University of Tehran

Earth Observation and Geomatics Engineering Online ISSN: 2588-4360

Homepage: https://eoge.ut.ac.ir/

Comparative Analysis of Machine Learning and Deep Learning Algorithms for UAV-Based Rooftop Classification

(Selected Paper in the 8th ISPRS Geospatial Conference 2025, University of Tehran, Iran)

Saba Kazemi[1], **Mohammad Karimi**[2]✉, and **Parastoo Pilehforooshha**[3]

1. Department of Geospatial Information Systems, Faculty of Surveying Engineering, K. N. Toosi University of Technology, Tehran, Iran. E-mail: sabakazemi@email.kntu.ac.ir
2. Corresponding author, Department of Geospatial Information Systems, Faculty of Surveying Engineering, K. N. Toosi University of Technology, Tehran, Iran. E-mail: mkarimi@kntu.ac.ir
3. Department of Geospatial Information Systems, Faculty of Surveying Engineering, K. N. Toosi University of Technology, Tehran, Iran. E-mail: pilehforoosh@kntu.ac.ir

Article Info

Article type:
Research Article

Article history:
Received 2026-01-15
Received in revised form 2026-02-14
Accepted 2026-02-15
Available online 2026-06-02

Keywords:
Rooftop Classification,
Machine Learning,
Deep Learning,
Ensemble Learning,
Transfer Learning.

ABSTRACT

Rooftop type classification plays a crucial role in numerous urban applications such as 3D city modelling, solar potential estimation, disaster management, and smart city development. Leveraging high-resolution UAV imagery and advances in machine learning and deep learning, this study conducts a comparative evaluation of traditional machine learning and deep learning algorithms for classifying rooftop types, gable, half-hip, and complex, using orthophotos from Rasht City, Iran. In order to achieve that, firstly a structured and annotated dataset of rooftops was constructed through manual digitization and image cropping, followed by extensive training data augmentation. In the second step, two handcrafted feature sets (HOG and HOG+LBP) were used to train five machine learning classifiers (SVM, Random Forest, KNN, XGBoost, and Logistic Regression). The best results among these was achieved by SVM (F1-score = 0.74), while KNN performed the weakest. Also, ensemble learning methods were utilized through the aggregation of diverse model predictions. Using ensemble learning techniques, particularly stacking, significantly enhanced classification accuracy, reaching an F1-score of 0.79. In the third step, deep learning models including a custom CNN and a MobileNetV2-based transfer learning approach were utilized. The latter achieved the highest overall classification accuracy (83.3%) and macro F1-score (83%), outperforming all traditional methods by learning spatial hierarchies directly from data. The results demonstrate that while classical methods benefit from interpretable feature design and faster training, deep learning showed better within-domain generalization on our Rasht dataset.

Cite this article: Kazemi, S., Karimi, M., & Pilehforooshha, P. (2025). Comparative Analysis of Machine Learning and Deep Learning Algorithms for UAV-Based Rooftop Classification. *Earth Observation and Geomatics Engineering*, Volume 9, Issue 2, Pages 40-53. http//doi.org/ 10.22059/eoge.2026.409749.1197

Publisher: University of Tehran.
DOI: http//doi.org/ 10.22059/eoge.2026.409749.1197

1. Introduction

Urban buildings form one of the most fundamental structural components of cities, playing a central role in shaping the built environment. Among the various parts of a building, the rooftop stands out as a particularly important element, not only from an architectural or structural perspective, but also due to its relevance in numerous urban applications. Rooftop characteristics are vital for tasks such as 3D city modelling, solar potential estimation, disaster risk analysis, energy efficiency assessment, and urban planning (Spasov et al. 2023).

Automatic rooftop type classification has become technically feasible with the increasing availability of high-resolution images captured by unmanned aerial vehicles (UAVs), as well as the rapid advancement of machine learning (ML) and deep learning (DL) techniques (Meng et al. 2023). Such classification enables detailed semantic understanding of urban structures, which is crucial for tasks like semantic segmentation of buildings in geospatial databases (Spasov et al. 2023). Specifically, incorporating rooftop type information helps enrich the semantic content of urban scenes, allowing for more informed decision-making in applications like smart city development and building information modelling (Pan et al. 2020). In this context, the aim of the present paper is rooftop type classification using ML and DL algorithms.

Recent advances in DL have significantly transformed image classification tasks by enabling models to learn hierarchical feature representations directly from raw data, removing the dependency on handcrafted features (Cai et al. 2021). CNN-based architectures such as ResNet, VGG, and MobileNet have been applied to rooftop classification with promising results, especially when fine-tuned via transfer learning on pretrained weights like ImageNet (Meng et al. 2023, Spasov et al. 2023). Among these, lightweight models such as MobileNet are particularly suitable for deployment in resource-constrained environments, including UAV-based real-time mapping (Cai et al. 2021).

Despite these advances, multiple challenges persist. For instance, rooftops exhibit considerable intra-class variability due to occlusions, shadows, and hybrid shapes, conditions that often result in misclassifications, particularly among structurally similar classes like gable and hip roofs (Pan et al. 2020). In addition, class imbalance and limited training samples for rare rooftop types (e.g., half-hip or complex structures) remain a significant barrier to generalization (Castagno et al. 2018, Mutreja et al. 2025).

To address these limitations, recent research suggests leveraging data augmentation and multi-source fusion techniques to improve model robustness and classification accuracy (van Driel 2019). Also, traditional ML algorithms that rely on handcrafted features such as HOG and LBP offer interpretable solutions but may lack the representational power needed to handle these complex variations (Meng et al. 2023). Moreover, existing datasets are often geographically biased or exclude rooftops affected by occlusion, further limiting the adaptability of trained models to unseen urban contexts (Yildirim et al. 2025). These limitations highlight the need for a comparative approach for rooftop type classification that not only leverages both ML and DL paradigms under controlled experimental settings, but also systematically investigates the effects of different feature representations, ensemble strategies, and network architectures.

In response to these gaps, this study presents a comprehensive evaluation of various effective ML and DL models for the automatic classification of urban building rooftops from high-resolution UAV imagery. The research is structured in four steps including data preparation, ML-based classification, DL-based classification, and results evaluation.

- In the first step (data preparation) a structured and annotated dataset of rooftop images is constructed which has categorized the rooftops into three distinct architectural types (gable, half-hip, and complex).
- In the second step (ML-based classification), the effectiveness of ML algorithms in rooftop classification is investigated using handcrafted features, by comparing two feature configurations: (1) Histogram of Oriented Gradients (HOG) alone, and (2) a combined HOG and Local Binary Patterns (LBP) vector, in order to assess the impact of texture information on model results. Also, ensemble learning methods (hard voting, weighted voting, stacking) was utilized through the aggregation of diverse model predictions.
- In the third step (DL-based classification), the evaluation of effective DL algorithms (including CNN and transfer learning strategy using the MobileNetV2 architecture) for rooftop type classification is assessed.
- Finally, a comparative analysis between the results of ML and DL approaches under the same experimental condition is conducted.

