V}{255} \tag{1}$$

where V is the intensity component rate. When the saturation component rate is higher than or equal to $th_{sat}(V)$, the pixels are determined with respect to that object. In this case, the window does contain an object.

B. PROPOSED IMAGE CLASSIFICATION ALGORITHM

Once the images are preprocessed in the previous stage, the classification process will take place using MOCNN model. The proposed MOCNN framework consists of three phases. The first phase is initializing the defined population based on the proposed particle encoding scheme. For two objective optimizations, the multi-objective PSO technique known as OMOPSO [26] is used in the subsequent phase. Finally, in the Pareto set, the non-dominant solutions are derived from the original CNNs; the user may select one based on what is needed. The entire system structure is shown in Figure 1. The dataset is divided into a test set and a training set, and the set subjected to training is further segmented into test and training parts. The test and training parts are then subjected to the proposed OMOPSO technique. While the objective is evaluated, the testing part is used to improve testing accuracy, and the training part is used to train the NN that is used as the classification accuracy objective. Non-dominant solutions are created through the proposed OMOPSO technique that is used to optimize CNN frameworks. For actual use, one non-dominant solution might be chosen based on the trade-off between the hardware resource ability and classification accuracy. For the chosen CNN framework, the evaluation requires fine-tuning, and the entire test and training sets are used to derive the end classification accuracy.

1) THE PSO ALGORITHM

Initially, PSO had evolved from swarm behavior, like in fish schools and bird flocks. PSO received increasingly more attention and became a wide field of study known as swarm intelligence. This technique is used to search the space of an objective function by modifying the trajectories of single agents, termed particles, which have piecewise paths generated by a positional vector in a quasi-stochastic manner.

The trend of a swarming particle has two main units: a stochastic unit and a deterministic unit. Every particle is based on the position of the current global best g^* and the corresponding best location x_l^*, while it includes a random movement. Assume that x_l and v_l are the vectors of, respectively, position and velocity of particle l. A new velocity vector is computed using the function below [13].

$$v_l^{z+1} = v_l^z + \alpha\varepsilon_1\left[g^* - x_l^z\right] + \beta\varepsilon_2\left[x_l^* - x_l^z\right] \tag{2}$$

where ε_1 and ε_2 are random vectors, and all entries are assigned values from 0 and 1. The parameters α and β are

referred to as learning measures and can be represented as $\alpha \approx \beta \approx 2$.

The initial position of every particle is shared in an identical fashion, which can be sampled in several regions. This is considered vital in several issues. The primary velocity of a particle is assigned to 0, that is, $v_l^{z=0} = 0$. Hence, the new positions are found by

$$x_l^{z+1} = x_l^z + v_l^{z+1} \tag{3}$$

Though v_l has different values, it is often bounded by $[0; v_{\max}]$.

There are several well-known variants that expand on the PSO technique, and a useful adjustment is to exploit the inertia function $\theta(z)$, so that v_l^z is substituted by $\theta(z)v_l^z$:

$$v_l^{z+1} = \theta v_l^z + \alpha\varepsilon_1\left[g^* - x_l^z\right] + \beta\varepsilon_2 \odot \left[x_l^* - x_l^z\right] \tag{4}$$

where θ consumes the values from 0 and 1. In a simple case, the inertia function is considered a constant, generally $\theta \approx 0.5 \sim 0.9$. It is used both to establish the virtual mass and to retain the movement of the particles, and it converges rapidly.

2) PARTICLE ENCODING STRATEGY

The hyperparameters that need to be optimized in the DenseNet are the block count and the layer count in every block, and every block's growth rate. A vector with two lengths can represent the growth rate and layer count for every block. With the block count × fixed length of 2, the growth rate and layer count in every block might be encoded when the block count is described. Vector examples that carry out the DenseNet hyperparameters with 3 blocks are shown in Figure 2.

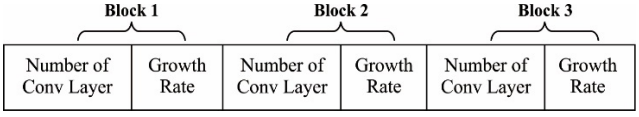

FIGURE 2. An example of a particle vector.

The block count needs to be configured initially as determined through the proposed encoding scheme. This gives two benefits. Initially, with fixed dimensions, the OMOPSO might be effectively working over a consequent search space; the hyperparameters of the DenseNet are encoded into fixed-length vectors after fixing the block counts wherever OMOPSO might be used. In the search space, when the *i*th particle block shifts to the optimal position while carrying out the OMOPSO evolutionary operators, the block count is fixed. When carrying out OMOPSO, one method to resolve the issue is to combine the hyperparameters to create fixed-length particles. When the block counts are not fixed, that might create many disturbances within the search space through the concept of breaking every block to the best location. Another way is to shift the matched blocks to their best location; this reduces the flying particles by maintaining a few blocks in previous positions. Hence, the simpler and more efficient solution is to fix the block counts through the proposed encoding method.

