VOL. 20 (1) (2023), XX-XXp-ISSN: 693-7554, e-ISSN:2654-3990

Landscape Image Classification Using Convolutional Neural Network on Intel Image Classification Dataset

Winarnie[1*], Hery Oktafiandi[2], Pebriyanti Panjaitan[3], M.Fajar Ramadhan[4], Yohanes[5]

[1,3,4] Teknik Informatika, Satu University, Bandung, West Java, Indonesia.
[2,5] Sistem Informasi, Satu University, Bandung, West Java, Indonesia

Abstract

Abstract Image classification is a part of computer vision and artificial intelligence. It helps computers recognize and categorize information on their own. This research is about creating a Convolutional Neural Network model for image classification. The goal is to recognize six types of urban landscapes using the Intel Image Classification dataset from Kaggle.The first step was to get the images ready. This included making all the images the size adding more data to the images and normalizing the pixels. This helps the model work better and not get too good at recognizing the training images. The dataset was split into two parts: 80% for training and 20% for testing.The Convolutional Neural Network model has four layers that look at the images layers that help reduce the amount of data and three layers that are fully connected with functions called ReLU and Softmax. What is new about this study is that it created a Convolutional Neural Network model that works well without using models that are already trained or transfer learning. The proposed CNN was developed without relying on pretrained models or transfer learning, resulting in a relatively compact architecture. Furthermore, the trained model was successfully converted into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats, demonstrating its deployment compatibility across cloud, web, and mobile software environments.

Keywords: *image classification, CNN, computer vision, Intel Image Classification, TensorFlow*

1. Introduction

Image classification is one of the fundamental tasks in computer vision and artificial intelligence, enabling computers to automatically recognize and categorize visual objects [1]. The rapid advancement of deep learning, particularly Convolutional Neural Networks (CNNs), has significantly improved image classification performance across various application domains, including environmental monitoring, autonomous systems, medical imaging, and smart city development. One of the major challenges in image classification lies in the complexity of visual information, where variations in illumination, viewpoint, object scale, background, color distribution, and object shape can substantially affect recognition performance. CNNs effectively address these challenges by automatically learning hierarchical spatial features through convolution and pooling operations, thereby eliminating the need for handcrafted feature extraction. [2], [3].

This study employs the Intel Image Classification Dataset obtained from Kaggle, which contains more than 14,000 color images representing six landscape categories: buildings, forests, glaciers, mountains, seas, and streets. The dataset has become a widely used benchmark for evaluating image classification algorithms because it contains considerable intra-class variation and inter-class similarity between natural and urban environments, making it suitable for assessing the robustness of CNN-based models[4],[5] .

Previous studies have successfully applied CNNs to landscape image classification using both conventional CNN architectures and deep pretrained models. Early CNN architectures such as LeNet and AlexNet demonstrated promising performance on benchmark datasets, while more recent studies have increasingly adopted transfer learning approaches using pretrained networks such as VGGNet, ResNet, EfficientNet, and MobileNetV2 to improve classification accuracy[6],[7]. Although these deep architectures generally achieve excellent predictive performance, they require substantially higher computational resources,

*Corresponding author. *E-mail address:* winarnie@univ.satu.ac.id
Received: 15 June 2026, Accepted: 30 July 2026 and available online 31 July 2026
DOI: https://doi.org/10.33751/komputasi.v23i2.112

including larger memory consumption, greater model complexity, and longer inference time. Consequently, their deployment on embedded systems, mobile devices, edge computing platforms, and other resource-constrained environments remains challenging Moreover, existing studies on the Intel Image Classification Dataset primarily emphasize maximizing classification accuracy through increasingly deeper architectures, while relatively little attention has been devoted to designing lightweight CNN models that can be trained entirely from scratch without relying on pretrained weights or transfer learning. This reveals an important research gap concerning the development of computationally efficient CNN architectures capable of maintaining competitive classification performance while reducing model complexity and improving deployment flexibility[8],[9].

To address this gap, this study proposes a lightweight CNN architecture specifically designed for multiclass landscape image classification using the Intel Image Classification Dataset. The proposed architecture consists of four convolutional blocks followed by three fully connected layers, resulting in approximately 2.28 million trainable parameters, which is considerably smaller than many commonly used deep CNN models. Unlike transfer learning approaches, the proposed network is trained entirely from scratch using data augmentation, normalization, Early Stopping, and learning-rate scheduling techniques to improve model generalization while avoiding overfitting.

The proposed model is comprehensively evaluated using multiple performance metrics, including accuracy, precision, recall, F1-score, and confusion matrix analysis to assess classification performance across all landscape categories. Furthermore, the trained model is successfully converted into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats, enabling deployment on cloud, web, mobile, and embedded platforms without requiring architectural modifications.

Accordingly, the main contributions of this research are as follows:

1. A lightweight CNN architecture for multiclass landscape image classification that achieves competitive classification performance while maintaining relatively low computational complexity.
2. Training entirely from scratch, eliminating dependence on pretrained models or transfer learning and making the proposed approach more suitable for environments with limited computational resources.
3. Comprehensive performance evaluation using accuracy, precision, recall, F1-score, and confusion matrix analysis to demonstrate balanced classification performance across all six landscape classes.
4. Multi-platform deployment capability through successful conversion of the trained model into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats, enabling practical implementation in cloud, mobile, web, and deployment compatibility applications.