The remainder of this paper is organized as follows. Section 2 describes the dataset and methodology. Section 3 presents experimental results and model evaluations. Finally, Section 4 concludes the paper.

2. Methods

The overall workflow adopted in this study is illustrated in Figure 1. As shown, the methodology is organized into four main stages: data preparation, ML-based classification, DL-based classification, and model evaluation which is described in the following.

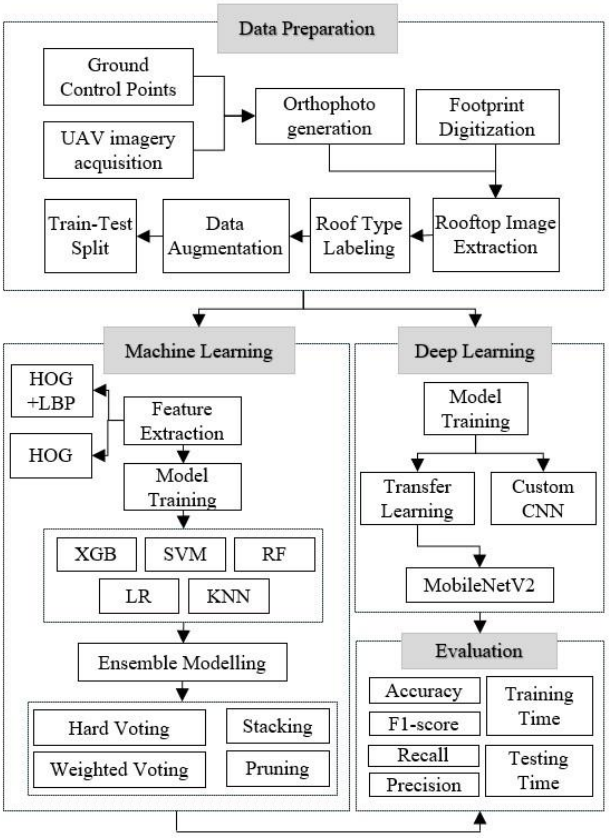
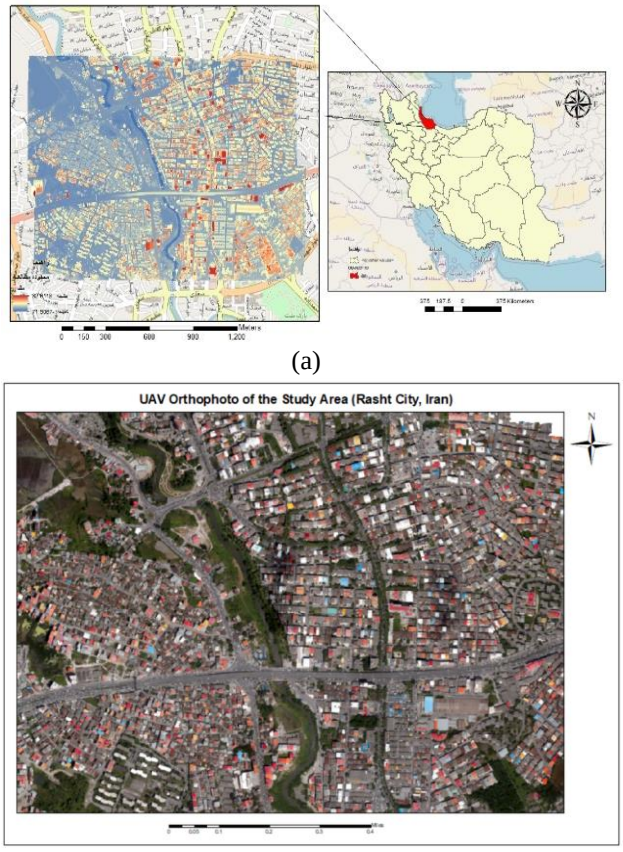


Figure 1. Overview of the methodology



(a)

(b)

Figure 2. Location (a) and UAV-derived orthophoto (b) of the study area.

2.1. Data Preparation

The dataset utilized in this study consists of aerial imagery captured over the Golsar District, located in District 1 of Rasht City, Iran (Figure 2a). A total of 763 high-resolution images were acquired using an UAV, providing a comprehensive top-down view of urban rooftops (Figure 2b). To prepare the imagery for analysis, all raw aerial images were imported into Agisoft Metashape software for photogrammetric processing. The initial step involved setting up the correct geographic coordinate system, corresponding to the real-world location of the study area. This was followed by aerial triangulation, aligning the 763 images through feature matching and bundle adjustment. The triangulated block was then georeferenced using ground control points (GCPs).

After the images were properly aligned and referenced, a high-resolution orthophoto of the area was generated. Then, building footprints were manually digitized with high geometric accuracy. These polygonal boundaries were used to extract individual roof images by cropping the orthophoto. This process yielded a refined dataset of rooftop samples, each corresponding to a unique building structure.

Based on local architectural characteristics, the rooftops were categorized into three classes: gable, half-hip and complex. A selection of cropped rooftop images used in this study is shown in Figure 3. Each image was labelled according to its roof type and stored in a structured JSON file, which includes the image filename and its corresponding class label. To improve model generalization and reduce overfitting, data augmentation techniques were applied only to the training subset after the stratified train–validation–test split at two levels. For ML algorithms, each rooftop image was rotated by 90°, 180°, and 270°, effectively quadrupling the dataset size and introducing rotational variability while preserving the stability of hand-crafted features. In contrast, DL models benefit from learning spatially invariant representations directly from raw images and therefore require a more diverse set of transformations. Accordingly, the applied augmentations included random rotations of up to 20° to simulate different

UAV viewing angles, zooming in and out within a 20% range to account for scale variations, horizontal and vertical translations to increase robustness to positional shifts, and horizontal flipping to simulate changes in object orientation. Additionally, all pixel values were rescaled to the [0, 1] range to normalize the input data.

We intentionally used different augmentation strategies for ML and DL models because these paradigms exploit augmentation in fundamentally different ways. For classical ML, discrete 90° rotations preserve the stability of handcrafted descriptors (HOG/HOG+LBP) while increasing rotational variability with minimal risk of feature distortion. In contrast, CNN-based models benefit from richer, continuous geometric transformations (random rotations, translations, zoom, and flipping) that promote invariance learning directly from pixel space and reduce overfitting. Importantly, the impact of augmentation choices on DL performance was explicitly investigated in an ablation study (Table 10), where richer transformations yielded the best macro F1-score, and removing rotation caused a noticeable drop.

Finally, the dataset was split into training (80%) and test (20%) subsets using a stratified strategy. For deep learning, a validation subset was further split from the training data (e.g., 10% of the training set) and was used only for model monitoring and early stopping, while the test set was used only once for final reporting.