3) POPULATION INITIALIZATION

Depending on the hardware resource capability and network efficiency, every dimensional range needs to perform initially before population initialization. When a layer count within a block is small—for instance, when the layer count is smaller than 2—there would be connections of shortcuts constructed within a dense block, and smaller feature map counts, that is, a smaller growth rate would not be created either efficient feature maps. At the same time, when the growth rate and layer count are high, the hardware resources needed to execute the experiment might exceed the actual hardware capacity. For every dimension, the particular range of the experiment can be modeled. The primary population is produced randomly based on the dimensional range. A random value is generated between the primary dimension and the end dimension. The entire primary population with fixed size is produced through the individual generation procedure, repeating until the size of the population is satisfactory.

4) OBJECTIVE EVALUATION

In MOCNN objective evaluation, when the proposed MOCNN simultaneously optimizes classification accuracy, it is estimated and returned as individual objectives. The training dataset is divided into two segments, the test part and the training part, deriving the classification accuracy prior to individual training using a DenseNet with certain hyperparameters, and using the technique of backpropagation. In the training part, the individual is trained and examined through the test part with an adaptive learning rate known as the Adam optimization. The default environmental setup consists of two exponential decay rates, $\beta_1 = 0.9$ and $\beta_2 = 0.999$; the learning rate or step size $\alpha = 0.001$; and a constant, $\epsilon = 10E - 8$, indicating a very small number, which prevents any division by zero in the implementation. The optimization goal of the projected MOCNN is a classification accuracy enhancement; FLOPs are estimated for each individual according to computational cost, which becomes the next goal, so that the proposed MOCNN can attempt to reduce the FLOP counts.

Since CNN training time is high compared to estimating FLOPs, two techniques can be used to minimize the computational cost of deriving the classification accuracy. To minimize the training process epochs, a terminating condition is executed when there is no improvement in accuracy in about10 epochs. This minimizes the training time.

Searching CNN frameworks is efficient because the complexity of various individuals might vary significantly. Different numbers of epochs may be needed to fully train the various individuals. For instance, the CNN framework might be simpler with one or two layers with a small feature map count; in this case, a small number of epochs is required for CNN training. Or the CNN framework might be a composite a huge feature map count and hundreds of layers in every layer, and more epochs would be needed to train the complicated