Rather than attempting to outperform state-of-the-art deep learning architectures solely in terms of predictive accuracy, this research demonstrates that a carefully designed lightweight CNN trained from scratch can achieve competitive classification performance while providing lower computational complexity, improved deployment flexibility, and greater suitability for real-world applications operating under resource constraint

2. Methods

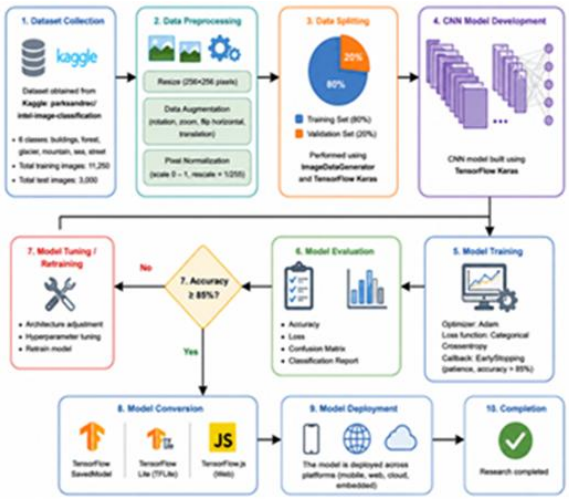


Figure 1. Flowchart of landscape image classification research with CNN

Figure 1 illustrates the overall workflow of the proposed landscape image classification system. The research consists of seven sequential stages: dataset acquisition, data preprocessing, dataset partitioning, CNN model development, model training, performance evaluation, and model deployment. The Intel Image

Classification Dataset was first collected from Kaggle, followed by image preprocessing, including resizing, normalization, and data augmentation. The processed dataset was then partitioned into training, validation, and testing subsets before being used to train the proposed lightweight CNN model. Finally, the trained model was evaluated using multiple classification metrics and converted into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats to support deployment on web, mobile, cloud, and embedded platforms. [10] [11], Finally we converted the model into formats like TensorFlow SavedModel, TensorFlow Lite and TensorFlow.js. This means the model can be used on platforms like web applications, mobile devices, cloud services and embedded systems[12]. The steps we took shown in Figure 1 helped us make a landscape image classification system that's accurate and works well.

2.1. Dataset

The Intel Image Classification Dataset was selected because it contains diverse visual characteristics, including variations in texture, illumination, color distribution, and object appearance. These characteristics make the dataset suitable for evaluating the robustness and generalization capability of CNN-based image classification models.

This study employed the Intel Image Classification Dataset obtained from Kaggle. The dataset contains 14,230 RGB images belonging to six landscape categories:

- Buildings
- Forest
- Glacier
- Mountain
- Sea
- Street

The dataset consists of 11,230 images designated for model development and 3,000 images reserved as an independent testing dataset. The testing images supplied with the original dataset were kept completely isolated during model development to ensure an unbiased evaluation of the final model.

2.2. Preprocessing

Data preprocessing aims to standardize the input representation while improving the convergence, stability, and generalization capability of the CNN model. [13].

1. Image Resizing
   All images were resized to 255 × 255 pixels to ensure consistent input dimensions for the CNN model.
2. Pixel Normalization
   Pixel values were normalized from the range [0,255] to [0,1] using a rescaling factor of 1/255. This process accelerates model convergence and improves training stability.
3. Data Augmentation
   To reduce overfitting and increase training data diversity, online data augmentation was applied using TensorFlow ImageDataGenerator with the following parameters:

Table 1. Data augmentation

Parameter	Value
Rotation	15°
Zoom Range	0.20
Width Shift	0.10
Height Shift	0.10
Shear Range	0.10
Horizontal Flip	Enabled
Rescaling	1/255

These changes make fake training pictures while keeping the meaning of the pictures the same. They do this by creating pictures that are similar to the original pictures, which is really useful for training. The pictures that are made look a lot like the pictures and they still have the same meaning as the original pictures, which is the same thing as the semantic content of the images. This is done by the transformations, which make the pictures and it is really helpful for the training of the pictures. The transformations are very important, for making the training samples and they do a great job of preserving the meaning of the pictures. [14].

2.3. Dataset Splitting

The original training subset containing 11,230 images was further divided into training and validation subsets using the validation_split mechanism provided by TensorFlow Keras.The partitioning scheme was:. [15].

Table 2. Data distribution

Dataset	Number of Images	Percentage
Training	8984	64%
Validation	2246	16%
Testing	3000	20%
Total	14230	100%

The testing dataset remained completely independent throughout the training and validation processes[16], No testing images were used during parameter optimization or model selection, thereby preventing information leakage and ensuring a fair evaluation of model performance[17].

