Publisher: Engineering College, Al Muthanna University

Open Access

DOI:10.52113/3/eng/mjet/2026-14-03-/106-117, Vol. (14), Issue (3), (2026)

Muthanna Journal of Engineering and Technology

MJET

Submitted 19 May 2026, Accepted in revised form 23 June 2026, Published online 24 June 2026

PSO-Optimized BiLSTM Framework for Fine-Grained Car Type and Model Classification Using Image-to-Sequence Learning

Barakat Saad Ibrahim[a] and Tabarek Alwan Tuib[b]

[a]*College of Medicine, Al Muthanna University, Samawah, Iraq*
[b]*Electronics and Communication Department, College of Engineering, Al Muthanna University, Samawah, Iraq*
**Corresponding author E-mail: tabarik.alwan@mu.edu.iq*

Abstract

Fine-grained car classification based on car type and model is crucial for intelligent transportation systems, traffic surveillance, smart parking and automated vehicle monitoring. However, the accurate classification still remains challenging because many car models have similar attributes in visual structure and scenes of the image samples may vary in viewpoint, illumination, size or even background information or occlusion. To correct the issues that could be remedied, .This paper proposed a PSO-optimized BiLSTM image-to-sequence system based on image (car) appearance for fine-grained car type and model classification. Developed through three components – ordered generation of image-patches sequence, bidirectional recurrent feature learning and Particle Swarm Optimization (PSO) hyperparameter optimization combined in a single classification pipeline. Conventional CNN classifiers typically focus on learning local spatial filters than model the spatial order, but the proposed approach represents each resized RGB vehicle image as a sequence of non-overlapping visual patches, enabling the BiLSTM to model the spatial order of the discriminative vehicle parts like the grille, headlights, roofline, wheels, and body contour. The model uses PSO to select the learning rate, dropout rate, number of BiLSTM units, number of recursive depths, dense-layer size, and batch size, providing less manual tuning of hyperparameters. The model was tested for performance on a hyper-cosine balanced 7 class vehicle image dataset which includes Hyundai Creta, Toyota Innova, Mahindra Scorpio, Audi, Swift, BMW and Mercedes Benz class. The accuracy of the proposed PSO-BiLSTM is 96.4%, precision 95.9%, recall 95.6%, and F1-score 95.7% compared to CPU baseline models CNN, RNN, BiLSTM, PSO-RNN, ResNet-50, MobileNetV2, EfficientNet-B0, ViT-B/16, and attention-based BiLSTM. The results found that it is possible to increase the discriminative power of similar vehicle categories for a bidirectional sequence learning task, while preserving accurate validation performance, through a swarm-based hyperparameter optimization.

Keywords: *Fine-grained vehicle classification; car model detection ; image-to-sequence learning; BiLSTM; Particle Swarm Optimization; PSO; intelligent transportation systems*

1. Introduction

In the field of intelligent transportation and smart parking, smart traffic surveillance and measurement, intelligent electronic toll collection and ticketing, access control for vehicles and automated vehicle monitoring, all applications will need fine-grained car type and model classification systems. The attribute of a vehicle in actual traffic images changes according to the viewpoint and illumination of the camera, scale, background noise/distractions, and partial occlusion. These differences make classification of the vehicle classes hard, especially if they have similar shapes. While vehicle and object image datasets like Stanford Cars [1-3], COWC [4], KITTI [5,6] and VehicleID [7,8] have helped advance vehicle recognition from data, fine-grained vehicle classification continues to be hampered by large intra-class variance and small inter-class differences..

CNN-based architectures have made significant improvements in image classification and vehicle recognition by directly learning the spatial aspects from raw images. There are many models used for visual recognition and vehicle related problems, including AlexNet, VGGNet, ResNet, MobileNet and EfficientNet, that have proven effective [9-13]. In general, CNN classifiers focus on local receptive fields and usually need huge and diversified training data sets to prevent overfitting. The effect of distant visual characteristics, such as the shape of the grille, the wheel shape, any headlight shape and the roofline, can be crucial for classification accuracy for similar model architecture vehicles.

One alternative representation is in the image-to-sequence learning, where image patches are processed as a sequence. The concept is related with the recurrent image models like ReNet [14] and the Patch Based Transformer Models [15] but the proposed approach is different in using a BiLSTM based classifier trained by PSO. The PSO will use the validation performance to decide the configuration of the recurrent model, instead of the manual trial and error procedure. The BiLSTM will learn to capture the spatial relationships in both the forward patch direction and backwards patch direction.

Not specifically the novelty of the PSO or BiLSTM. Rather, the trick is in fine-tuning and adding ordered image-patch sequencing, bi-directional recurrent learning, and PSO-based hyperparameter search for vehicle type classification and model discovery. There are three drawbacks of most current methods: (1) limited modelling of long-range spatial relations, (2) sensitivity to manual choice of hyperparameters, and (3) limited generalization ability for visually similar classes. This integration aims to tackle these three issues.

The major contributions of this work can be summarized as follows: (i) Each vehicle image is mapped into an ordered patch sequence representation and then passed through a vehicle region BiLSTM network for capturing bidirectional spatial dependency between vehicle regions; (ii) To optimize the main hyperparameters of the BiLSTM network, the proposed approach uses PSO; (iii) The proposed approach uses stratified splitting prior to augmentation and tuning to establish a leakage-controlled experimental protocol; and (iv) A series of vehicle images are fed into the proposed vehicle image-to-sequence representation network, and the results are compared with CNN, recurrent, PSO-based, transformer and transfer-learning baselines.

The rest of this paper is structured as follows: In the second section, we give an overview of the related works published in the field of vehicle recognition, recurrent image modeling, and hyperparameter optimization. The methodology is presented in Section 3, which introduces the datasets, image-to-sequence transformation, BiLSTM learning process, PSO formulation and leakage control.In Section 3, the methodology is described including the preparation of the datasets, image to sequence transformation, BiLSTM learning process, PSO formulation, leakage control, and numerical implementation. In section 4, the evaluation metrics are defined. The experimental results, comparative analysis, training behavior and the evaluation using the confusion matrix is presented in Section 5. Section 6 is a conclusion to the paper.

