Intel Image Scene Classification of Landscapes

Second Assignment for the Machine Learning Topics Course

José Mendes
DETI
University of Aveiro
NMEC: 107188
Contribution: 50%

Diana Miranda
DETI
University of Aveiro
NMEC: 107457
Contribution: 50%

***Abstract*—Over the past decade, image classification has become a critical technology, supporting complex tasks such as medical diagnostics and autonomous robotics. Within this field, scene image classification has gained considerable attention. In this paper, we review the current state of the art in this subfield and utilize the Intel Image Classification dataset, which contains images of diverse natural scenes from around the globe, to develop four distinct CNN models. These models are designed to accurately identify various types of scenery depicted in images, including buildings, forests, glaciers, mountains, seas, and streets.**

***Index Terms*—Intel Image Classification, Image Scene Classification, Machine Learning, Multiclass Classification, Neural Network, Deep Learning, CNN, Convolutional Neural Network**

I. Introduction

Scene image classification is an increasingly vital aspect of computer vision, playing a crucial role in numerous applications, including medical diagnostics, autonomous navigation, environmental monitoring, digital content recommendation, and tourism planning. Unlike basic image classification, which typically involves identifying a limited number of objects within an image, scene classification requires the analysis of complex and varied visual environments.

This paper aims to achieve two primary goals. First, we provide a comprehensive review of the current advancements in scene image classification, analyzing and comparing five relevant studies in this area. Second, we develop, optimize, and evaluate four distinct Convolutional Neural Networks (CNNs) using the Intel Image Classification dataset. This dataset encompasses a wide range of natural scenes from different parts of the world, presenting a challenging multiclass classification problem.

We aim to demonstrate the effectiveness of various machine learning techniques in accurately predicting the type of scenery depicted in images, such as buildings, forests, glaciers, mountains, seas, and streets. The paper concludes with a detailed presentation of our results, comparing them side-by-side along with the insights gained from our experiments.

II. State of The Art

A comprehensive state-of-the-art review was performed to find the most suitable models for application to this dataset and identify the type of studies that had already been conducted. The results of these searches present five relevant studies below:

A. Fine-tuned EfficientNet and MobileNetV2 Models for Intel Images Classification

In the research "Fine-tuned EfficientNet and MobileNetV2 Models for Intel Images Classification" [4], the performance of two deep learning models, MobileNetV2 and EfficientNet, was tested in terms of image sorting in Intel collections [2]. They were tested at learning rates of 0.01 and 0.09 and MobileNetV2 reached the 50th training session with a great accuracy of 99.67%, while the best value reached by EfficientNet was 92.11%, like it's shown in Figure 1. Generally, the findings from this study underline the importance of the right selection of the model and its settings during fine-tuning to get a good image classification accuracy. Consequently, findings can be transferred to work with other applications connected with object recognition and pattern detection. The study thus emphasizes the importance of careful model selection concerning computational resources, task requirements, and data characteristics, providing a foundation for further advances in deep learning for image labelling.

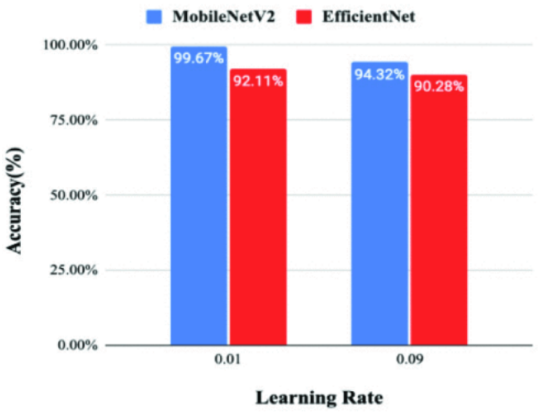

Fig. 1. Accuracy of the MobileNetV2 and EfficientNet model for intel image classification [4]

B. Intel CNN vs VGG16 vs MobileNet

The study "Intel CNN vs VGG16 vs MobileNet" [7] shows the performance of CNN, VGG16, and MobileNet on the same

dataset used by this paper. The study began with a CNN and with 15 epochs, the model reached 76% accuracy with the validation data (Figure 2 Right side).

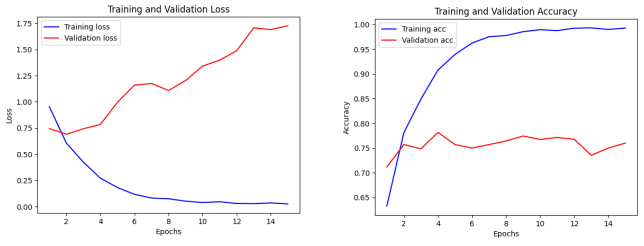

Fig. 2. Comparison between training and validation data: Loss (Left side) and Accuracy (Right side) with CNN Model [7]

The study then moved on to the VGG16 model, which also trained for 15 epochs and achieved an 86% accuracy with the validation data (Figure 3 Right side).