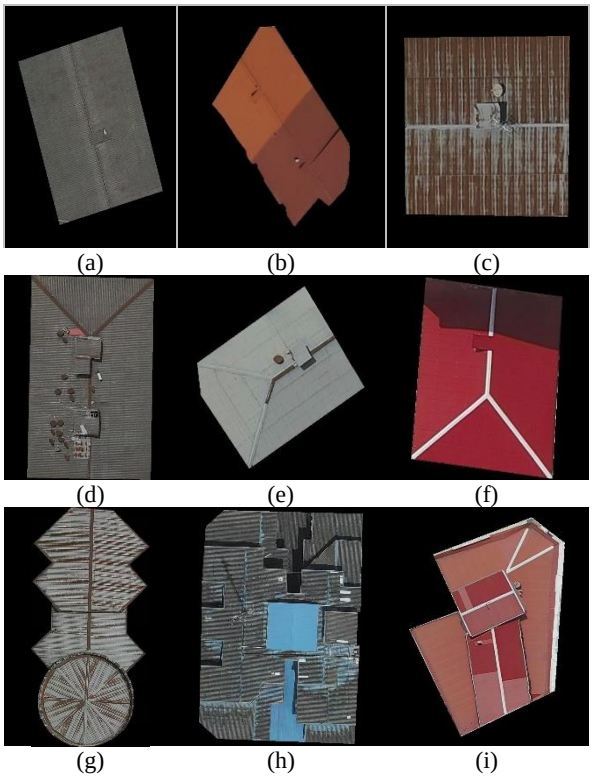


Figure 3. Sample rooftop images: gable class (a–c), half-hip class (d–f), and complex class (g–i)

2.2. ML-Based Classification

In this paper five ML algorithms are implemented to evaluate the effectiveness of traditional ML algorithms in rooftop type classification. The algorithms include Support Vector Machine (SVM), Random Forest (RF), K-nearest neighbors (KNN), Extreme Gradient Boosting (XGBoost) and Logistic Regression (LR). These algorithms are selected due to their effectiveness in image-based classification tasks.

Each algorithm is trained using handcrafted features extracted from rooftop images. For this purpose, two feature sets are considered for comparison: (1) Histogram of Oriented Gradients (HOG) descriptors, and (2) a combined feature vector consisting of both HOG and Local Binary Patterns (LBP). This allowed for the evaluation of the impact of texture descriptors in addition to gradient-based features. For fair comparison, hyperparameters for each classifier were optimized through grid search and cross-validation on the training set. The same train-test split was used across all experiments to ensure consistency. To further enhance prediction stability, an ensemble approach was employed. This ensemble method is known to reduce variance and benefit from the diversity of decision boundaries among the models.

The following subsections provide a detailed overview of the feature extraction methods used to construct the input vectors for classification (Section 2.2.1), followed by individual descriptions of each ML algorithm (Sections 2.2.2 to 2.2.6). Finally, the ensemble strategy based on majority voting is discussed in Section 2.2.7.

2.2.1. Feature Extraction

In this study, features were extracted from rooftop images to enable classical ML algorithms to perform classification tasks. Two well-established descriptors were employed: Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP).

The HOG descriptor is designed to capture the shape and structure of objects by computing the distribution of gradient orientations in localized regions of an image. This method is particularly effective in representing edge information and was originally developed for object detection (Dalal et al. 2005). To complement gradient information with texture-based features, Local Binary Patterns (LBP) were also extracted. LBP is a powerful and lightweight texture descriptor that operates by thresholding each pixel with respect to its surrounding neighbors and encoding the results into binary patterns (Ojala et al. 2002).

Two separate feature sets were constructed for evaluation: (1) HOG-only, consisting solely of the HOG descriptors; and (2) HOG + LBP, a concatenated feature vector that combines both descriptors. This dual-feature approach was adopted to investigate whether the addition of LBP, which emphasizes surface texture, contributes meaningfully to the classification of rooftop types, or

whether HOG alone provides sufficient structural information.

2.2.2. Support Vector Machine

Support Vector Machines (SVMs) are supervised learning models designed for classification tasks. The idea behind SVMs is to find an optimal hyperplane that separates data points of different classes with the maximum margin (Cortes & Vapnik, 1995).

The generalization ability of SVM is highly dependent on key hyperparameters that control model flexibility. The kernel function determines how input data is transformed into higher-dimensional space, with common options including linear, RBF, polynomial, and sigmoid kernels. The regularization parameter (C) controls the trade-off between maximizing the margin and minimizing classification error; smaller values lead to better generalization, while larger ones aim for stricter separation. The gamma parameter, used in non-linear kernels like RBF and polynomial, influences the reach of individual training points, lower values result in smoother decision boundaries, while higher values create tighter ones. For polynomial kernels, the degree parameter sets the complexity of the decision surface (Hsu et al. 2003).

2.2.3. Random Forest

Random Forest is an ensemble learning algorithm that builds multiple decision trees during training and combines their outputs to make a final prediction. Unlike a single decision tree, which can be prone to overfitting, Random Forest reduces variance by introducing randomness into both the training data (via bootstrapped sampling) and the feature selection process at each split (Breiman, 2001).

The classification behavior of a Random Forest classifier depends on several key hyperparameters. The number of trees controls the size of the ensemble; more trees generally lead to more stable predictions but higher computational cost. The maximum depth limits how deep each decision tree can grow, helping prevent overfitting. The minimum number of samples required to split an internal node affects how detailed the splits are; lower values lead to more complex trees, while higher values simplify the model (Breiman, 2001).

2.2.4. K-Nearest Neighbors

One of the approaches employed for rooftop classification in this study is the k-nearest neighbors (k-NN) algorithm, a simple yet powerful non-parametric classification technique. The core idea behind k-NN is that, given a new, unclassified data point, the algorithm finds the k most similar (i.e., nearest) instances from the labelled training data, based on a distance metric such as Euclidean distance, and assigns the new point to the class that is most frequent among those k neighbors (Cover & Hart, 1967).

The classification behavior of the k-NN algorithm is primarily governed by three hyperparameters. The first is the number of neighbors (k), which affects the model's sensitivity, smaller values can capture fine details but may lead to overfitting, while larger values provide more stable predictions. The second is the distance metric used to measure similarity between data points, with common choices including Euclidean and Manhattan distances. The third is the weighting scheme, which determines how much influence each neighbor has on the final prediction; this can be uniform (equal weighting) or distance-based, where closer neighbors contribute more significantly (Cover & Hart, 1967).

2.2.5. XGBoost

Extreme Gradient Boosting (XGBoost) is a highly efficient and scalable ML algorithm that builds an ensemble of decision trees using a gradient boosting framework. Unlike traditional boosting methods, XGBoost includes regularization terms, which helps prevent overfitting and improves generalization, particularly important in rooftop classification tasks with high variability. The algorithm supports parallel tree construction and handles missing data internally by learning optimal split directions (Chen & Guestrin, 2016).

The effectiveness of XGBoost depends on three core hyperparameters. The maximum depth controls how deep each decision tree can grow, directly affecting the model's ability to capture complex patterns, deeper trees increase model capacity but also the risk of overfitting. The learning rate determines the contribution of each new tree to the ensemble; smaller values lead to slower but more stable learning and often result in better generalization. The number of estimators defines how many trees are built in total; increasing this number can improve generalization and stability, but at the cost of increased training time and potential overfitting if not properly regularized. (Chen & Guestrin, 2016).