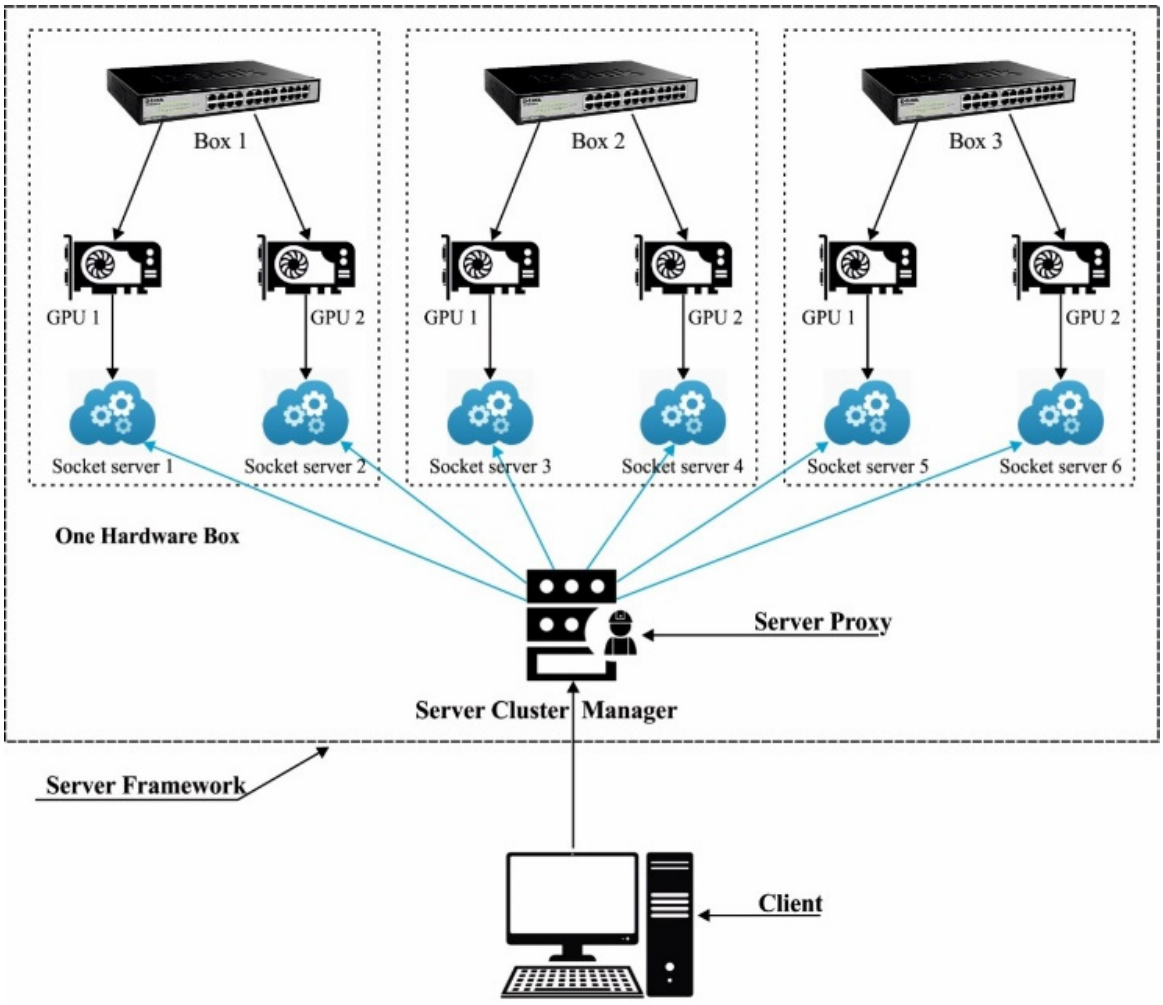

FIGURE 3. The infrastructure used to boost the experiment.

CNN entirely. For training CNNS with different levels of complexity, it is difficult to assign a fixed number of epochs needed through objective evaluation.

The proposed MOCNN sets epoch counts that are high enough to entirely train the highly composite CNNs in the search space, and uses the condition of early termination to terminate the training process to reduce computational costs. In every generation, each individual is examined through the evaluation of objectives, and many CNNs are examined over the entire evolution process. Between that, it might represent the individuals as the similar frameworks of CNN trained duplicate and examine. In order to prevent the similar CNNs from duplicate training, for every individual, the derived classification accuracy can be saved in the memory prior to the completion of the program and loaded at the beginning of the program. Prior to individual training in objective evaluation, an individual search of saved classification accuracy is carried out initially, and training is done if the search yields no results. Adam optimization is selected as a technique of backpropagation, and the entire training dataset is used for CNN evaluation.

C. INFRASTRUCTURE USED TO BOOSTTHE MOCNN

The evaluation of the objective is a bottleneck for the proposed MOCNN technique, and deriving the classification accuracy through evaluating and training the individual is also a bottleneck. A simple and general objective evaluation can be executed using one GPU card for every individual. One technique to enhance the CNN training performance is to influence the functionality of multi-GPU through the widely used method of CNN training over multiple GPUs to increase the training speed. We propose a structure for reducing the time cost of the proposed MOCNN, shown in Figure 3, which can influence all accessible GPU cards over numerous machines to perform subsequent objective evaluations of the individual batch. The relevant Python library is built and available as open source. The framework contains a server

FIGURE 4. Sample images from the UCM dataset.

group executing that is derived at framework top with three boxes that are machines, wherever the two GPU cards are derived over every box.

A socket server is running on every GPU card to manage the requests and listen the client. A CNN that needs to be examined might be moved to the server from the client, and the training is done by the server that examines the CNN. Then the classification accuracy is returned to the client. There are two reasons to use one GPU card in spite of using a full box as a socket server. As the practical framework probably has hardware boxes with different numbers of GPU cards installed, when the full box is being used as a socket server, every socket server has a different capacity from every other. The computational cost of individual training is probably the same when the proposed MOCNN and the individual batch are similar to the socket server count that might be transmitted to the server clusters for evaluation of the objective.

The client wants to gather the outcomes of batch evaluation while all the individuals within the batch are evaluated to maintain the order of the individual evaluation results so that it is the same as transmission. It is important to manage the socket server's capacity to minimize its idle time. While the client is still waiting for the batch evaluation to be finished, a few higher-capacity socket servers might finish earlier. The efficiency of the multi-GPU mode is based on certain structures, and a few structures cannot attain the multiple GPU optimal usage, mostly because resources need to be distributed securely through numerous program threads while examined with one thread. If every GPU card is used as a socket server, shared resource handling is not a problem; hence, the GPU is used efficiently. In the middle of the structure, there is a server cluster manager; for instance, a server proxy that handles the objective evaluation concurrency through the socket server cluster.