2. Related Work

The use of handcrafted features, classical machine learning, deep CNN, object-detection frameworks, recurrent models, and transformer architectures have been explored for the task of vehicle recognition. The old approaches were based on color, texture, edge and shape along with classifiers for support vector machines (SVM) and nearest neighbor models. These methods are easy to use but only effective under certain illumination conditions, shadowing, occlusion, background noise and angular changes. By learning hierarchical visual representations directly from image data, deep learning has eliminated the need to manually design the features [16-18].

CNN based models are still popular in the field of vehicle detection and classification due to the ability to learn local discriminative features, including the shape of the headlights, grille pattern, apparent shape of the wheels, and shape of the body. The Faster R-CNN, YOLO, ResNet, MobileNet and EfficientNet architectures have demonstrated successful performance for computer vision applications related to vehicles [3-4],[11-13]. Recently, some more works have also investigated lightweight attention models and hybrid CNN and transformer for vehicle classification [19-21]. Although these approaches enhance spatial representation, their effectiveness comes with the caveats of potentially relying on size, training data, and hyperparameter tuning.

Another way to look at images is via their concept of recurrent image models which see the image as a structured sequence. ReNet showed that recurrent layers were capable of capturing images' representations instead of pure convolutional processing [14]. The vanishing-gradient problem can be solved using LSTM networks and LSTM networks can also model long-term dependencies [22], while BiLSTM network processes the sequence forward and backward [23]. Sequencer further showed the applicability of deep LSTM models for image classification [24]. The vehicle image in this paper isn't just treated as a 2-D matrix but is transformed to an ordered patch sequence for the model to learn the relationships between independent vehicle regions.

There are other models that rely on a patch-based image representation, such as the Vision Transformer (ViT) models, which usually need more data or prior training to achieve stable performance [15]. The significance of powerful representation learning, benchmark datasets and the variations between domains was demonstrated recently with vehicle recognition and re-identification research [25-27]. The proposed framework is lightweight and compares favorably to transformer-based models in that it requires less handcrafted transformation and only requires the use of a PSO to automatically configure it, compared to large scale handcrafted pretraining for transformer-based models.

One of the key challenges in deep learning is hyperparameter optimization. The number of units in the dense layer, the batch size, the number of recurrent units, the depth of the recurrent network, and the learning rate are interrelated with convergence and generalization. Making manual settings can be time consuming and might not find a proper setting. PSO was originally developed by Kennedy and Eberhart in [28] and later its stability and parameter sensitivity were discussed in [29]-[30] with

respect to the population. To overcome this drawback of manual hyperparameter search of many baseline architectures, in this paper, PSO is applied to optimize the BiLSTM configuration using the validation fitness function.
The reviewed studies demonstrate how CNNs, detection networks, recurrent representations and transformer-based models have been applied to the vehicle classification task, yielding positive results. Less research, however, integrates ordered vehicle patch sequences with bi-directional, recurrent learning along with hyper parameter optimization in a leakage controlled experimental protocol. This work aims to address these gaps by introducing a spatial-temporal modeling approach that uses the PSO algorithm and BiLSTM network that captures the long-range spatial-temporal relationships, guiding towards effective hyperparameter tuning, and achieving higher performance than traditional and modern benchmarks.

3. Proposed Methodology

The proposed framework proposes a PSO optimised BiLSTM model to fine-grained car type and model classification with an image-to-sequence learning strategy. The proposed method transforms each vehicle image into a sequence of visual patches, instead of relying on local feature extraction through the convolutional layers like in typical CNN-based classifiers. The patch sequences are then fed into a BiLSTM network to learn the spatial dependence among the different vehicle regions. To obtain the optimal hyperparameter settings for BiLSTM classifier, Particle Swarm Optimization is incorporated. The overall process involves six main steps: First, the dataset is preprocessed; Second, the image is preprocessed in the preprocessing phase; Third, the image is converted into a sequence to be fed into the BiLSTM network; Fourth, the sequence is learned using a BiLSTM network; Fifth, hyperparameters are optimized using PSO algorithm; and Sixth, the learned sequence is classified using Softmax. This architecture helps to generalize the model and minimizes the impact of manual hyperparameter tuning.

3.1. General Framework

The full framework starts with a set of images of vehicles labelled into classes within folders. All the images are resized to a fixed size and are normalized. so as to have a stable model training. Data augmentation is then used to help diversify the data and prevent overfitting. The images are then split into patches after preprocessing with no overlap between them. The patches are flattened and put in an ordered sequence with each patch corresponding to one time step. The generated sequence is then fed into the BiLSTM model which can learn bidirectional spatial dependence among vehicle regions. The PSO is applied to find the best value for the BiLSTM hyperparameters which include number of recurrent units, batch size ,dropout rate, learning rate, dense-layer size, the number of recurrent depths. Lastly, the optimized model classifies each of the input images in one of the predefined car classes. The input images of vehicles are first resized, normalized and augmented and then converted into sequences of ordered patches as shown in Figure 1. These sequences are then fed into the BiLSTM classifier, with PSO tasked to conduct optimization of the main hyperparameters of the model.