The final testing dataset had 3,000 images that the model had never seen before. We did not use this dataset to train or validate the model. Instead we used it to see how well the final model worked and to get an idea of how well it could classify new landscape images. The way we divided up the data is shown in Table 1. We used the Intel Image Classification dataset, for this study. We made sure that the model was working well with the landscape categories: buildings, forest, glacier, mountain, sea and street. The dataset was a part of the study and we used it to train and test the model[18]. The use of separate training, validation, and testing datasets helps ensure that the developed CNN model can learn effectively, avoid overfitting, and achieve reliable performance when applied to unseen data. [19].

2.4. CNN Architecture

The proposed Convolutional Neural Network (CNN) was developed using the TensorFlow framework with the Keras API. Rather than adopting a deep transfer learning model, the architecture was designed from scratch to achieve a balance between classification performance and computational efficiency while maintaining a relatively small number of trainable parameters. This design objective makes the model demonstrates deployment compatibility in environments with limited computational resources.[20]. The network accepts RGB images with an input resolution of 255 × 255 × 3 pixels. Feature extraction is performed through four convolutional blocks arranged sequentially. Each convolutional layer employs the Rectified Linear Unit (ReLU) activation function to introduce non-linearity and accelerate convergence during training. A MaxPooling layer follows every convolutional layer to progressively reduce the spatial dimensions of the extracted feature maps while preserving the most informative visual characteristics.[21].

The convolutional layers use an increasing number of filters to enable hierarchical feature extraction. The first convolutional layer contains 32 filters, followed by 64 filters in the second layer, 128 filters in the third layer, and 64 filters in the fourth layer. This configuration allows the network to capture low-level visual patterns, such as edges and textures, in the initial layers while progressively learning higher-level semantic representations in the deeper layers.

After feature extraction, the multidimensional feature maps are transformed into a one-dimensional feature vector using a Flatten layer. The flattened features are subsequently processed by three fully connected (Dense) layers containing 128, 64, and 32 neurons, respectively. Each Dense layer employs the ReLU activation function to improve feature representation and nonlinear decision boundaries. The final classification layer consists of six output neurons with the Softmax activation function, producing the probability distribution for the six landscape categories: buildings, forest, glacier, mountain, sea, and street.

Overall, the proposed CNN contains approximately 2,275,854 trainable parameters. Compared with deeper pretrained architectures, the proposed model maintains a considerably lower architectural complexity while achieving reliable classification performance on the Intel Image Classification Dataset. This lightweight design also facilitates model conversion into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats for deployment across desktop, mobile, web, and embedded platforms.

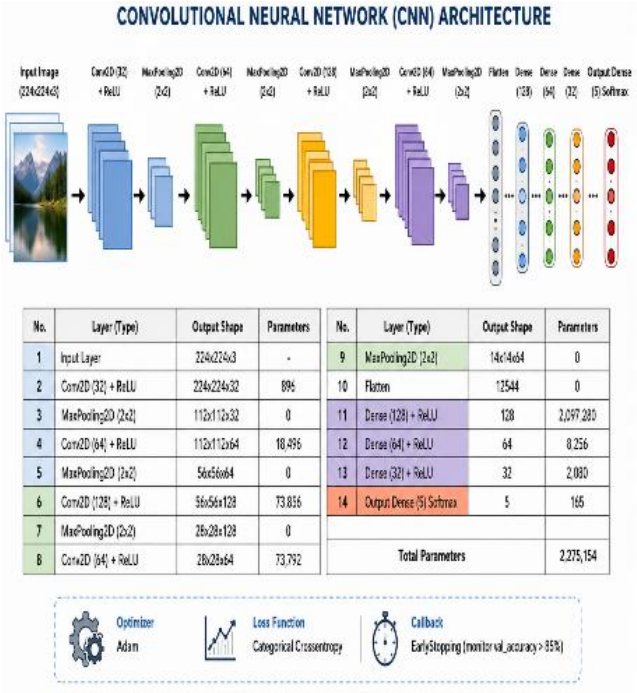
The CNN was trained using the Adam optimizer together with the categorical cross-entropy loss function, which is appropriate for multiclass image classification problems. The maximum number of training epochs was set to 20, with a batch size of 32. An EarlyStopping callback was implemented to terminate the training process automatically once the validation accuracy exceeded approximately 85% and no meaningful improvement was observed during subsequent iterations. This strategy reduced unnecessary computation while helping to prevent overfitting.

The proposed architecture was intentionally designed to balance predictive performance and computational efficiency. Instead of employing very deep pretrained networks, the model utilizes a moderate number of convolutional layers with progressively increasing filters to extract hierarchical image features while maintaining manageable computational complexity. The combination of image preprocessing, data augmentation, early stopping, and a compact CNN architecture contributes to stable training behaviour and good generalization performance on unseen landscape images.