All the CNNs receive the proxy server, which must be examined and saved as the CNN pool. In the cluster, all the socket servers find the availability of the server proxy. It depends on the number of available servers, and it derives

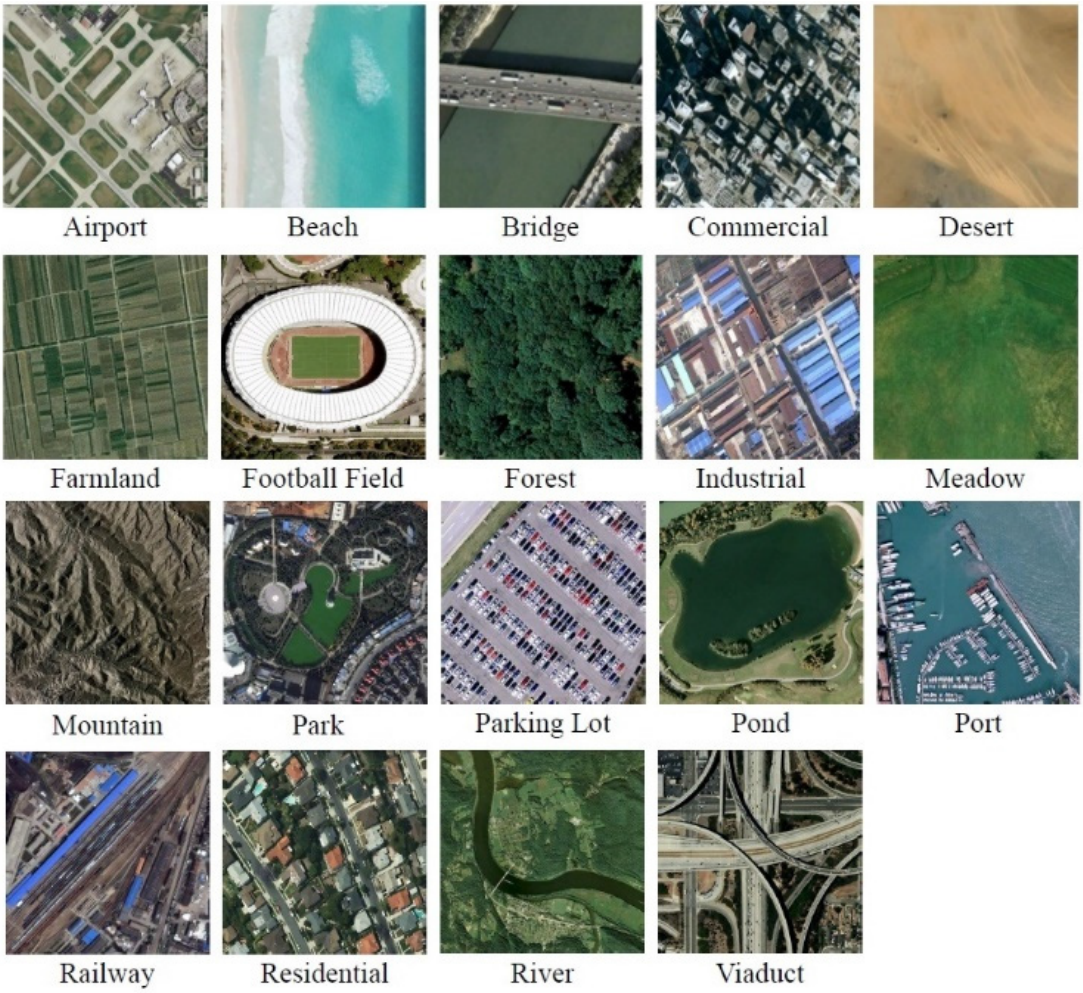

FIGURE 5. Sample images from the WHU-RS dataset.

the unevaluated CNN batches, in which it is similar to the server availability and shared every CNN within the batch simultaneously to the socket servers available. For the whole CNN, the proxy server waits to gather the evaluation outcomes that might be joined with the CNNs. Through the repetition of the steps, the entire CNNs offer the classification accuracy joined to it and the evaluated CNNs are returned to the client.