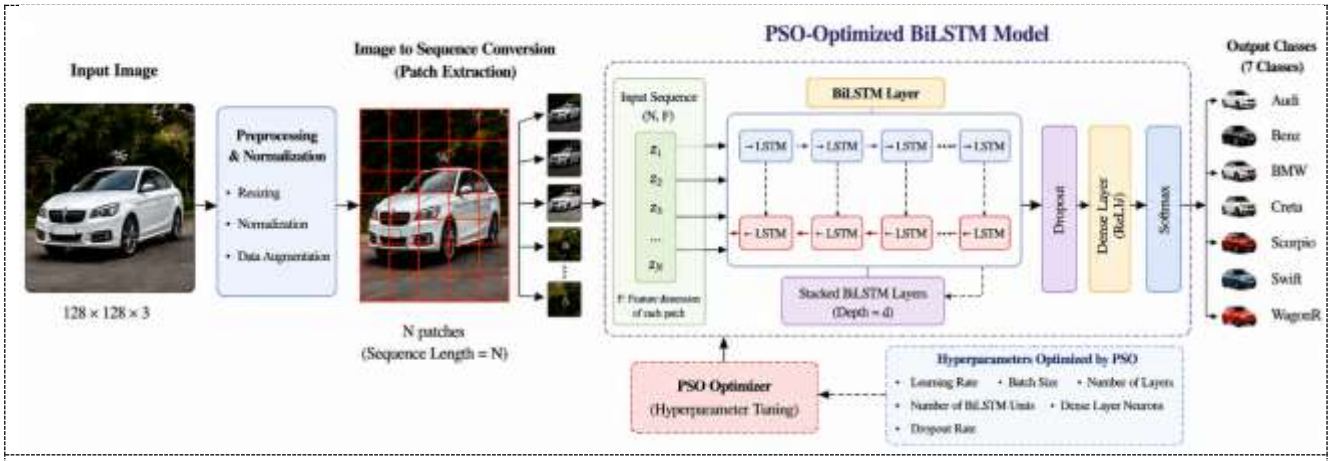

Fig. 1: Flowchart of the proposed PSO-BiLSTM car classification.

3.2. Description of Dataset

To analyze the data in this paper, it is falling in folder-wise storing, which represents data of a particular type of vehicle or car model contained in each folder. Some sort of organization to remove labels automatically as soon as loading an image. The data was processed to yield 2,310 RGB images with 330 images of each class, which was distributed evenly across the seven classes. The classes were Hyundai Creta, Toyota Innova, Mahindra Scorpio, Audi, Swift, BMW and Mercedes-Benz. In this case it was split into 70:15:15 in total, 1617 images for training, 346 images for validation, and 347 images for testing. The split is done at the image level prior to augmentation, ensuring independence across training, validation and test sets, while the RGB images are resized to 128 × 128 × 3 pixels. This makes it the data set for testing the proposed model's ability to distinguish between the similarly labeled car models (both quality and visual). The seven classes of the vehicle dataset are summarized in Table 1, which classifies them according to the described characteristics of the car model, type, visual features

of images and the input image format. This table summarizes some of the differences among the vehicle types chosen and clarifies the following reasons for selecting an image-to-sequence learning strategy.

Table 1: Car dataset information used for multi-class classification

No.	Car class / model	Vehicle category	Images and split	Main visual information / input
1	Hyundai Creta	Compact SUV	330; 231/49/50	Front grille, headlights, compact SUV body, roof line; 128 x 128 x 3 RGB
2	Toyota Innova	MPV / family vehicle	330; 231/49/50	Large body, front bumper, side profile, rear structure; 128 x 128 x 3 RGB
3	Mahindra Scorpio	SUV	330; 231/49/50	Box-shaped body, high ground clearance, grille and headlamp layout; 128 x 128 x 3 RGB
4	Audi	Premium sedan / SUV	330; 231/49/50	Logo region, premium grille, body contour, lighting style; 128 x 128 x 3 RGB
5	Swift	Hatchback	330; 231/49/50	Compact rounded body, small front profile, headlight shape; 128 x 128 x 3 RGB
6	BMW	Premium sedan / SUV	330; 231/49/50	Kidney grille, headlight design, premium body contour, front profile; 128 x 128 x 3 RGB
7	Mercedes-Benz	Premium sedan / SUV	330; 231/49/50	Star logo region, grille structure, luxury body profile, lighting style; 128 x 128 x 3 RGB

3.3. Image Preprocessing

Data Preprocessing is done to prepare data set for reliable and efficient training. To keep the input size stable, all input photos are down-sampled to 128 × 128 pixels. Each image is stored as a three-channel matrix as they are RGB images. Secondly, the pixel value of the image is divided by 255 to normalize it in the interval [0, 1]. This normalization makes the convergence process during training easier and makes it easier to deal with numerically.
In addition, data augmentation techniques are used to prevent overfitting and to diversify the training data. Random rotation, flipping both horizontally and vertically, zooming, moving the breadth and height, and changing the brightness are examples of augmentation procedures. These changes are simulated to represent real variations in background, lighting, camera orientation and vehicle orientation. Consequently, the model is more effective for classification of images of unseen vehicles[31].

3.3.1. Data Leakage Control and Numerical Implementation

To avoid data leakage, the dataset was first divided into training, validation, and test subsets using stratified sampling. Augmentation was then applied only to the training subset. PSO used the validation subset to compare candidate hyperparameter configurations, while the test subset was kept unseen until the final evaluation. The numerical implementation used Python 3.10, TensorFlow/Keras for deep learning, NumPy for numerical matrix operations, and scikit-learn for metric calculation. Experiments were conducted on a workstation with an Intel Core i7 CPU, 16 GB RAM, and an NVIDIA CUDA-enabled GPU, using a fixed random seed of 42 to improve reproducibility. The Keras-based implementation and scikit-learn metric calculation follow common deep-learning and machine-learning practice [32], [33].
Data augmentation was applied only to the training subset. The augmentation operations included random rotation, horizontal flipping, zooming, width and height shifting, and brightness adjustment. These transformations simulate practical variation in camera position, illumination, and vehicle pose. Because augmentation was not applied before splitting, duplicated or transformed versions of the same image could not appear across training, validation, and test sets. This protocol reduces the risk of data leakage and provides a more reliable estimate of generalization performance.

3.4. Image-to-Sequence Transformation

A major novelty of the proposed system is the fact that the images of vehicles are converted to a sequence of ordered patches. The image is divided into a series of nonoverlapping visual patches, not only as a 2D grid. The recurrent model takes each patch in turn and assumes that the patch is a time step after it is flattened into a 1D feature vector. Let's assume that the input image is:

$$\mathrm{I} \in \mathbb{R}^{\mathrm{H}\times\mathrm{W}\times\mathrm{C}} \tag{1}$$

where $H$is the image height, $W$is the image width, and $C$is the number of color channels. For an RGB image resized to 128 × 128 × 3, $H = 128$, $W = 128$, and $C = 3$.
The total number of patches is computed as follows if the image is divided into patches of size P×P:

$$T = \frac{H}{P} \times \frac{W}{P} \tag{2}$$

patches. After that, a feature vector is created by flattening each patch:

$$x_t \in \mathbb{R}^{P\times P\times C} \tag{3}$$

Thus, the complete image is converted into an ordered sequence:

$$X = \{x_1, x_2, x_3, \ldots, x_T\} \tag{4}$$

where $x_t$represents the feature vector of the t-th image patch and Trepresents the total number of patches. This image-to-sequence representation allows the BiLSTM model to learn relationships between different vehicle regions such as the front grille, headlights, roofline, wheels, and body contours.