Table 3. CNN architecture

Layer	Configuration	Output
Input	255 × 255 × 3 RGB Image	255 × 255 × 3
Conv2D-1	32 filters, ReLU	Feature Map
MaxPooling-1	2 × 2	Reduced Feature Map
Conv2D-2	64 filters, ReLU	Feature Map
MaxPooling-2	2 × 2	Reduced Feature Map
Conv2D-3	128 filters, ReLU	Feature Map
MaxPooling-3	2 × 2	Reduced Feature Map
Conv2D-4	64 filters, ReLU	Feature Map
MaxPooling-4	2 × 2	Reduced Feature Map
Flatten	Feature Vector	1-D Vector
Dense-1	128 neurons, ReLU	128
Dense-2	64 neurons, ReLU	64
Dense-3	32 neurons, ReLU	32
Output	6 neurons, Softmax	6 Classes

The Convolutional Neural Network (CNN) architecture used in this study is shown in Figure 2. Overall, the proposed architecture contains approximately 2,275,854 trainable parameters, making it substantially smaller than many deep transfer learning architectures while maintaining competitive classification performance.



No.	Layer (Type)	Output Shape	Parameters
1	Input Layer	224x224x3	-
2	Conv2D (32) + ReLU	224x224x32	896
3	MaxPooling2D (2x2)	112x112x32	0
4	Conv2D (64) + ReLU	112x112x64	18,496
5	MaxPooling2D (2x2)	56x56x64	0
6	Conv2D (128) + ReLU	56x56x128	73,856
7	MaxPooling2D (2x2)	28x28x128	0
8	Conv2D (64) + ReLU	28x28x64	73,792

No.	Layer (Type)	Output Shape	Parameters
9	MaxPooling2D (2x2)	14x14x64	0
10	Flatten	12544	0
11	Dense (128) + ReLU	128	2,097,280
12	Dense (64) + ReLU	64	8,256
13	Dense (32) + ReLU	32	2,080
14	Output Dense (5) Softmax	5	165
	Total Parameters		2,275,154

Figure 2. Convolutional Neural Network (CNN) architecture used in the research

2.5. Model Training

Model training was performed using the TensorFlow–Keras framework. All experiments were conducted in the Google Colaboratory (Google Colab) environment using GPU acceleration. The proposed CNN was trained, evaluated, and prepared for deployment using TensorFlow, including conversion into TensorFlow Lite and TensorFlow.js formats. The experiments utilized an NVIDIA Tesla T4 GPU provided by Google Colab. The hyperparameters used during model training are summarized in Table 4.

Table 4. CNN training hyperparameters

Hyperparameter	Value
Optimizer	Adam
Loss Function	Categorical Crossentropy
Batch Size	32
Maximum Epochs	20
Early Stopping	Enabled
Validation Split	20%

A custom TensorFlow callback was implemented to automatically stop training once both training accuracy and validation accuracy exceeded approximately 85%, thereby preventing unnecessary training epochs while maintaining satisfactory model performance. Although the maximum number of training epochs was set to 20, the training process terminated automatically before reaching the maximum epoch whenever the stopping criterion was satisfied. This strategy reduced unnecessary computation while minimizing the risk of overfitting.

2.6. Performance Evaluation

The trained CNN model was evaluated using several standard classification metrics derived from the confusion matrix, including:

- Accuracy
- Precision
- Recall
- F1-score
- Confusion Matrix

Training accuracy, validation accuracy, testing accuracy, training loss, validation loss, and testing loss were analyzed to assess the learning behavior and generalization capability of the proposed model.

The evaluation metrics are defined as follows:

- Accuracy measures the overall proportion of correctly classified images.
- Precision measures the proportion of correctly predicted positive instances.
- Recall measures the ability of the model to identify all relevant instances.
- F1-score represents the harmonic mean of precision and recall.

2.7. Deployment Model

To demonstrate deployment readiness, the trained CNN model was exported into three deployment formats:

- TensorFlow SavedModel
- TensorFlow Lite (TFLite)
- TensorFlow.js

These formats enable the model to be integrated into cloud-based systems, mobile applications, web applications, and multiple software deployment environments while maintaining classification performance.

3. Result and Discussion

3.1. Training Performance

The proposed CNN model was trained using the Intel Image Classification Dataset, which consists of 11,230 images for model development and 3,000 images for independent testing. During model development, the training subset was further divided into training and validation datasets using an 80:20 validation split. Training employed the Adam optimizer, categorical cross-entropy loss function, a batch size of 32, and an EarlyStopping strategy to reduce overfitting and unnecessary computation

The model demonstrated stable convergence throughout the training process. Although the maximum number of epochs was set to 20, the training process terminated automatically after the validation accuracy exceeded approximately 85% and no significant improvement was observed during subsequent iterations. This behavior indicates that the EarlyStopping mechanism successfully prevented excessive training while maintaining good predictive performance. The final evaluation results are summarized in Table 4.

Table 5. The final evaluation results

Performance Metric	Training	Testing
Accuracy	85.85%	85.47%
Loss	0.3778	0.4115

The small difference between the training and testing accuracies (0.38%) indicates that the proposed CNN generalized well to previously unseen data. Likewise, the relatively small increase in testing loss suggests that the model did not exhibit significant overfitting and maintained stable classification performance across different datasets. These observations demonstrate that the combination of data augmentation, normalization, and early stopping effectively improved the model's generalization capability

Figure 3. CNN Model Learning Curve on Intel Image Classification Dataset

Figure 3 illustrates the learning curves obtained during training. Both the training and validation accuracy increased consistently throughout the optimization process, while the corresponding loss values gradually decreased. The absence of large fluctuations between the two curves further indicates stable convergence and effective regularization during training.