To influence the use of multiple GPU cards on numerous machines, a technique that has an objective evaluation of CNN counts might function as a client. Many concurrent operations are managed through the server cluster and proxy. The client usage is direct, and all the CNNs need to be sent to the server proxy and wait for the response from the server proxy. The full EC technique executes as a client that is a major part of the program. In the entire population, individuals are sent to the server proxy at the start of every generation. The major program using the EC technique may continue when the evaluated individuals are returning from the server proxy. The proposed technique is executed in a similar way to executing on a distinct machine, and adjustments are transmitted to the individuals through the objective evaluation server proxy in spite of CNN evaluation through itself.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

This section investigates the significant characteristics of the proposed method using a set of two datasets. The proposed method is implemented using the Python programming language in a PC Intel(R) Core (TM) i7-7500 CPU at 2.70GHz, with 8GB and 1TB HDD. Different tests have been carried out. In addition, a comparative study with the existing models was also made.

A. DATASET

The presented model was tested on two open access datasets, the UC Merced (UCM) Land Use Dataset and the WHU-RS Dataset. The details are provided in Table 1.

1) UCM DATASET [7]

The data have been gathered manually from large aerial orthoimagery consisting of 21 different types of scenes. Every class contains 100 images of varying sizes

TABLE 1. Dataset details.

Parameter	UCM dataset	WHU-RS dataset
Source	[7]	[27]
Types	21	19
No. of images	100	950
Pixel size	256x256	600x600

of 256 × 256 pixels, and the pixel resolution is around one foot. Figure 4 shows sample images from the UCM dataset. Note that the UCM dataset exhibits very small inter-class diversity between some classes, which contain identical objects or identical textural patterns. This makes this dataset more challenging.

2) WHU-RS DATASET [27]

The data are gathered from Google Earth (Google Inc.), an open access dataset containing 950 images with a pixel size of 600 × 600 pixels, uniformly distributed among 19 scene classes. A few sample images are presented in Figure 5. An obvious difference in brightness, scalability and resolution creates a harsher platform than the UCM dataset.

B. RESULTS ANALYSIS

Compared to the various pre-trained CNN models [15], the proposed method performs as shown in Figure 6. The attained higher classifier accuracy indicates the effective capability of classifying pre-trained CNNs towards scene datasets. Two datasets were used, UCM and WHU-RS. For the UCM dataset, PlacesNet had a lower accuracy rate of 91.44%. The VGG-VD networks came short of desired outcome, obtaining 93.15% for VGG-VD19 and 94.07% for VGG-VD16. This is despite the fact that these networks have many layers and perform well with numerous natural image classification standards compared to shallow CNN models. The other methods, like VGG-M, VGG-F, CaffeNet, and AlexNet, performed well whereas the VGG-S performed better, attaining 94.60%. However, the proposed method attained the highest accuracy of 95.32%.

For the WHU-RS dataset, PlacesNet had a lower accuracy rate of 91.73%, as shown in Figure 7. The VGG-VD networks did not attain the desired outcomes, obtaining 94.36% for VGG-VD19 and 94.35% for VGG-VD16. The other methods, like VGG-M, VGG-F, CaffeNet, and AlexNet, performed poorly. VGG-S performed better, attaining 95.46%. The proposed method had the highest accuracy of 96.84%.

PlacesNet failed to outperform AlexNet with natural scene datasets, performs the worst, mainly compared to AlexNet, when textural and structural patterns within scenes are very different from the actual scenes.

In the UCM dataset, the computation time of CNN features was examined with every pre-trained CNN method, as shown in Figure 8. CaffeNet, PlacesNet, AlexNet, and VGG-F took more or less the same computation time because their frameworks are similar. Their computation time was 90 seconds. VGG-VD19 and VGG-VD16 took more computation time than the other models, as they have a large number of layers. Their computation time was 500 seconds and 410 seconds, respectively. VGG-S and VGG-F did not perform impressively. The proposed method had the lowest computation time of 80 seconds.

Table 2 presents the results and classification accuracy attained with different other recently proposed standard techniques with the UCM dataset. GoogLeNet+Fine-tune adjusts the pre-trained CNN using the target dataset and derives CNN activations directly through image feeding, with no modification, in pre-trained CNN parameters. Our method performed considerably better with the WHU-RS dataset. For both public benchmark datasets, superior classification results were obtained.

TABLE 2. Comparative accuracy analysis with various models.