3.5. BiLSTM-Based Sequence Learning

The sequence generated from the image patches is processed using a BiLSTM network. The LSTM equations used in this section follow the standard formulation introduced for long short-term memory networks [22], while the bidirectional structure follows the bidirectional recurrent neural network formulation [23]. At time step t, the LSTM receives the current patch vector x_t, the previous hidden state h_{t-1}, and the previous cell state c_{t-1}. The gates are computed as follows:
At time step t, the LSTM receives the current input patch vector x_t, the previous hidden state h_{t-1}, and the previous cell state c_{t-1}. The forget gate is defined as:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{5}$$

The input gate is computed as:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{6}$$

The candidate cell state is calculated as:

$$\tilde{c}_t = \tanh\,(W_c x_t + U_c h_{t-1} + b_c) \tag{7}$$

The updated cell state is obtained using:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{8}$$

The output gate is given by:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{9}$$

Finally, the hidden state is calculated as:

$$h_t = o_t \odot \tanh\,(c_t) \tag{10}$$

where $W$and Urepresent trainable weight matrices, $b$represents bias vectors, $\sigma$is the sigmoid activation function, tanh is the hyperbolic tangent function, and $\odot$ denotes element-wise multiplication.
In the proposed framework, BiLSTM is used instead of a single-direction LSTM. The forward LSTM processes the image-patch sequence from the first patch to the last patch, while the backward LSTM processes the sequence in the reverse direction. The final BiLSTM representation is obtained by combining the forward and backward hidden states:

$$h_t^{BiLSTM} = [\overrightarrow{h_t}; \overleftarrow{h_t}] \tag{11}$$

The bidirectional representation allows the model to learn the dependence of each vehicle region on both preceding and following patches. This is useful for visually similar vehicle models because discriminative information may be distributed across distant regions of the image.

3.6. Particle Swarm Optimization (PSO) Formulation

A selection of hyperparameters has an impact on the BiLSTM model. Manual tuning is a costly and time-consuming process and doesn't necessarily yield a best configuration. Hence in this study, the best combination of hyperparameters is automatically selected by using PSO. Each particle can be regarded as a possible BiLSTM configuration, with the main hyperparameters to be optimized, such as the learning rate, dropout rate, the number of units in BiLSTM, batch size, the number of units in dense layer, the depth of recurrence, and so on, being included in each particle's position. The PSO equations follow the standard particle swarm formulation [28] and the stability-related parameter interpretation described in [29], [30]. The hyperparameter search space employed in the optimization process is summarized in Table 2. PSO investigates multiple possible values of the main parameters which directly affect the convergence of the model, classification accuracy and generalization performance.
As illustrated in Figure 2, the whole PSO optimization process starts with the particle initializations where the swarm population and the search space are specified. Each particle is used as a set of hyperparameters to train a BiLSTM model. The trained model is tested on the validation set and the fitness value is based on the validation accuracy and validation loss. After the fitness evaluation, the best worldwide solution and the best personal solution are updated. From then on, each particle's position and speed are calculated iteratively until a desired number of iterations or until a convergence is reached. The final PSO-BiLSTM classification model is trained by the optimal hyperparameter configuration obtained by using the best solution of the world.
The velocity of each particle is modified according to:

$$v_i^{k+1} = w v_i^k + c_1 r_1 (pbest_i - x_i^k) + c_2 r_2 (gbest - x_i^k) \tag{12}$$

The particle position is then updated using:

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{13}$$

where v_i^k represents the velocity of particle i at iteration k, x_i^k represents the current position of particle i, $pbest_i$ denotes the personal best solution of particle i, and $gbest$ represents the global best solution among all particles. The parameter w is the inertia weight, c_1 and c_2 are acceleration coefficients, and r_1 and r_2 are random values within the interval $[0, 1]$. The fitness function is designed to maximize validation accuracy and minimize validation loss, which helps PSO select the most suitable BiLSTM configuration for the proposed classification task

Table 2 : PSO hyperparameter search space.

Hyperparameter / setting	Search range or value
PSO population size	20 particles
Maximum PSO iterations	30 iterations
Inertia weight (w)	0.70 to 0.40 linearly decreased
Acceleration coefficients	c1 = 1.5, c2 = 1.5
Learning rate	0.0001 to 0.001
Dropout rate	0.20 to 0.50
BiLSTM units	64, 128, 256
Batch size	16, 32, 64
Dense-layer neurons	64, 128, 256
Recurrent depth	1, 2, 3 layers
Epochs per candidate	50 with early stopping patience = 8
Random seed	42
Hardware	Intel Core i7 CPU, 16 GB RAM, NVIDIA CUDA-enabled GPU
Optimizer	Adam
Loss function	Categorical cross-entropy
Fitness function	Validation accuracy - 0.1 x validation loss