2.2.6. Logistic Regression

Logistic regression is a statistical method used for modeling a binary outcome. Unlike linear regression, which predicts a continuous outcome, logistic regression estimates the probability that a given input belongs to a particular category (Park, 2013).

The effectiveness of logistic regression depends on a few key hyperparameters. The regularization strength (C) controls the inverse of the penalty applied to model complexity; smaller values of C apply stronger regularization, which can help prevent overfitting. The solver determines the optimization algorithm used to find the model coefficients. Common choices include liblinear (suitable for smaller datasets or L1-regularization) and lbfgs (efficient for larger datasets and L2-regularization) (Ahmed Arafa et al. 2022).

2.2.7. Ensemble Modelling

To enhance the robustness and generalization of the classification, several ensemble strategies were explored. The objective was to combine previous base models in a way

that leverages their individual strengths while compensating for their weaknesses. The following approaches were evaluated in a stepwise fashion (Michalski et al. 1997):

- Model Pruning: Initially, a full ensemble of all trained models was considered. However, some models, particularly those with consistently low evaluation metrics, were found to negatively affect the combined outcome. These models were therefore excluded from the ensemble to reduce noise and bias.
- Hard Voting: The first combination technique implemented was majority voting, also known as hard voting. In this approach, each model casts a vote for its predicted class, and the final decision corresponds to the class with the highest number of votes.
- Weighted Voting: To account for the varying predictive quality of the remaining models, a weighted voting mechanism was used. Each model's vote was scaled according to a weight proportional to its macro-F1 score obtained via cross-validation on the training set.
- Stacking: Finally, a stacking ensemble was developed. In this architecture, the predictions from several base learners serve as input features for a meta-learner, which is trained to produce the final prediction.

2.3. DL-Based Classification

Deep learning has demonstrated exceptional effectiveness in image classification tasks due to its ability to automatically learn hierarchical feature representations from raw pixel data. Unlike handcrafted descriptors such as HOG or LBP, DL models extract spatial and structural patterns directly from images through a sequence of convolutional, activation, and pooling layers. This reduces the dependency on manual feature engineering and enables end-to-end learning within a single unified framework (LeCun et al. 2015).

Two DL strategies were examined in this research. The first approach involved designing and training a custom convolutional neural network (CNN) from scratch, specifically tailored to the rooftop dataset. The second approach adopted a transfer learning strategy using the MobileNetV2 architecture, an efficient and lightweight CNN pretrained on the large-scale ImageNet dataset. In this setup, the base layers of MobileNetV2 were frozen to retain generic visual features, while custom classification layers were added and trained on the rooftop images. The following sections provide descriptions of both architectures and their respective training processes.

2.3.1. Custom CNN Architecture

Convolutional Neural Networks (CNNs) are a class of DL models specifically designed to process data with a grid-like topology, such as images. They utilize layers of learnable convolutional filters to automatically detect spatial hierarchies of features, reducing the need for manual feature engineering. A typical CNN architecture consists of convolutional layers that extract local patterns, pooling layers that reduce spatial dimensionality, and fully connected layers that perform classification based on the learned features (Purwono et al. 2022).

In this study, a compact and efficient custom CNN architecture was trained from scratch on the rooftop image dataset. The network consists of three convolutional layers with ReLU activations and max-pooling, followed by a dense layer with 128 units and dropout (rate = 0.5) to prevent overfitting and a softmax layer as the output layer determining class probabilities for gable, half-hip, and complex roofs. Input images were resized to 128×128 pixels and normalized. To improve generalization, data augmentation techniques such as rotation, translation, zooming, and flipping were applied only during training.

2.3.2. Transfer Learning Using Mobilenetv2

MobileNetV2 is a lightweight convolutional neural network architecture specifically designed for image classification. It improves upon its predecessor (MobileNetV1) through two architectural innovations: inverted residuals and linear bottlenecks. These mechanisms allow the network to maintain representational power while significantly reducing computational cost and memory usage. By connecting narrow intermediate layers (bottlenecks) instead of wide layers, and minimizing the use of nonlinearities within these connections, MobileNetV2 ensures that critical spatial information is preserved even in a highly compressed model (Sandler et al., 2018).

In this paper, MobileNetV2, was used as a transfer learning backbone to improve rooftop classification without training a model from scratch. The base model was initialized with pre-trained ImageNet weights, and its final layers were replaced with a global average pooling layer, a dropout layer (to reduce overfitting), and a softmax output layer for class prediction. The original convolutional layers were frozen to retain general visual features, and only the new layers were trained.

2.4. Evaluation

In this study, Hyperparameter tuning for ML models was performed using 5-fold cross-validation on the training set, and the reported ‘Mean’ metrics (Tables 1–6) correspond to cross-validation averages. Final model performance was then reported once on the independent held-out test set (e.g., confusion matrices and final classification reports). The test set was not used for training, tuning, or model selection. The confusion matrix was employed as a fundamental evaluation tool to analyze classification results. In the context of multi-class rooftop classification, the confusion matrix components are defined as follows: True Positive (TP) represents the number of samples correctly assigned to a given class; False Positive (FP) denotes the number of samples incorrectly assigned to that class; False Negative (FN) corresponds to the number of samples belonging to a

class but misclassified as another; and True Negative (TN) indicates the number of samples correctly identified as not belonging to the class under consideration. Based on these definitions, the evaluation metrics used in this study are defined in Equations (1–5):

Accuracy: the proportion of correctly classified samples over the total number of samples. In the multi-class setting, accuracy corresponds to the ratio of correctly predicted instances (i.e., the sum of the diagonal elements of the confusion matrix) to the total number of samples (Equation 1).

$$Accuracy = \frac{\sum_{i=1}^{C} TP_i}{N} \quad (1)$$

Precision: the proportion of correctly predicted positive samples to the total predicted positives for each class i (Equation 2).

$$Precision = \frac{TP_i}{TP_i + FP_i} \quad (2)$$

Recall (Sensitivity): the proportion of correctly predicted positive samples to all actual positives for each class i (Equation 3).

$$Recall = \frac{TP_i}{TP_i + FN_i} \quad (3)$$

F1-Score: the harmonic mean of precision and recall, balancing both false positives and false negatives (Equation 4)

$$F1 = \frac{Precision \times Recall}{Precision + Recall} \times 2 \quad (4)$$

To evaluate all classes regardless of imbalance, macro-averaging is employed in this paper (Equation 5).

$$Metric_{macro} = \frac{1}{C}\sum_{i=1}^{C} Metric_i \quad (5)$$

where $Metric_i$ is the metric value for class i and C is the number of classes.

It should be noted that, in this paper, in addition to classification metrics, all models are evaluated based on computational efficiency. For this purpose, two critical timing indicators including training time and prediction time are measured.

3. Results and Discussion

This section presents a comprehensive evaluation of the ML and DL models applied to the rooftop classification task. The classification results of each classifier was assessed. In the following, firstly the results of traditional ML algorithms are presented followed by a comparison of ensemble learning models and DL approaches.