The smooth behavior of both accuracy and loss curves indicates that the adopted data augmentation strategy effectively increased data diversity and reduced overfitting. Moreover, the EarlyStopping mechanism terminated training at an appropriate stage, avoiding unnecessary computation while preserving model generalization.

These findings indicate that the proposed lightweight CNN successfully learned discriminative landscape features while maintaining stable performance on previously unseen images. Despite having a considerably simpler architecture than deep transfer learning models such as VGGNet, ResNet, and DenseNet, the proposed model achieved competitive classification performance with substantially lower computational complexity.

3.2. Classification Performance

To obtain a more comprehensive evaluation than overall accuracy alone, the trained CNN model was assessed using precision, recall, and F1-score for each landscape category. The evaluation results are presented in Table 6.

Table 6. Model evaluation results

Class	Precision	Recall	F1-Score
buildings	0.89	0.85	0.87
forest	0.92	0.98	0.95
glacier	0.75	0.88	0.81
mountain	0.83	0.74	0.78
sea	0.87	0.82	0.85
street	0.90	0.87	0.88

Among the six categories, the forest class achieved the highest classification performance, with an F1-score of 0.95. This result can be attributed to the distinctive visual characteristics of forest scenes, particularly the dominant green color distribution and repetitive vegetation textures, which are more easily distinguished by convolutional feature extractors.

Conversely, the mountain category obtained the lowest F1-score (0.78). Mountain scenes frequently share visual characteristics with glacier images, including rocky surfaces, snow-covered regions, and similar color distributions. Such similarities reduce class separability and increase the likelihood of misclassification

Overall, all landscape categories achieved F1-scores exceeding 0.78, demonstrating that the proposed CNN provides balanced multiclass classification performance across different landscape types. The macro-average F1-score of approximately 0.86 further confirms the robustness of the proposed architecture.

3.3. Confusion Matrix

The confusion matrix presented in Figure 4 provides a detailed visualization of the classification performance of the proposed CNN model across the six landscape categories. Most predictions are concentrated along the main diagonal of the matrix, indicating that the majority of testing images were correctly classified. This observation is consistent with the overall testing accuracy of 85.47%, confirming that the proposed CNN successfully learned representative visual features from the Intel Image Classification Dataset.

Despite the satisfactory overall performance, several categories exhibited relatively frequent misclassification due to similarities in their visual characteristics. The largest source of classification error occurred between the mountain and glacier classes. Both categories commonly contain snow-covered surfaces, rocky textures, and similar edge structures, resulting in highly overlapping visual representations. Moreover, variations in illumination, seasonal conditions, and camera viewpoints further reduce the discriminative capability of appearance-based feature extraction, making these two categories inherently difficult to distinguish.

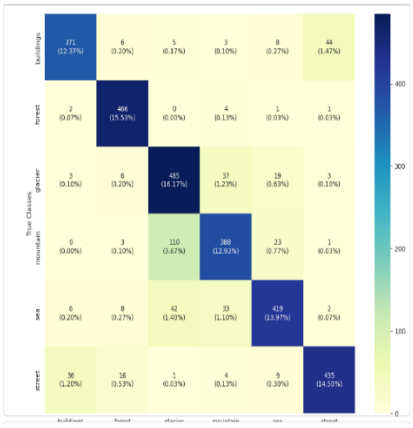
Another notable confusion was observed between the sea and glacier categories. Although these classes represent different natural environments, they frequently share dominant blue and white color distributions. Images captured under bright lighting conditions may contain reflections on water surfaces that resemble snow or ice regions, leading the CNN to generate similar feature representations. This finding suggests that color information alone is insufficient for reliably separating these categories, and additional contextual or semantic features may improve classification performance.

Misclassification also occurred between the buildings and street classes. Urban landscape images often

include multiple objects within the same scene, such as roads, buildings, vehicles, sidewalks, and other architectural elements. Consequently, the extracted features may simultaneously represent characteristics of both classes, increasing the probability of incorrect predictions. This ambiguity is expected because the semantic boundaries between urban environments are generally less distinct than those observed in natural landscape categories.

Conversely, the forest category achieved the highest classification accuracy, consistent with its superior precision, recall, and F1-score. Forest images exhibit highly distinctive visual characteristics, including dense vegetation, dominant green color distributions, and repetitive natural textures. These features are spatially consistent across different forest scenes, enabling the convolutional layers to learn discriminative representations more effectively than for visually overlapping categories such as mountains and glaciers.