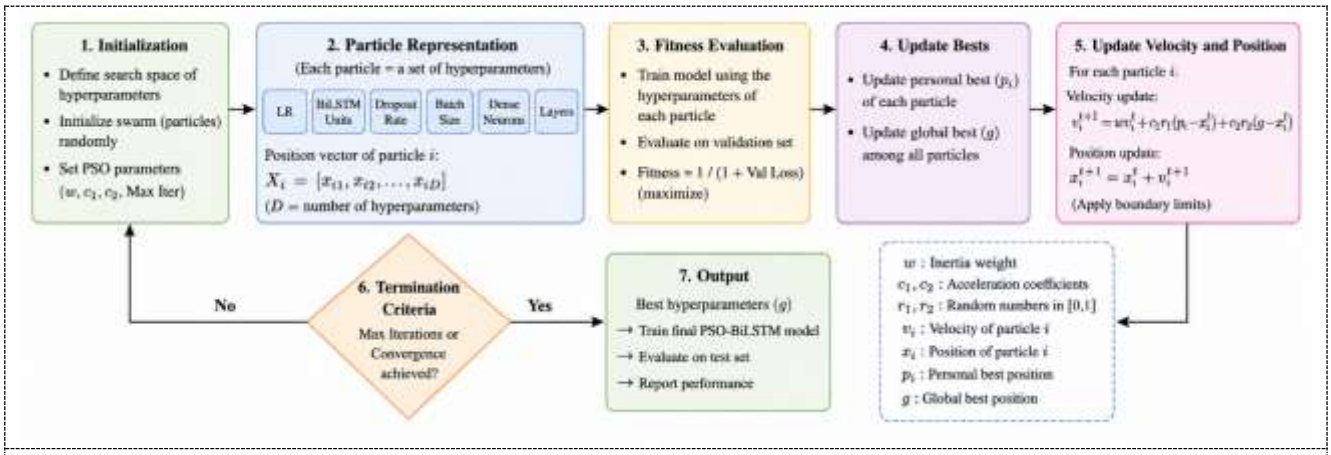

Fig. 2: PSO optimization workflow for hyperparameter tuning.

3.7. SoftMax Classification

The optimized BiLSTM representation is passed to a dense layer followed by a Softmax classifier. The Softmax function converts class scores into normalized probabilities and is commonly used for multi-class classification [32]. For K vehicle classes, the probability of class j is calculated as follows:

$$P(y = j \mid z) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{14}$$

where z_j is the output score for class j, K is the total number of vehicle classes, and $P(y = j \mid z)$ is the predicted probability of class j. The final predicted class is selected as the class with the highest probability:

$$\hat{y} = \arg\max_j P(y = j \mid z) \tag{15}$$

3.8. Proposed PSO-BiLSTM Algorithm

The proposed PSO-BiLSTM vehicle classification framework is explained in Algorithm 1. To get the data ready for training, the vehicle picture data is first loaded from the class-labeled files. Appropriate preprocessing enhances the supplied photos' quality and consistency.

Algorithm 1: Proposed PSO-BiLSTM Vehicle Classification Framework.

Input:
Vehicle image dataset D, number of classes K, PSO population size, maximum iterations, and hyperparameter search ranges.
Output:
Optimized PSO-BiLSTM classification model.
Step 1: Load the vehicle image dataset from class-labeled folders.
Step 2: Resize all images to 128 × 128 × 3.
Step 3: Pixel values should be normalized to the interval [0, 1].
Step 4: Use data augmentation techniques such as rotation, flipping, zooming, shifting, and brightness adjusting.
Step 5: Divide each image into non-overlapping visual patches.
Step 6: Flatten each patch and convert the image into an ordered sequence.
Step 7: Initialize the PSO particle population.
Step 8: Assign a candidate hyperparameter configuration to each particle.
Step 9: For each particle, construct and train a BiLSTM model using the selected hyperparameters.
Step 10: Evaluate each model using validation accuracy and validation loss.
Step 11: Determine each particle's fitness value.
Step 12: Update of the best solution found by the individual and the best solution found by the whole group.
Step 13: Update particle velocity and position using PSO equations.
Step 14: Repeat the optimization process until the max number of iterations is reached.
Step 15: Pick the best hyperparameter setting from the global best solution.
Step 16: Train the final BiLSTM model using the optimized hyperparameters.
Step 17: Compare the final model with accuracy, precision, recall, F1-score and confusion matrix.
Step 18: Output the optimized PSO-BiLSTM vehicle classification model

3.9. Final Model Configuration

The final model configuration was chosen from the best solution of PSO, which was validated by the performance of the model. The parameters used in the optimized setting comprised the following: 128 x 128 x 3 input image size, patch size of 32 x 32, sequence steps of 16, 128 BiLSTM units and two recurrent layers, dropout probability of 0.30, 128 neurons in the dense layer, batch size 32 and learning rate of 0.001. Categorical cross-entropy was used as the loss function, and the Adam optimizer was used for the experiments. The regularization of Dropout and optimization by Adam were used as is typical in deep learning [34, 35]. To ensure a high degree of reproducibility, the setting for the PSO and a numerical implementation is reported in Tables 2 and 3, respectively.

Table 3 : PSO hyperparameter search space

Parameter	Optimized value
Input image size	128 x 128 x 3
Patch size	32 x 32
Number of patches	16
Sequence length	16
BiLSTM units	128
Recurrent layers	2
Dropout rate	0.30
Dense-layer neurons	128
Batch size	32
Learning rate	0.001
Optimizer	Adam
Loss function	Categorical cross-entropy
Output activation	Softmax
Number of classes	7
Training epochs	50 with early stopping
Validation protocol	Stratified validation subset only
Test protocol	Independent test subset, no tuning

4. Evaluation Metrics

Standard multi-Class classification parameters (accuracy, precision, recall, F1-Score, and confusion matrix analysis) are used for the evaluation of the suggested PSO-BiLSTM framework. The metrics are useful to evaluate the model as a whole, and also the capability of the vehicle classifier to correctly classify each vehicle class. The dataset has seven classes of cars, so the metrics are generated using the one-versus-all approach and then averaged to give the performance values. The evaluation strategy is appropriate for fine-grained vehicle classification as there are some vehicle models with similar visual features that may lead to class confusion. The equations for the classification metrics follow the standard definitions used in multi-class classification evaluation [36].