3.1. Results of ML Methods

For the purpose of assessing ML-based algorithms in rooftop type classification, the SVM classifier was first evaluated through hyperparameter tuning across multiple kernels and parameter settings under 5-fold cross-validation, using both HOG and HOG+LBP feature sets as explained in Section 2.2.2. The difference between HOG and HOG+LBP was marginal, suggesting that the additional LBP texture information provides only a small complementary benefit under the current dataset and experimental protocol.

Among the evaluated configurations, the SVM with the RBF kernel achieved the best and most consistent result when combined with HOG features, as reported in Table 1. In particular, the models with $C = 100$and $C = 10$yielded identical results, reaching a mean accuracy of 0.74 and a mean F1-score of 0.74 under 5-fold cross-validation, indicating a stable solution with respect to the regularization parameter. A moderate decrease in classification results was observed when reducing C to 1, suggesting that stronger regularization limits the model's capacity to separate rooftop classes in the HOG feature space. In comparison, the polynomial kernel with degree 2 achieved lower overall result, with a mean F1-score of 0.68, highlighting the superior suitability of the RBF kernel for capturing the non-linear relationships inherent in rooftop morphology. Linear SVM configurations were generally less effective than RBF-based models, indicating that the class boundaries in the handcrafted feature space are not well represented by a purely linear decision surface. Sigmoid kernel settings produced the weakest F1-scores overall and were consistently dominated by other kernels, indicating limited suitability for this rooftop classification task. These trends are summarized in Figure 4, which reports the mean cross-validation F1-score for all tested SVM hyperparameter configurations.

After that, Random Forest algorithm was utilized through hyperparameter tuning across different tree depths, numbers of estimators, and split criteria as explained in Section 2.2.3. The highest F1-score achieved was 0.67 (Table 2), obtained using number of trees = ٤00, tree depth = ١٠, and minimum split size = 5. This configuration also resulted in the best overall accuracy (0.67) with moderate training time (16.2 s), making it suitable for real-world scenarios. Interestingly, the model showed relatively stable behavior across depth and split settings, with deeper trees (tree depth = 10) often outperforming shallower ones, especially when paired with a larger number of estimators. In contrast, configurations with limited depth or smaller tree ensembles showed a noticeable decline in F1-score, dropping to around 0.62 in some cases.

Note that, while the addition of LBP to HOG features offered a slight boost in classification results, the overall success of Random Forest stemmed more from its ensemble structure and resistance to overfitting, rather than from sensitivity to feature types.

Table 1. Top-performing hyperparameter configurations of the SVM classifier with HOG features using the RBF kernel.

Kernel	C	γ	Mean Accuracy	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
rbf	10	scale	0.74	0.74	0.74	0.74	24.25	5.25
rbf	10	scale	0.74	0.74	0.74	0.74	24.41	5.28
rbf	1	scale	0.71	0.72	0.71	0.71	23.45	5.20
Poly (d=2)	10	scale	0.69	0.74	0.68	0.68	24.93	4.54

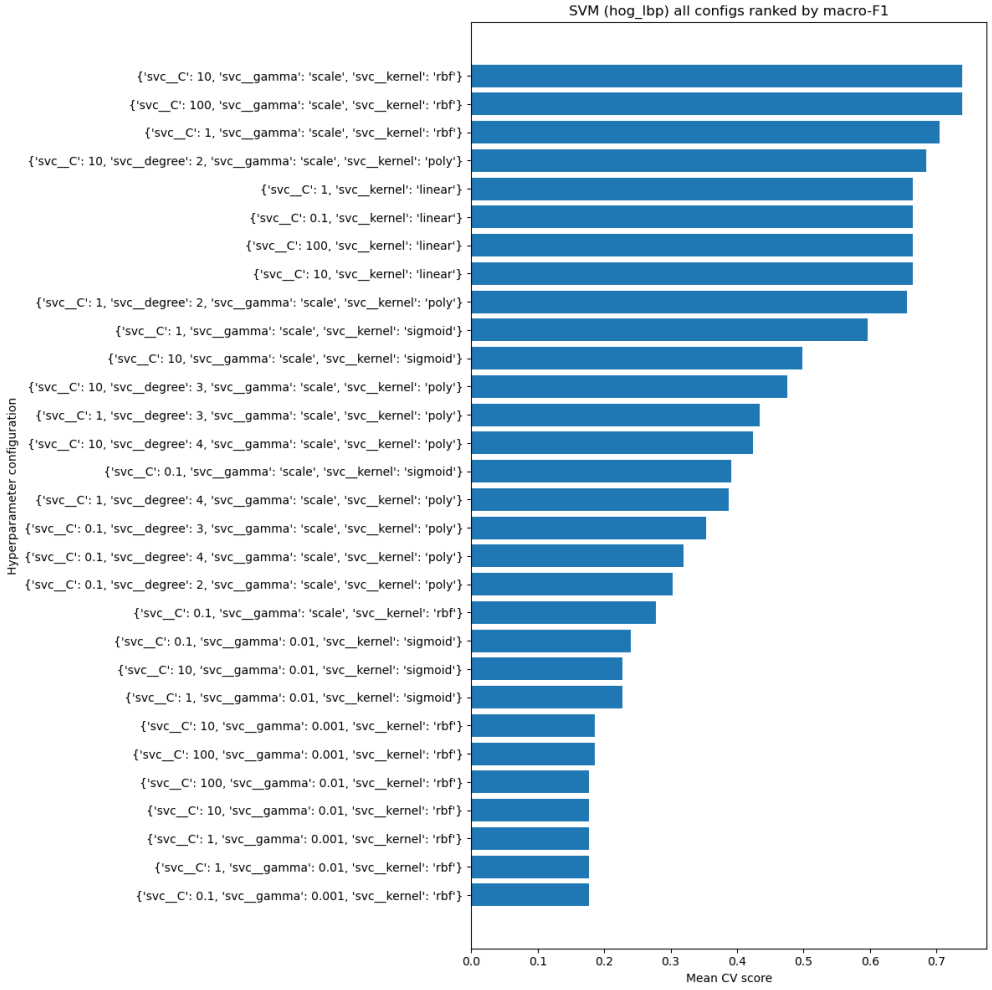


Figure 4. Mean F1-score from cross-validation for different SVM hyperparameter configurations.

Table 2. Selected Random Forest settings with highest macro F1-scores and corresponding runtime details.

Max depth	Min samples split	N estimator	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
10	5	400	0.69	0.66	0.67	16.20	2.62
None	10	200	0.68	0.66	0.67	1.56	1.13
30	10	200	0.68	0.66	0.67	1.57	0.99
20	2	400	0.68	0.66	0.67	18.01	2.86

Next, to evaluate K-nearest neighbors (KNN) classifier, various combinations of hyperparameters were tested as explained in Section 2.2.4. However, the classifier demonstrated overall weak classification results in the rooftop classification task. The best configuration, using 3 neighbors, Manhattan distance (p = 1), and distance-based weighting, achieved a macro F1-score of only 0.61 (Table 3). While distance-based weighting consistently led to better results than uniform weighting, and smaller k values generally produced slightly better recall and F1 scores, the differences remained minor.