Method	Accuracy (%)
SPM [28]	74.00
SCK [7]	72.52
SPCK++ [29]	77.38
SC+Pooling [21]	81.67
SG+UFL [22]	86.64
CCM-BOVW [8]	86.64
PSR [9]	89.10
UFL-SC[30]	90.26
MSIFT[31]	90.97
COPD [20]	91.33
Dirichlet[32]	92.80
VLAT[33]	94.30
CaffeNet [25]	93.42
OverFeat [25]	90.91
GoogLeNet+Fine-tune [24]	97.10
DenseNet	89.21
Proposed	97.88

The SCK method had a poor accuracy of 72.52%.SPM and SPCK++ were the next worst performers, with respectively 74% and 77.38%. SG+UFL and CCM-BOVW both had an accuracy of 86.64%, whereas DenseNet had an accuracy

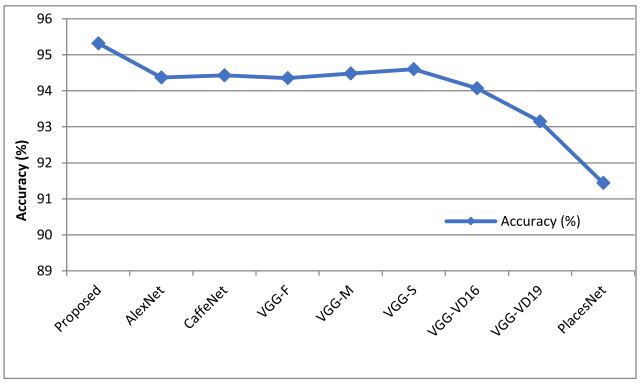

FIGURE 6. Comparative accuracy analysis of different CNN models using the UCM dataset.

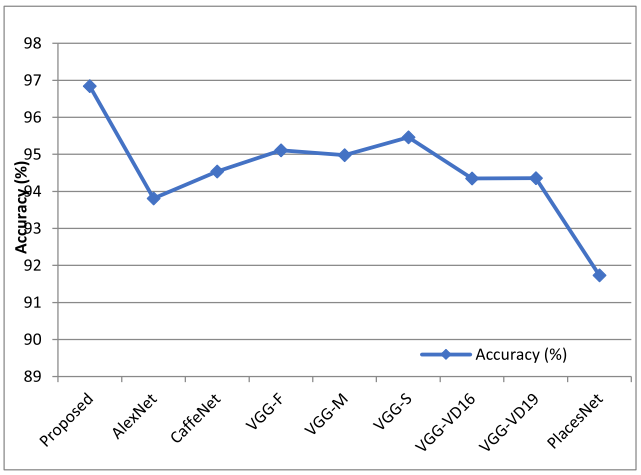

FIGURE 7. Comparative accuracy analysis of different CNN models using the WHU-RS dataset.

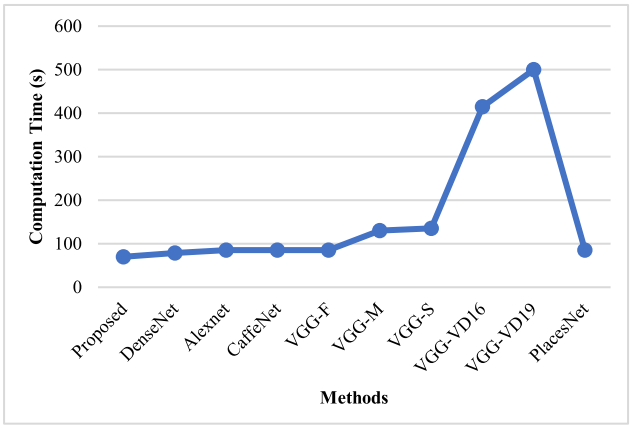

FIGURE 8. Comparative computation time analysis of different CNN models.

of 89.21%. Dirichlet, CaffeNet, and VLAT had accuracies of 92.8%, 93.42%, and 94.3%, respectively. Other methods did not perform impressively. GoogLeNet+Fine-tune performed better than the other methods, with an accuracy of 97.1%. However, the proposed method performed best of all, with an accuracy of 97.88%.

These results showed that the proposed model yields superior classification using the datasets considered. The proposed model has the lowest computation time, at 80 seconds, and the highest classification accuracy with the given datasets. As expected, the proposed model outperforms all the other techniques. Our technique is very straightforward, using a simple linear classifier to test and train, and extracts the features from convolutional or fully connected layers of pre-trained CNN. The proposed technique is much more accurate than the other recently proposed methods, even with a small training sample count.

V. CONCLUSION

This paper presents a new multi-objective optimization model for evolving state-of-the-art deep CNNs for scene classification, generating non-dominant solutions developed in an automated way at the Pareto front. The proposed method initially allows the UAV to capture videos. Then the videos are divided into a set of frames. Then preprocessing of the video frames is done to allow further processing. Afterward, the training part takes place using OMOPSO with a CNN. Once the training process is completed, a model is derived. During the testing phase, the proposed CNN-based model is run, and the provided input image undergoes effective scene classification. A set of two benchmark datasets was used to test the performance of the scene classification model, and a detailed study has been made, comparing the model to other well-known models. The proposed method has the lowest computation time (80 seconds) and the highest accuracy (97.88%) compared to all the other methods. The simulation results show that the proposed model is effective. In the future, this work can be further improved through hyperparameter-tuning models to reduce computation time even more.