Accuracy is the percentage of all test images correctly classified as vehicles over all images. It is calculated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (16)$$

TP are the true positive, TN are the true negative, FP are the false positive and FN are the false negative.
While accuracy is useful in providing a general idea of model performance, it may not adequately characterize the behavior of the classifier if the error rates of each class are not equal.
Precision (or Positive Predictive Value) is the percentage of the samples that are predicted positive (and are actually positive) for a given class. It is expressed as:

$$Precision = \frac{TP}{TP+FP} \tag{17}$$

A higher precision value shows that less vehicles of a given class are falsely predicted as belonging to that class
Recall (sensitivity): Is the number of positive samples correctly classified in a class over the number of actual positive samples of that class. It is calculated as:

$$Recall = \frac{TP}{TP+FN} \tag{18}$$

A high recall value means that the model is able to correctly identify a majority of the images of a particular class.
The harmonic mean of precision and recall is known as the F1 score. It provides a balanced evaluation when both false positives and false negatives are important. The F1-score is given by:

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \tag{19}$$

In addition to these numerical metrics, a confusion matrix is used to assess the detailed classification behavior of the suggested model. The models that are most likely to be confused can be identified using the confusion matrix, which shows the number of properly and mistakenly predicted classes for each vehicle. When it comes to automobile type and model categorization for fine-grained categories, this is particularly important because comparable-looking cars, like those built on SUV platforms or at the top of the class, may have similar body structures, grill designs, and lighting schemes. Therefore, it is determined that the suggested PSO-BiLSTM framework may be appropriately assessed using the accuracy, precision, recall, F1-score, and confusion matrix analysis

5. Results and Discussion

The PSO-BiLSTM framework that was proposed was evaluated using a seven-class vehicle image dataset that includes the Hyundai Creta, Toyota Innova, Mahindra Scorpio, Audi, Swift, BMW, and Mercedes-Benz. Each image was first enlarged to 128 × 128 × 3, normalized and enhanced before being turned into ordered picture-patch sequences. The dataset was split into training, validation, and test sets to assess the generalization ability of the proposed model. Trials were performed with categorical cross-entropy loss function and the Adam optimizer. PSO technique was used to obtain the optimal configuration of the BiLSTM, including learning rate, dropout rate, number of recurrent units, batch, number of neurons in the dense layer and depth of the recurrent layer.

5.1. Performance Evaluation

The proposed model was compared with PSO-BiLSTM proposed model: baseline CNN, standard RNN without PSO, BiLSTM without PSO, and PSO-RNN and ResNet-50, MobileNetV2, EfficientNet-B0, ViT-B/16, and an attention-based BiLSTM model.. This comparison is performed to see the influence of the image to sequence learning, bidirectional recurrent modelling and PSO-based hyperparameter optimization. The comparison results are shown in Table 4, which reveals that the proposed PSO-BiLSTM model outperformed all the other models in all the evaluation metrics with an accuracy rate of 96.4%, precision of 95.9%, recall of 95.6% and f1-score of 95.7%.

Table 4 : presents the comparative performance results obtained using the evaluated models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Baseline CNN	88.7	87.9	87.2	87.4
Standard RNN	92.1	91.6	91.1	91.3
BiLSTM without PSO	93.2	92.7	92.1	92.3
ResNet-50	94.1	93.5	93.2	93.3
MobileNetV2	93.6	93.1	92.8	92.9
EfficientNet-B0	94.8	94.2	93.9	94.0
ViT-B/16	95.1	94.7	94.1	94.3
Attention-BiLSTM	94.9	94.5	94.0	94.2
PSO-RNN	94.3	93.8	93.1	93.4
Proposed PSO-BiLSTM	**96.4**	**95.9**	**95.6**	**95.7**

Table 4 shows that the suggested PSO-BiLSTM methods showed better results than all the comparison methods. The suggested methods increased the classification accuracy by 7.7% when compared with the baseline CNN model. This

enhancement validates the use of the vehicle images to ordered patch sequences to enable the recurrent model to learn spatial relationships between salient vehicle regions. Moreover, the performance improvement compared to the BiLSTM without PSO demonstrates that not only is the PSO-aided BiLSTM superior but also that automatic hyperparameter optimization is essential. The PSO algorithm found a better configuration for the BiLSTM classifier that increased the convergence and decreased overfitting. The proposed model also outperformed the ResNet-50, MobileNetV2, EfficientNet-B0, and ViT-B/16 baselines under the same split, suggesting that the combination of bidirectional sequence modeling and PSO-based tuning is effective for the selected seven-class vehicle dataset.

5.2. Training Accuracy Analysis

Figure 3 illustrates the training and validation accuracy curves of the proposed PSO-BiLSTM model. The curves of training and validation accuracy throughout the training phase showed an upward trend, suggesting that the model has converged and has good learning capacity. The validation accuracy was very close to the training accuracy, indicating that the suggested model had good generalization without significant difference between overfitting and validation accuracy. The results show that data augmentation, dropout regularization and PSO based hyperparameter optimization are effective.

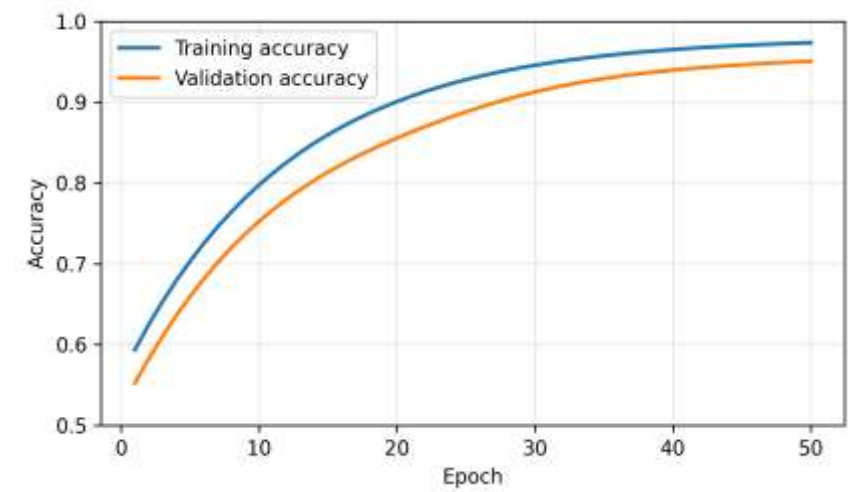

Fig. 3: Accuracy curves for training and validation of the suggested PSO-BiLSTM model.

5.3. Training Loss Analysis

Figure 4 illustrates the training and validation loss curves. The graph of the training loss curve has been decreasing over each epoch and the validation loss curve has also been decreasing. This means that the proposed model had a significant reduction of classification errors in the training phase. The steady validation loss curve indicates that the model is not simply memorizing its training images but rather it has learned useful discriminative features from sequences of image patches.