To further analyze this behavior, an additional experiment was conducted in which the dimensionality of the feature vectors was reduced using Principal Component

Analysis (PCA) prior to KNN classification. While moderate dimensionality reduction (100–200 components) did not improve classification results, more aggressive reduction to 30–50 components led to improved generalization on the test set, indicating partial mitigation of distance concentration effects in high-dimensional feature spaces.

Overall, the weak classification results of KNN can largely be explained by the curse of dimensionality associated with high-dimensional feature representations, which reduces the effectiveness of distance-based similarity measures. Furthermore, KNN does not learn an explicit decision boundary and is highly sensitive to class overlap, intra-class variability, and class imbalance. Consequently, despite careful hyperparameter tuning, KNN proved to be an unsuitable choice for the rooftop classification task compared to margin-based and ensemble learning approaches.

Table 3. Top-performing KNN configurations using Manhattan distance (p = 1) and distance-based weighting

Number of Neighbours	p	weights	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
3	1	distance	0.61	0.61	0.61	0.39	3.51
5	1	distance	0.61	0.59	0.58	0.33	3.38
7	1	distance	0.60	0.57	0.56	0.29	3.11
9	1	distance	0.61	0.56	0.55	۰.۲۷	3.42

To evaluate the XGBoost classifier, an extensive grid search over various hyperparameters was conducted as explained in Section 2.2.5. However, the overall impact of feature combination on model classification results was marginal; both feature sets produced comparable results, indicating that XGBoost is relatively robust to the choice of descriptors. The best results were achieved using learning rate = 0.2, max depth = 7, and number of estimators = 100, yielding a macro F1-score of 0.70 and accuracy of 0.70 (Table 4). This configuration effectively balanced precision and recall, demonstrating the classifier's ability to generalize across classes. Table 4 also summarizes the top four performing configurations using the combined features. These include different trade-offs between depth and learning rate, with all maintaining strong F1-scores above 0.70. In general, deeper trees (max depth = 7) and moderate-to-high learning rates (0.1 or 0.2) consistently led to better results.

Across all configurations, higher learning rates and deeper trees enhanced recall, while the F1-score declined slightly with very shallow models or low learning rates. Notably, even the suboptimal configurations with lower parameter values still yielded moderate F1-scores above 0.58, reflecting the model's inherent stability. However, one clear drawback was the considerable computational cost. Fit times were notably high, even exceeding 800 seconds for top-performing models, and surpassing 1000 seconds in some deeper configurations. This highlights a significant trade-off between classification accuracy and computational efficiency when using XGBoost, particularly in scenarios with limited computational resources or when real-time inference is required.

To evaluate the classification results of the Logistic Regression classifier, a variety of hyperparameters were explored. The addition of LBP led to a slight improvement in accuracy, but the difference was minimal, suggesting that LR is relatively insensitive to these feature modifications. The best results were observed using combined features with C = 0.1, lbfgs solver, and 800 iterations, achieving a macro F1-score of 0.66 and an accuracy of 0.66 (Table 5). Among all configurations, moderate regularization (C = 0.1) consistently yielded better results across both feature sets. In contrast, overly small or large values of C degraded classification accuracy due to under or over regularization respectively.

While the differences between solvers were not drastic, lbfgs showed a slight edge in terms of speed and F1-score in the best settings. Training time for LR was relatively low (generally below 15 seconds), making it a computationally efficient option. However, even the top-performing models remained moderately accurate and may not capture complex class boundaries effectively, indicating a limited expressive capacity for this task.

Table 4. Top four performing XGBoost configurations, showing classification metrics and computation times.

Max depth	Learning Rate	N estimators	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
7	0.2	100	0.71	0.70	0.70	465	0.10
7	0.1	100	0.71	0.69	0.70	872	0.15
5	0.2	50	0.71	0.69	0.70	305	0.06
5	0.2	100	0.70	0.69	0.69	470	0.80

Table 5. Top-performing Logistic Regression configurations.

Max iter	C	solver	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
800	0.1	lbfgs	0.67	0.66	0.66	5.06	0.55
800	1	lbfgs	0.67	0.66	0.66	2.02	0.65
800	0.01	lbfgs	0.67	0.66	0.66	5.23	0.65
800	0.01	liblinear	0.65	0.65	0.65	23.10	0.65

To compare the overall results of the evaluated models, Table 6 presents the best results for each classifier. Among all models, SVM achieved the highest macro F1-score (73.8%) and overall accuracy (73.9%), confirming its strong discriminative capability for this classification task. XGBoost followed closely, delivering a competitive F1-score of 70.1%. However, this accuracy came at a substantial computational cost, as its training time was significantly higher than that of other models (465 seconds), making it less suitable for time-sensitive applications. Random Forest offered moderate classification accuracy and macro F1-score (F1 = 65.1%) with relatively low training and prediction times, presenting a balanced trade-off for practical scenarios. In contrast, KNN exhibited the lowest accuracy despite its low training cost. This highlights its limited ability to generalize effectively in this multi-class setting. Logistic Regression (LR) produced consistent but modest results (F1=63.9%), but its fast training and inference times may justify its use in resource-constrained environments.

Despite using two different handcrafted representations, the effect of adding LBP texture information to HOG was limited across the experiments. In the SVM grid search, HOG+LBP achieved a F1-score of 0.738 compared with 0.736 for HOG under 5-fold cross-validation, while both produced the same test-set accuracy and F1-score (0.74). The feature dimensionality increased only slightly (8100 for HOG versus 8118 for HOG+LBP), since LBP was encoded as a compact histogram. Overall, these results suggest that HOG already captures much of the discriminative structure for the considered roof classes, and that LBP provides only a small complementary gain under the present dataset and evaluation setting.

Overall, while SVM remains the most effective model in terms of classification accuracy and macro F1-score, XGBoost stands out as a close alternative when computational resources are not a limiting factor. To further understand the behavior of the two top-performing classifiers, SVM and XGBoost, their confusion matrices are presented in Figure 5. According to confusion matrices, SVM shows a relatively balanced and high true positive rate across all three classes, with minimal confusion. XGBoost, while still strong overall, exhibits slightly more confusion between class 1 and class 3, indicating potential difficulty in distinguishing between certain rooftop types.

Table 6. Summary of the best-performing configurations for each classification model.