ACKNOWLEDGMENT

Partial work of Dr. Gyanendra Prasad Joshi was supported by Sejong University new faculty research funds.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] A. Darwish, A. E. Hassanien, and S. Das, "A survey of swarm and evolutionary computing approaches for deep learning," *Artif. Intell. Rev.*, vol. 53, no. 3, pp. 1767–1812, Mar. 2020.

[2] Z. Wu, W. Quan, and T. Zhang, "Resource allocation in UAV-aided vehicle localization frameworks," in *Proc. IEEE/CIC Int. Conf. Commun. Workshops China*, Aug. 2019, pp. 98–103.

[3] F. De Smedt, D. Hulens, and T. Goedeme. *On-Board Real-Time Tracking of Pedestrians on a UAV*. Accessed: Apr. 14, 2020. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_workshops_2015/W12/html/Smedt_On-Board_Real-Time_Tracking_2015_CVPR_paper.html

[4] S. Kapania, D. Saini, S. Goyal, N. Thakur, R. Jain, and P. Nagrath, "Multi object tracking with UAVs using deep SORT and YOLOv$_3$ retinaNet detection framework," in *Proc. st ACM Workshop Autonomous Intell. Mobile Syst.*, Bangalore, India, Jan. 2020, pp. 1–6, doi: 10.1145/3377283.3377284.

[5] R. K. Dewangan, A. Shukla, and W. W. Godfrey, "Three dimensional path planning using grey wolf optimizer for UAVs," *Int. J. Speech Technol.*, vol. 49, no. 6, pp. 2201–2217, Jun. 2019.

[6] L. Bampis and A. Gasteratos, "Revisiting the Bag-of-Visual-Words model: A hierarchical localization architecture for mobile systems," *Robot. Auto. Syst.*, vol. 113, pp. 104–119, Mar. 2019.

[7] Y. Yu, Y. Yuan, H. Guan, D. Li, and T. Gu, "Aeroplane detection from high-resolution remotely sensed imagery using bag-of-visual-words based Hough forests," *Int. J. Remote Sens.*, vol. 41, no. 1, pp. 114–131, 2020.

[8] L.-J. Zhao, P. Tang, and L.-Z. Huo, "Land-use scene classification using a concentric circle-structured multiscale Bag-of-Visual-Words model," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 7, no. 12, pp. 4620–4631, Dec. 2014.

[9] S. Chen and Y. Tian, "Pyramid of spatial relatons for scene-level land use classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 1947–1957, Apr. 2015.

[10] T. Ojala, M. Pietikainen, and T. Maenpaa, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002, doi: 10.1109/TPAMI.2002.1017623.

[11] G. Liu, G.-S. Xia, W. Yang, and L. Zhang, "Texture analysis with shape co-occurrence patterns," in *Proc. 22nd Int. Conf. Pattern Recognit.*, Aug. 2014, pp. 1627–1632.

[12] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.

[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[15] F. Hu, G.-S. Xia, J. Hu, and L. Zhang, "Transferring deep convolutional neural networks for the scene classification of high-resolution remote sensing imagery," *Remote Sens.*, vol. 7, no. 11, pp. 14680–14707, Nov. 2015, doi: 10.3390/rs71114680.

[16] O. Ghorbanzadeh, T. Blaschke, K. Gholamnia, S. Meena, D. Tiede, and J. Aryal, "Evaluation of different machine learning methods and deep-learning convolutional neural networks for landslide detection," *Remote Sens.*, vol. 11, no. 2, p. 196, Jan. 2019.

[17] A. Carrio, C. Sampedro, A. Rodriguez-Ramos, and P. Campoy. (Aug. 2017). *A Review of Deep Learning Methods and Applications for Unmanned Aerial Vehicles*. Accessed: Jun. 20, 2020. [Online]. Available: https://www.hindawi.com/journals/js/2017/3296874/

[18] F. Al-Turjman, H. Zahmatkesh, and L. Mostarda, "Quantifying uncertainty in Internet of medical things and big-data services using intelligence and deep learning," *IEEE Access*, vol. 7, pp. 115749–115759, 2019.

[19] L. Ye, L. Wang, Y. Sun, R. Zhu, and Y. Wei, "Aerial scene classification via an ensemble extreme learning machine classifier based on discriminative hybrid convolutional neural networks features," *Int. J. Remote Sens.*, vol. 40, no. 7, pp. 2759–2783, Apr. 2019.