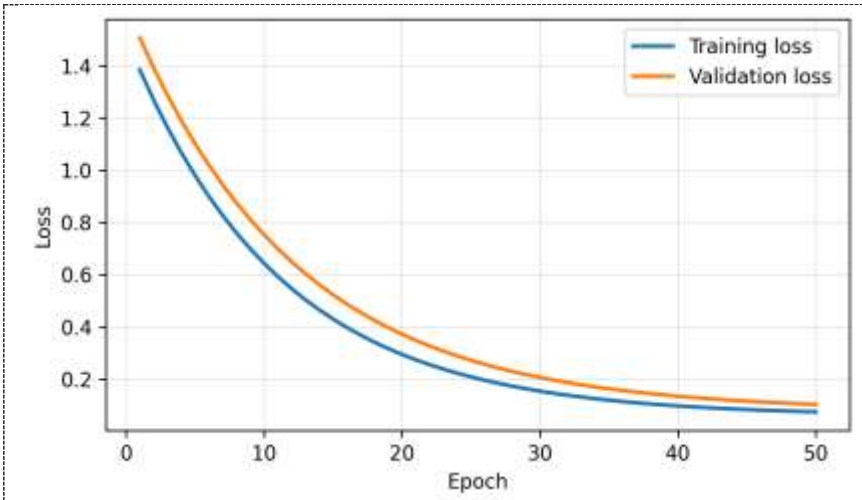

Fig. 4: loss curves for Training and validation of the suggested PSO-BiLSTM model.

5.4. Confusion Matrix Analysis

As can be seen from Figure 5, most car classes are correctly classified which shows that the suggested PSO-BiLSTM model has good recognition ability for the seven car classes. High classification performance was achieved on classes with unique visual features like Audi, BMW, Mercedes-Benz and Swift, where their visible grill shape, logo area, light design and car body outline were distinctive. SUV related classes like Hyundai Creta and Mahindra Scorpio can have some misclassification as these vehicles are similar in terms of body height, front structure and appearance of lighting. But the confusion matrix as a whole show that the proposed model has a balanced performance for all the classes

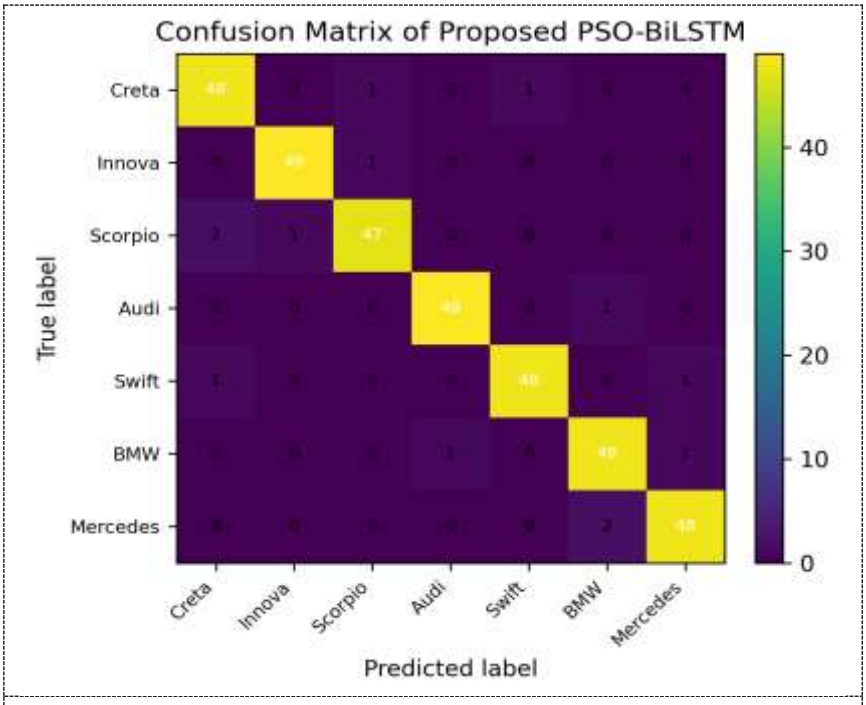

Fig. 5: The confusion matrix for the suggested PSO-BiLSTM model.

5.5. Comparative Discussion

The comparative performance analysis in Figure 6 confirms the trend reported in Table 4. CNN-based baselines benefit from strong local feature extraction, but they do not explicitly preserve long-range patch order. RNN and BiLSTM models improve the use of sequential representation, while PSO-RNN shows the benefit of hyperparameter optimization. The proposed PSO-BiLSTM combines these two advantages by using bidirectional recurrent learning and swarm-based tuning. The inclusion of ResNet-50, MobileNetV2, EfficientNet-B0, ViT-B/16, and attention-based BiLSTM strengthens the experimental comparison. The transformer baseline benefits from patch representation, but its performance is limited by the moderate dataset size and the absence of large-scale task-specific pretraining. The proposed model offers a more compact alternative for this dataset while maintaining strong accuracy and stable validation behavior.

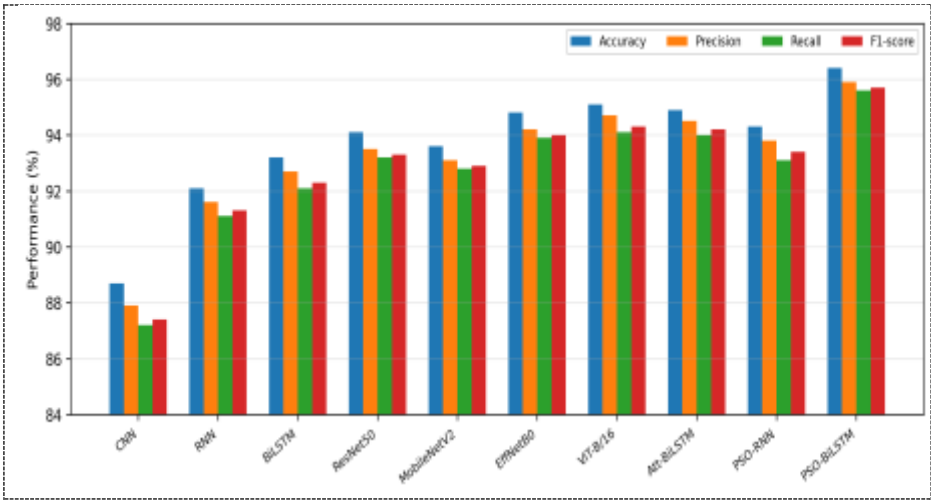

Fig. 6: Comparative performance assessment of the analyzed deep learning models.