Model	Mean Accuracy	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
SVM	73.7%	74.4%	73.7%	73.6%	24.25	5.25
RF	66%	69%	66%	67%	16.20	2.62
KNN	61%	61%	61%	61%	0.39	3.51
XGB	70.0%	71%	70%	70%	465	0.10
LR	64.3%	67%	66%	66%	5.06	0.55

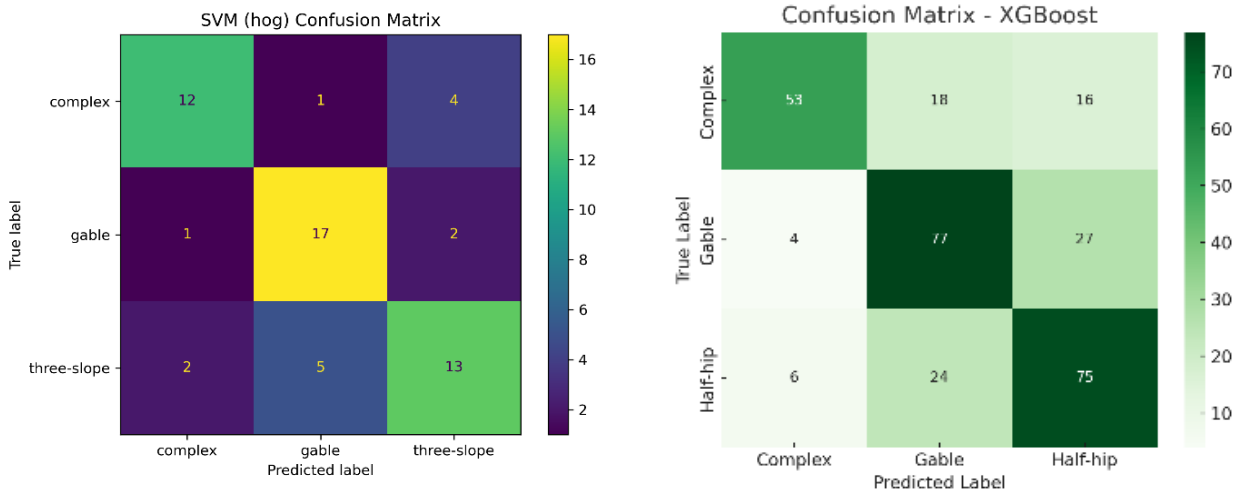


Figure 5. Confusion matrices for the best-performing SVM and XGBoost classifiers.

3.2. Results of Ensemble Modelling

To further improve classification accuracy, multiple ensemble methods were explored as explained in Section 2.2.7 (Table 7). The baseline approach, Hard Voting, All Models, achieved a macro F1-score of 0.74. After excluding weaker model (KNN), the macro F1-score improved slightly in the Hard Voting (Pruned Ensemble) method, reaching an F1-score of 0.75, which suggests that model pruning can reduce noise and enhance ensemble stability. The Weighted Voting approach, maintained a similar F1-score of 0.75. Although the gain in accuracy compared to hard voting was modest. The most significant improvement was observed in the Stacking Ensemble, which outperformed all other techniques with a macro F1-score of 0.79 and an overall accuracy of 79%. This approach effectively leveraged the diversity among base models by training a meta-learner to optimize the final prediction. Despite its computational overhead, stacking demonstrated superior generalization capability and predictive power.

These results confirm that ensemble techniques, particularly stacking, significantly enhance classification accuracy. The stacking ensemble attained the highest macro F1-score (0.79) and accuracy (79%) among all evaluated approaches. This classification accuracy and macro F1-score notably exceeds that of the best standalone model, SVM (F1 = 0.7ٰ), and also surpasses XGBoost (0.70), Random Forest (0.65), Logistic Regression (0.64), and KNN (0.61). However, this gain in accuracy came at a cost: the training time for the stacking model was substantially higher than all other approaches (947.24 seconds for training and 7.66 seconds for testing). This highlights a common trade-off in ensemble learning, where accuracy improvements through complex integration of models must be balanced against increased computational requirements.

Table 7. Comparison of different ensemble modelling strategies.

Method	Accuracy	Precision	Recall	F1
Hard Voting, All Models	74%	76%	74%	74%
Hard Voting (Pruned Ensemble)	75%	76%	75%	75%
Weighted Voting	75%	75%	74%	75%
Stacking Ensemble	**79%**	**79%**	**79%**	**79%**

3.3. Results of DL Methods

To evaluate the effectiveness of DL approaches in the classification task, two convolutional neural network architectures were implemented and assessed including CNN and MobileNetV2, as explained in Section 2.3.

CNN model achieved a test accuracy of 67% and a macro F1-score of 67%. As shown in Table 8, while this model was able to moderately capture class patterns, especially for the "complex" class, it failed to generalize well across all categories, particularly for "gable" and "half-hip", which exhibited lower recall and f1-scores. In contrast, the MobileNetV2 model, initialized with pretrained ImageNet weights, significantly outperformed the baseline CNN. After training, MobileNetV2 reached a test accuracy of 83.3% and a macro F1-score of 83%. This model demonstrated strong balance across all classes in both precision and recall, particularly achieving high recall for the "gable" class (91%) and overall superior f1-scores. The training time for MobileNetV2 (271.08 seconds) was substantially longer than the custom CNN (235.6 seconds), reflecting its larger architecture and increased data processing due to transfer learning. However, this increase in computational cost was offset by a significant gain in accuracy, making MobileNetV2 a far more suitable choice for real-world applications.

In addition to quantitative metrics, Figure 6 presents a spatial map of the classification outputs to demonstrate the geographic consistency of the predictions. The MobileNetV2 results are overlaid on the UAV orthophoto by joining polygon IDs with the corresponding cropped roof images. Visual inspection confirms that the model produces coherent roof-type patterns across neighboring buildings, while most errors occur near ambiguous roof geometries and occluded/shadowed areas, consistent with the confusion-matrix analysis.

Table 8. Comparison of DL Models on Roof Classification Task.

Model	Mean Accuracy	Mean Precision	Mean Recall	Mean F1	Fit Time	Score Time
CNN	67%	68%	68%	67%	235.6	0.19
MobileNetV2	83.3%	84%	83%	83%	271.08	1.35

Figure 6. Spatial visualization of roof-type classification over the study area. (a) Ground-truth labels derived from manual digitization and visual interpretation of the orthophoto. (b) MobileNetV2 predictions for the same rooftop polygons. Colors denote roof classes: complex (blue), gable (red), and half-hip (yellow), overlaid on the UAV orthophoto.

3.4. Ablation and Sensitivity Analysis

To better quantify how key design choices influence accuracy and computational cost, a set of ablation and sensitivity experiments was conducted. We examine (i) the sensitivity of the custom CNN to input image resolution, and (ii) the contribution of different data augmentation strategies when training the best-performing deep model (MobileNetV2). All experiments were performed under the same data split and evaluation protocol.

3.4.1. Resolution Sensitivity Analysis

To investigate the impact of input image resolution on the classification results, a resolution sensitivity analysis was conducted using the custom CNN. Input images were resized to three different spatial resolutions: 128×128, 192×192, and 224×224 pixels. For each resolution, the same network architecture, training protocol, and data augmentation strategy were applied. Each experiment was repeated three times with different random seeds, and the quantitative results are summarized in Table 9.

Table 9. Resolution sensitivity analysis for the best-performing CNN model.