[20] G. Cheng, J. Han, P. Zhou, and L. Guo, "Multi-class geospatial object detection and geographic image classification based on collection of part detectors," *ISPRS J. Photogramm. Remote Sens.*, vol. 98, pp. 119–132, Dec. 2014.

[21] A. M. Cheriyadat, "Unsupervised feature learning for aerial scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 1, pp. 439–451, Jan. 2014.

[22] F. Zhang, B. Du, and L. Zhang, "Saliency-guided unsupervised feature learning for scene classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 4, pp. 2175–2184, Apr. 2015.

[23] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3828–3836.

[24] M. Castelluccio, G. Poggi, C. Sansone, and L. Verdoliva, "Land use classification in remote sensing images by convolutional neural networks," 2015, *arXiv:1508.00092*. [Online]. Available: http://arxiv.org/abs/1508.00092

[25] O. A. B. Penatti, K. Nogueira, and J. A. dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2015, pp. 44–51, doi: 10.1109/CVPRW.2015.7301382.

[26] H. Yu, Y. Wang, and S. Xiao, "Multi-objective particle swarm optimization based on cooperative hybrid strategy," *Int. J. Speech Technol.*, vol. 50, no. 1, pp. 256–269, Jan. 2020.

[27] J. Hu, T. Jiang, X. Tong, G.-S. Xia, and L. Zhang, "A benchmark for scene classification of high spatial resolution remote sensing imagery," in *Proc. IEEE Int. Geosci. Remote Sens. Symp. (IGARSS)*, Jul. 2015, pp. 5003–5006, doi: 10.1109/IGARSS.2015.7326956.

[28] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Oct. 2006, pp. 2169–2178.

[29] Y. Yang and S. Newsam, "Spatial pyramid co-occurrence for image classification," in *Proc. Int. Conf. Comput. Vis.*, Nov. 2011, pp. 1465–1472, doi: 10.1109/ICCV.2011.6126403.

[30] F. Hu, G. Xia, Z. Wang, X. Huang, L. Zhang, and H. Sun, "Unsupervised feature learning via spectral clustering of multidimensional patches for remotely sensed scene classification," *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 8, pp. 2015–2030, 2015.

[31] A. Avramoviá and V. Risojeviá, "Block-based semantic classification of high-resolution multispectral aerial images," *Signal, Image Video Process.*, vol. 10, no. 1, pp. 75–84, Jan. 2016, doi: 10.1007/s11760-014-0704-x.

[32] T. Kobayashi, "Dirichlet-based histogram feature transform for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 3278–3285, doi: 10.1109/CVPR.2014.413.

[33] R. Negrel, D. Picard, and P.-H. Gosselin, "Evaluation of second-order visual features for land-use classification," in *Proc. 12th Int. Workshop Content-Based Multimedia Indexing (CBMI)*, Jun. 2014, pp. 1–5, doi: 10.1109/CBMI.2014.6849835.

[34] K. Shankar, Y. Zhang, Y. Liu, L. Wu, and C.-H. Chen, "Hyperparameter tuning deep learning for diabetic retinopathy fundus image classification," *IEEE Access*, vol. 8, pp. 118164–118173, 2020.

[35] A. Rajagopal, A. Ramachandran, K. Shankar, M. Khari, S. Jha, Y. Lee, and G. P. Joshi, "Fine-tuned residual network-based features with latent variable support vector machine-based optimal scene classification model for unmanned aerial vehicles," *IEEE Access*, vol. 8, pp. 118396–118404, 2020.

[36] I. V. Pustokhina, D. A. Pustokhin, D. Gupta, A. Khanna, K. Shankar, and G. N. Nguyen, "An effective training scheme for deep neural network in edge computing enabled Internet of medical things (IoMT) systems," *IEEE Access*, vol. 8, pp. 107112–107123, 2020.

[37] V. Porkodi, A. R. Singh, A. R. W. Sait, K. Shankar, E. Yang, C. Seo, and G. P. Joshi, "Resource provisioning for Cyber–Physical–Social system in Cloud-Fog-Edge computing using optimal flower pollination algorithm," *IEEE Access*, vol. 8, pp. 105311–105319, 2020.

[38] I. V. Pustokhina, D. A. Pustokhin, J. J. P. C. Rodrigues, D. Gupta, A. Khanna, K. Shankar, C. Seo, and G. P. Joshi, "Automatic vehicle license plate recognition using optimal K-Means with convolutional neural network for intelligent transportation systems," *IEEE Access*, vol. 8, pp. 92907–92917, 2020.