Overall, the confusion matrix demonstrates that the proposed lightweight CNN provides reliable multiclass classification performance while revealing several challenging category pairs that remain difficult to distinguish because of inherent visual similarities within the dataset. These observations indicate potential directions for future improvement, including the incorporation of attention mechanisms, feature fusion strategies, or transfer learning approaches to enhance discrimination between visually similar landscape categories.



	precision	recall	f1-score	support
buildings	0.89	0.85	0.87	437
forest	0.92	0.98	0.95	474
glacier	0.75	0.88	0.81	553
mountain	0.83	0.74	0.78	525
sea	0.87	0.82	0.85	510
street	0.90	0.87	0.88	501
accuracy			0.85	3000
macro avg	0.86	0.86	0.86	3000
weighted avg	0.86	0.85	0.85	3000

Figure 4. Confusion matrix of the proposed CNN model

3.4. Discussion

3.4.1. Model Performance Analysis

The proposed lightweight CNN achieved a testing accuracy of 85.47%, demonstrating that competitive classification performance can be obtained without employing deep pretrained architectures or transfer learning. Several factors contributed to this performance, including the use of data augmentation techniques, hierarchical feature extraction through convolution and max-pooling layers, ReLU activation functions, and the EarlyStopping strategy. Together, these components improved feature learning while reducing overfitting and computational cost.

3.4.2. Classification Error Analysis

Most classification errors occurred between glacier and mountain images because both classes contain highly similar visual characteristics, including snow-covered surfaces, rocky textures, and comparable color distributions. Likewise, sea images were occasionally confused with glacier scenes due to similar blue-white color compositions and lighting reflections. These observations suggest that incorporating attention mechanisms or additional contextual information may further improve feature discrimination.

3.4.3. Overfitting and Generalization Analysis

The small difference between training accuracy (85.85%) and testing accuracy (85.47%), together with the decreasing training and validation loss curves, indicates that the proposed CNN generalized well to unseen data. No evidence of significant overfitting was observed, demonstrating that the adopted preprocessing techniques, data augmentation, and EarlyStopping effectively improved model robustness.

3.5. Comparison with Previous Studies

To better position the contribution of this study, the proposed lightweight CNN architecture was qualitatively compared with representative CNN-based image classification studies reported in the literature.

Recent studies have demonstrated that transfer learning models such as VGG16, ResNet50, MobileNetV2, and EfficientNet generally achieve high classification accuracy because they leverage feature representations learned from large-scale datasets, such as ImageNet. Consequently, these architectures have become widely adopted for landscape image classification and other computer vision applications.

Despite their strong predictive capability, pretrained deep learning models usually contain substantially larger numbers of trainable parameters and require greater computational resources for both training and inference. Their implementation often involves higher memory consumption, longer inference times, and increased computational complexity, which may limit deployment flexibility in environments with constrained computing resources. In contrast, the CNN proposed in this study was intentionally designed as a lightweight architecture trained entirely from scratch without relying on pretrained weights or transfer learning. The architecture consists of four convolutional blocks followed by three fully connected layers, resulting in approximately 2.28 million trainable parameters while achieving a testing accuracy of 85.47% on the Intel Image Classification Dataset.

Unlike many previous studies that primarily focus on maximizing classification accuracy, this research emphasizes the balance between predictive performance, architectural simplicity, and deployment compatibility. The proposed CNN was successfully exported into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats, demonstrating that the trained model can be integrated into cloud-based systems, web applications, mobile applications, and embedded software environments without requiring architectural modifications. This deployment-oriented contribution extends beyond predictive performance by improving the practical applicability of the proposed model across multiple software platforms.

It should be emphasized that this study does not experimentally compare the proposed CNN with MobileNetV2, VGG16, ResNet50, or other transfer learning architectures under identical experimental conditions. Therefore, no direct conclusions regarding relative accuracy, computational efficiency, or inference speed can be drawn. Instead, this study demonstrates that a carefully designed lightweight CNN trained from scratch can achieve competitive classification performance while maintaining lower architectural complexity and broad deployment compatibility. Future research should conduct comprehensive benchmarking against modern transfer learning models using identical datasets, preprocessing strategies, and evaluation protocols to provide a fair and quantitative comparison.

Table 7. Qualitative Comparison between the Proposed CNN and Previous Studies

Aspect	Transfer Learning Models (VGG16, ResNet50, MobileNetV2, EfficientNet)	Proposed CNN
Learning strategy	Transfer learning using pretrained weights	CNN trained from scratch
Network complexity	Generally high	Relatively compact (≈2.28 million parameters)
Pretrained weights	Required	Not required
Dataset	Intel Image Classification Dataset or similar landscape datasets	Intel Image Classification Dataset
Primary objective	Maximizing classification accuracy	Balancing classification accuracy, architectural simplicity, and deployment compatibility
Model deployment	Rarely discussed	Successfully converted into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js
Experimental comparison	Reported individually in respective studies	No direct benchmarking with transfer learning models (acknowledged as a limitation)

3.6. Model Deployment Analysis

Unlike many previous CNN studies that only report classification accuracy, this work further demonstrates deployment readiness by exporting the trained model into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats. These deployment formats enable implementation across cloud servers, mobile applications, web browsers, and embedded devices while preserving the learned network parameters. Consequently, the proposed CNN provides greater practical applicability beyond experimental evaluation.