Input Resolution	Macro F1 (Mean ± Std)	Complex Recall (Mean ± Std)	Training Time (Mean, s)
128 × 128	0.67 ± 0.05	0.88 ± 0.05	~250
192 × 192	0.59 ± 0.06	0.96 ± 0.06	~560
224 × 224	0.53 ± 0.05	0.94 ± 0.05	~766

The results indicate that increasing the input resolution does not lead to a consistent improvement in overall classification result as measured by macro F1-score. The 128×128 resolution achieved the highest average macro F1-score ($0.6^{\vee}$), while the 192×192 resolution resulted in a slightly lower macro F1-score (0.59). A preliminary experiment at 224×224 resolution showed further degradation in macro F1-score (0.53), alongside a substantial increase in training time.

In contrast, the recall of the Complex rooftop class improved with increasing resolution. Specifically, the mean recall increased from 0.88 at 128×128 to 0.96 at 192×192, suggesting that higher resolutions better preserve fine-grained structural details characteristic of complex rooftops. However, this improvement in class-specific recall did not translate into higher overall result, likely due to increased sensitivity to noise and overfitting effects affecting the simpler roof classes.

From a computational perspective, higher resolutions significantly increased training time. Considering both accuracy and efficiency, the 128×128 resolution provides the best trade-off between classification results and computational cost for the proposed CNN model.

3.4.2. Augmentation Ablation

To justify the use of different augmentation strategies across learning paradigms, an ablation study was conducted using MobileNetV2, which achieved the best overall result in the deep learning experiments. Four training configurations were evaluated while keeping the network architecture, optimizer, and training protocol fixed: (i) no data augmentation, (ii) discrete rotations limited to multiples of 90°, (iii) a combination of random rotation, translation, zoom, and horizontal flipping, and (iv) translation, zoom, and flipping without rotation.

As reported in Table 10 the configuration combining random rotation, translation, zoom, and flipping achieved the highest macro F1-score (0.83), outperforming both the non-augmented baseline (0.78) and the setup using only discrete rotations (0.81). Notably, excluding rotation from the augmentation pipeline resulted in a marked macro F1 drop (0.78), highlighting the importance of rotation invariance for robust rooftop type classification. These findings support the use of richer geometric transformations when training deep convolutional models on UAV-based rooftop imagery.

Ablation results in Table 10 are reported on the validation subset split from the training data using the same fixed train/test split; the final selected MobileNetV2 configuration is evaluated once on the held-out test set.

Table 10. Ablation study of data augmentation strategies for MobileNetV2.

Training Setup	Accuracy	Precision	Recall	F1-score	Fit Time (s)	Score Time (s)
No data augmentation	0.79	0.79	0.80	0.79	228.43	1.29
Discrete rotations (0°, 90°, 180°, 270°)	0.81	0.81	0.81	0.81	238.02	1.29
Random rotation + translation + zoom + flipping	0.83	0.84	0.83	0.83	271.08	1.35
Translation + zoom + flipping (no rotation)	0.79	0.81	0.78	0.78	268.55	1.30

4. Conclusions

This study conducted a comprehensive comparative evaluation of traditional ML and DL models for UAV-based rooftop type classification. It introduces a new UAV rooftop dataset collected over Rasht, Iran. By systematically analyzing both handcrafted feature approaches and deep feature learning strategies, the study provides quantitative insights into the trade-offs between accuracy, interpretability, and computational efficiency in urban rooftop recognition.

Results demonstrated that the integration of texture-based descriptors (LBP) with gradient-based features (HOG) achieved similar F1-score to HOG-only, with comparable training time under the same preprocessing and CV protocol. Ensembles can approach DL accuracy, but may incur higher training cost depending on tuning and feature extraction; however, inference can remain efficient. Finally, transfer learning with MobileNetV2 achieved the best overall results (F1 = 0.83) and provided balanced accuracy across all roof types, though with higher training time.

Overall, the study reveals a clear hierarchy in classification accuracy and computational efficiency: traditional ML models prioritize interpretability and fast training/inference, ensemble methods capitalize on complementary decision boundaries to deliver balanced, consistently improved accuracy, and deep learning, especially transfer learning architectures like MobileNetV2, achieves the strongest generalization on complex urban rooftops. Our results go beyond verifying that deep learning outperforms classical models, we provide a deployment-oriented benchmark and show that lightweight ML ensembles can achieve near-DL results.

A limitation of this study is that the experiments were conducted using rooftop data from a single urban area, and therefore cross-city generalization was not explicitly evaluated. Future work should aim to enhance the scalability of rooftop classification by addressing several key directions. First, expanding the dataset in both size and geographic coverage by incorporating rooftop imagery from cities with different architectural characteristics, together with appropriate model adaptation strategies such as fine-tuning or architectural adjustments, would help improve robustness when applying the model to new urban environments. Second, increasing the number of rooftop classes beyond the current three considered classes would allow for more detailed urban semantic mapping and better reflect real-world architectural variability. Lastly, integrating multimodal data sources, such as LiDAR point clouds and multispectral UAV imagery could significantly enhance classification accuracy, particularly in challenging conditions such as shadows or occlusions.

References

Ahmed Arafa, A. H., M. Radad, M. M. Badawy and N. El-Fishawy (2022). Logistic regression hyperparameter optimization for cancer classification. Menoufia Journal of Electronic Engineering Research 31(1): 1-8.

Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

Cai, Y., He, H., Yang, K., Fatholahi, S. N., Ma, L., Xu, L., & Li, J. (2021). A comparative study of deep learning approaches to rooftop detection in aerial images. Canadian Journal of Remote Sensing, 47(3), 413-431.

Castagno, J., & Atkins, E. (2018). Roof shape classification from LiDAR and satellite image data fusion using supervised learning. Sensors, 18(11), 3960.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining,

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning, 20, 273-297.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on information theory, 13(1), 21-27.

Meng, S., Soleimani-Babakamali, M. H., & Taciroglu, E. (2023). Automatic roof type classification through machine learning for regional wind risk assessment. arXiv preprint arXiv:2305.17315.

Mutreja, G., & Bittner, K. (2025). Efficient Building Roof Type Classification: A Domain-Specific Self-Supervised Approach. arXiv preprint arXiv:2503.22251.

Pan, Z., Xu, J., Guo, Y., Hu, Y., & Wang, G. (2020). Deep learning segmentation and classification for urban village using a worldview satellite image based on U-Net. Remote Sensing, 12(10), 1574.

Park, H.-A. (2013). An introduction to logistic regression: from basic concepts to interpretation with particular attention to nursing domain. Journal of Korean academy of nursing, 43(2), 154-164.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE conference on computer vision and pattern recognition,

Spasov, A., Petrova-Antonova, D., & Hristov, E. (2023). a Comparison Study on Deep Learning Models for Building Rooftop Classification. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 48, 47-53.

van Driel, B. Roof Type Classification in Aerial Imagery Using Convolutional Neural Networks.

Yildirim, M., & Karsli, F. (2025). A Novel Deep Learning Based Classification of Building Roof Types Using Point Cloud Data. Journal of the Indian Society of Remote Sensing, 53(3), 815-826.