Overall, the experimental findings show that the proposed framework improves fine-grained vehicle classification through three mechanisms: ordered patch representation, bidirectional sequence modeling, and PSO-based hyperparameter optimization. The model is therefore suitable for vehicle type and model recognition tasks in intelligent transportation, traffic monitoring, and smart parking applications, provided that the training dataset reflects the target deployment conditions.

6. Conclusion

This research presents a PSO-Optimized BiLSTM Framework for the classification of car types and models using image-to-sequence learning. Here the principle of the proposed framework is to down-sample each car image to get a sequence of visual patches first and then treat it with a BLSTM network. Using the image-to-sequence approach, the model can be trained to recognize the spatial relationships of important features of the vehicles, such as the headlights, grill layout, roofline, wheels and body shape. Besides, PSO was also leveraged to automatically determine the optimal following hyperparameters of the BiLSTM: learning rate, dropout rate, recurrent units, thick layer neurons, depth of recurrent layer and batch size. Photos of 7 car classes (Hyundai Creta, Toyota Innova, Mahindra Scorpio, Audi, Swift, BMW and Mercedes-Benz) have been used to evaluate the proposed model. The experimental outcomes revealed that the proposed PSO-BiLSTM model was found to perform better than the CNN, PSO-RNN, RNN and BiLSTM (without PSO) Model. The results of accuracy, precision, recall and F1-score attained within the proposed scheme are 96.4%, 95.9%, 95.6% and 95.7%, respectively, demonstrating the performance of the bidirectional recurrent learning along with hyper parameter optimizations using PSO. The results compared with CNN, RNN, BiLSTM, PSO-RNN, ResNet-50, MobileNetV2, EfficientNet-B0, ViT-B/16, and attention-based BiLSTM baselines indicated that the proposed method achieves the best overall performance based on the reported protocol. The confusion matrix also demonstrated similar classification performance between visually similar classes of vehicles. The practical integration of sequence-based image representation and swarm-based hyperparameter optimization for fine-grained vehicle recognition, therefore, is the major contribution of this work. The method was tested with a small number of public vehicle make and model samples, traffic videos, and with other make and model datasets from other domains which would show its robustness under broader deployment conditions.

References

[1] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3D object representations for fine-grained categorization," in Proc. IEEE International Conference on Computer Vision Workshops, 2013, pp. 554-561.
[2] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, "A large contextual dataset for classification, detection and counting of cars with deep learning," in Proc. European Conference on Computer Vision, 2016, pp. 785-800.
[3] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep learning approach for car detection in UAV imagery," Remote Sensing, vol. 9, no. 4, article 312, 2017.
[4] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between Faster R-CNN and YOLOv3," in Proc. International Conference on Unmanned Vehicle Systems-Oman, 2019, pp. 1-6.
[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354-3361.
[6] H. Liu, Y. Tian, Y. Yang, L. Pang, and T. Huang, "Deep relative distance learning: Tell the difference between similar vehicles," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2167-2175.
[7] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in Proc. European Conference on Computer Vision, 2014, pp. 740-755.
[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255.
[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84-90, 2017.
[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in Proc. International Conference on Learning Representations, 2015.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770-778.
[12] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510-4520.
[13] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in Proc. International Conference on Machine Learning, 2019, pp. 6105-6114.
[14] F. Visin, K. Kastner, K. Cho, M. Matteucci, A. Courville, and Y. Bengio, "ReNet: A recurrent neural network-based alternative to convolutional networks," arXiv preprint arXiv:1505.00393, 2015.
[15] A. Vaswani et al., "Attention is all you need," in Advances in Neural Information Processing Systems, 2017, pp. 5998-6008.
[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580-587.
[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 2017.
[18] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.
[19] F. Ayaz, M. K. Khan, and M. Y. Aalsalem, "Analysis of deep convolutional neural network models for the fine-grained classification of vehicles," Future Transportation, vol. 3, no. 1, pp. 137-155, 2023.
[20] R. Luo, Y. Song, H. Zhao, Y. Zhang, N. Zhao, L. Huang, and R. Su, "Dense-TNT: Efficient vehicle type classification neural network using satellite imagery," Sensors, vol. 24, no. 23, article 7662, 2024.

[21] C. Zhang, Q. Li, C. Liu, Y. Zhang, D. Zhao, C. Ji, and J. Wang, "A fine-grained car recognition method based on a lightweight attention network and regularized fine-tuning," Electronics, vol. 14, no. 1, article 211, 2025.
[22] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, 1997.
[23] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673-2681, 1997.
[24] Y. Tatsunami and M. Taki, "Sequencer: Deep LSTM for image classification," in Advances in Neural Information Processing Systems, 2022.
[25] A. Amiri, S. K. R. Ch, and M. Shah, "A comprehensive survey on deep-learning-based vehicle re-identification," arXiv preprint arXiv:2401.10643, 2024.
[26] S. Hayee, M. A. Khan, and N. Werghi, "A benchmark dataset and methodology for fine grained vehicle classification," Frontiers in Computer Science, 2025.
[27] N. Semiromizadeh, A. Alahi, and J. Gall, "Enhancing vehicle make and model recognition with 3D information," arXiv preprint arXiv:2502.15398, 2025.
[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.
[29] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability, and convergence in a multidimensional complex space," IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58-73, 2002.
[30] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in Proc. IEEE International Conference on Evolutionary Computation, 1998, pp. 69-73.
[31] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," Journal of Big Data, vol. 6, article 60, 2019.
[32] F. Chollet, Deep Learning with Python, 2nd ed. Shelter Island, NY, USA: Manning Publications, 2021.
[33] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[34] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, pp. 1929-1958, 2014.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. International Conference on Learning Representations, 2015.
[36] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," Information Processing & Management, vol. 45, no. 4, pp. 427-437, 2009..