3.7. Research Implications

The findings of this study demonstrate that lightweight CNN architectures remain a viable solution for practical landscape image classification tasks. The proposed model provides an attractive alternative for applications requiring efficient computation, including environmental monitoring, intelligent tourism systems, geographic information systems, and edge AI applications. Future research may further improve classification performance by integrating attention mechanisms, advanced data augmentation techniques, or hybrid CNN-transformer architectures while maintaining computational efficien

4. Conclusion

This study successfully developed a lightweight Convolutional Neural Network (CNN) model for multiclass landscape image classification using the Intel Image Classification Dataset. The proposed architecture consists of four convolutional layers followed by three fully connected layers, resulting in approximately 2.28 million trainable parameters. Despite its relatively simple architecture and the absence of pretrained weights or transfer learning, the model effectively learned representative visual features and achieved competitive classification performance while maintaining low computational complexity.

Experimental evaluation demonstrated that the proposed CNN achieved a training accuracy of 85.85% and a testing accuracy of 85.47%, with only a small difference between training and testing performance. This indicates that the model generalized well to previously unseen images without significant overfitting. The combination of image resizing, pixel normalization, data augmentation, and the EarlyStopping strategy effectively improved model robustness and learning stability throughout the training process.

The classification results further revealed that the proposed model performed consistently across the six landscape categories. The forest category achieved the highest precision, recall, and F1-score due to its distinctive vegetation patterns and dominant green color distribution, whereas the mountain category produced the lowest classification performance because of its strong visual similarity to glacier images. The confusion matrix further confirmed that most classification errors occurred between the mountain and glacier categories, followed by sea and glacier, and buildings and street, indicating that similarities in texture, color distribution, and scene composition remain challenging for appearance-based CNN models.

Compared with many recent studies that employ deep pretrained architectures such as MobileNetV2, ResNet, EfficientNet, and VGG through transfer learning, the proposed model emphasizes architectural simplicity and computational efficiency while maintaining competitive classification accuracy. Although transfer learning models generally achieve higher predictive performance, they require substantially greater computational resources, larger memory capacity, and longer inference time. Therefore, the proposed lightweight CNN provides an attractive alternative for applications operating in resource-constrained environments. Since pretrained architectures were not experimentally evaluated under identical conditions in this study, these comparisons should be interpreted qualitatively rather than as direct performance comparisons.

Another important contribution of this research is the successful deployment of the trained model into TensorFlow SavedModel, TensorFlow Lite, and TensorFlow.js formats. This demonstrates that the same trained network can be deployed across multiple computing platforms, including cloud services, web applications, mobile devices, and embedded systems, thereby increasing its practical applicability for real-world intelligent image classification applications.

Nevertheless, several limitations should be acknowledged. First, the proposed CNN was evaluated using only the Intel Image Classification Dataset, and its generalization capability on other landscape image datasets has not yet been investigated. Second, this study did not perform experimental comparisons with state-of-the-art transfer learning models under identical training conditions. Third, although deployment readiness was demonstrated through model conversion into multiple formats, quantitative evaluations of inference time, memory consumption, and energy efficiency were beyond the scope of this work.

Future research may address these limitations by conducting comprehensive benchmarking against modern deep learning architectures, including MobileNetV2, EfficientNet, ResNet, and Vision Transformer models, using identical experimental settings. Additional improvements may also be achieved by incorporating attention mechanisms, feature fusion strategies, hybrid CNN–Transformer architectures, or advanced data augmentation techniques to improve discrimination between visually similar landscape categories such as glaciers and mountains. Furthermore, evaluating inference latency, model compression techniques, and energy consumption on mobile and embedded devices would provide stronger evidence regarding the suitability of lightweight CNN architectures for real-world edge computing applications.

Overall, this study demonstrates that a carefully designed lightweight CNN trained entirely from scratch can provide reliable multiclass landscape image classification while maintaining moderate computational complexity and broad deployment flexibility. Rather than focusing solely on maximizing classification accuracy, this research highlights the importance of balancing predictive performance, computational efficiency, and practical deployment, thereby offering an effective alternative for intelligent image classification systems operating in environments with limited computational resources.

References

[1] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, "A review of convolutional neural networks in computer vision," Artif. Intell. Rev., vol. 57, no. 4, p. 99, Mar. 2024, doi: 10.1007/s10462-024-10721-6.

[2] T. H. Qur'ani, S. Bahri, and G. Gunawan, "Optimalisasi Model Convolutional Neural Network dengan Arsitektur MobileNetV2 Pada Sistem Otomatis Deteksi Penyakit Tanaman Jagung Berdasarkan Citra Daun," Simpatik J. Sist. Inf. dan Inform., vol. 5, no. 2, pp. 82–91, Dec. 2025, doi: 10.31294/simpatik.v5i2.9976.

[3] M. Muslihati, S. Sahibu, and I. Taufik, "Implementasi Algoritma Convolutional Neural Network untuk Klasifikasi Jenis Sampah Organik dan Non Organik," MALCOM Indones. J. Mach. Learn. Comput. Sci., vol. 4, no. 3, pp. 840–852, May 2024, doi: 10.57152/malcom.v4i3.1346.

[4] E. Yulisara et al., "Comparative Study of Convolutional Neural Network Architectures and Optimizers for Flower Image Classification," Public Res. J. Eng. Data Technol. Comput. Sci., vol. 3, no. 2, pp. 126–137, 2026, [Online]. Available: https://journal.irpi.or.id/index.php/predatecs/article/view/2110?utm_source=chatgpt.com

[5] H. Oktafiandi and Winarnie, "Implementasi Algoritma Convolution Neural Network pada Klasifikasi Limbah dengan Arsitektur MobileNet," in Prosiding Wijayakusuma National Conference (WiNCo), Nov. 2023, pp. 99–106. doi: 10.56655/winco.v4i1.196.

[6] L. Chen, S. Li, Q. Bai, J. Yang, S. Jiang, and Y. Miao, "Review of Image Classification Algorithms Based on Convolutional Neural Networks," Remote Sens., vol. 13, no. 22, p. 4712, Nov. 2021, doi: 10.3390/rs13224712.

[7] W. Hua, C. Li, and X. Wang, "Review of Convolutional Neural Network Models and Image Classification," Acad. J. Sci. Technol., vol. 10, no. 3, pp. 178–184, Apr. 2024, doi: 10.54097/644jqv20.

[8] K. R. Ariawan, A. A. G. Ekayana, I. P. Y. Indrawan, K. R. Winatha, and I. N. A. F. Setiawan, "Performance Comparasion of DenseNet-121 and MobileNetV2 for Cacao Fruit Disease Image Classification," Indones. J. Data Sci., vol. 6, no. 1, pp. 30–38, 2025, doi: 10.56705/ijodas.v6i1.233.

[9] A. Howard et al., "Searching for mobileNetV3," Proc. IEEE Int. Conf. Comput. Vis., pp. 1314–1324, 2019, doi: 10.1109/ICCV.2019.00140.

[10] D. Carlos, D. E. Herwindiati, and C. Lubis, "Implementasi Algoritma Convolutional Neural Networks Untuk Klasifikasi Jenis Cat Tembok Menggunakan Arsitektur MobileNet," Build. Informatics, Technol. Sci., vol. 6, no. 1, pp. 395–402, Jun. 2024, doi: 10.47065/bits.v6i1.5322.

[11] Jeki, M. Widyaningsih, and L. Rusdiana, "Klasifikasi Citra Hewan Khas Suku Dayak Menggunakan Convolution Neural Network," Sainteks, vol. 22, no. 1, pp. 87–98, Apr. 2025, doi: 10.30595/sainteks.v22i1.25874.

[12] A. Prayoga, Maimunah, P. Sukmasetya, Muhammad Resa Arif Yudianto, and Rofi Abul Hasani, "Arsitektur Convolutional Neural Network untuk Model Klasifikasi Citra Batik Yogyakarta," J. Appl. Comput. Sci. Technol., vol. 4, no. 2, pp. 82–89, Nov. 2023, doi: 10.52158/jacost.v4i2.486.

[13] A. Hassan, M. Refaat, and A. Hemeida, "Image classification based deep learning: A Review," Aswan Univ. J. Sci. Technol., vol. 2, no. 1, pp. 11–35, Jun. 2022, doi: 10.21608/aujst.2022.259887.

[14] L. Alzubaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," J. Big Data, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[15] K. Athalia, T. Tiffany, K. A. D. Setiawan, B. Ferrari, and C. Lubis, "Classification of diseases in snake plants using convolutional neural network," J. Comput. Networks, Archit. High Perform. Comput., vol. 6, no. 1, pp. 55–66, Dec. 2023, doi: 10.47709/cnahpc.v6i1.3201.

[16] H. Chen, G. Zhou, W. He, X. Duan, and H. Jiang, "Classification and identification of agricultural products based on improved MobileNetV2," Sci. Rep., vol. 14, no. 1, pp. 1–14, 2024, doi: 10.1038/s41598-024-53349-w.

[17] W. P. Kusumo and C. S. K. Aditya, "Klasifikasi Citra Makanan Berdasarkan Asal Daerah Menggunakan Convolutional Neural Network," Techno.Com, vol. 23, no. 1, pp. 87–95, Feb. 2024, doi: 10.62411/tc.v23i1.9735.

[18] Y. Octavianus and D. Avianto, “Modifikasi Arsitektur dalam Convolutional Neural Network untuk Klasifikasi Batik Lampung dan Batik Yogyakarta,” J. Indones. Manaj. Inform. dan Komun., vol. 6, no. 1, pp. 522–535, Jan. 2025, doi: 10.35870/jimik.v6i1.1223.

[19] A. R. Dani and I. Handayani, “Klasifikasi Motif Batik Yogyakarta Menggunakan Metode GLCM dan CNN,” J. Teknol. Terpadu, vol. 10, no. 2, pp. 142–156, Dec. 2024, doi: 10.54914/jtt.v10i2.1451.

[20] T. Barman and S. Susan, “Multi-Label Remote Sensing Image Classification using MobileNetV2,” in 2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT), Jun. 2024, pp. 1–4. doi: 10.1109/ICCCNT61001.2024.10725506.

[21] R. A. Hermawan, I. Taufik, and Y. Aditia Gerhana, “Klasifikasi Citra Ras Kucing Berbasis CNN dengan Metode MobileNet-V2,” Intern. (Information Syst. Journal), vol. 8, no. 1, pp. 70–84, Jun. 2025, doi: 10.32627/internal.v8i1.1390.