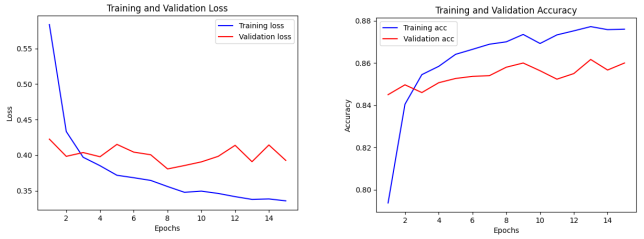

Fig. 3. Comparison between training and validation data: Loss (Left side) and Accuracy (Right side) with VGG16 Model [7]

Finally, the MobileNet model was tested, and configured similarly to the VGG16 model. It achieved 88% accuracy on the validation data (Figure 4 Right side).

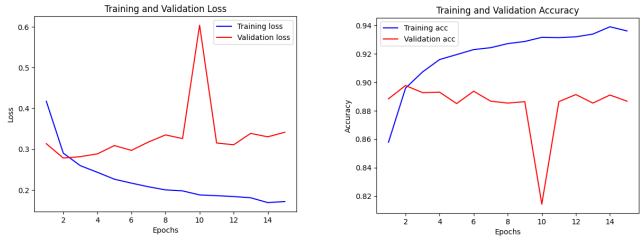

Fig. 4. Comparison between training and validation data: Loss (Left side) and Accuracy (Right side) with MobileNet Model [7]

Based on the visual representations of the graphs, the study concludes that the CNN model seems to be underfitting. In contrast, VGG16 seems to be doing exceptionally well, and MobileNet is showing signs of overfitting. From all the performance that can be garnered from all three models, VGG16 seems to be the best approach

C. RepConv: A novel architecture for image scene classification on Intel scenes dataset

This study, "RepConv: A Novel Architecture for Image Scene Classification on the Intel Scenes Dataset" [6], forms a contribution to the problem of image understanding and scene classification by presenting the novel machine learning model RepConv, fast convergence, and efficient performance without data-dependent weights. The RepConv model has been experimentally tested on the Intel scenes dataset and outperformed four benchmark models, achieving 93.55 ± 0.11 for training and 75.54 ± 0.14 for validation data with fewer epochs and parameters. Further, the study has introduced a new binary classification problem through re-categorization of the dataset into natural scenes (forest, glacier, mountain, and sea) and real scenes (building and streets), which has shown unprecedented accuracies of 98.08 ± 0.05 for training and 92.70 ± 0.08 for validation data (Figure 5).

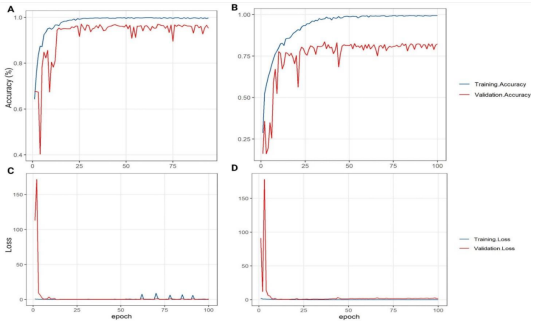

Fig. 5. Training and validation accuracy of the RepConv model. A: Accuracy of natural scenes vs real scenes. B: Accuracy of 6 classes classification (building; forest; glacier; mountain; sea; street). C: Loss of natural scenes vs real scenes. D: Loss of 6 classes classification. [6]

D. Intel Image classification - VGG16 - Val_Acc: 92.4%

The study "Intel Image Classification - VGG16 - Val_Acc: 92.4%" [3] also explores the VGG16 model on the Intel image classification dataset. In the beginning, the model reached 84% accuracy through data validation. After applying fine-tuning, the accuracy increased up to 92% (Figure 6). The study concludes that even though the model has 92% accuracy in the validation data, the training and validation accuracy graph shows overfitting, which should be mitigated by increasing the Dropout layer value.

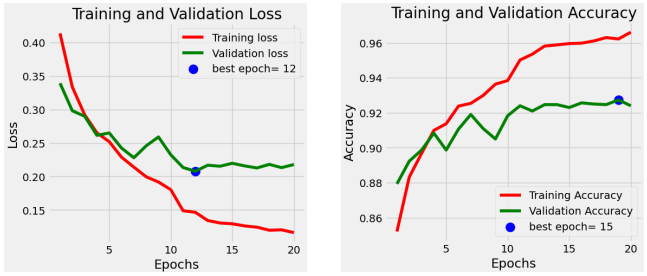

Fig. 6. Comparison between training and validation data: Loss (Left side) and Accuracy (Right side) [3]

E. Multi-Class Image Classification using Alexnet Deep Learning Network implemented in Keras API

In this research, "Multi-Class Image Classification using Alexnet Deep Learning Network implemented in Keras API"

[8], a CNN using the AlexNet architecture is created and tested using the Intel Image Classification dataset. This model was trained using the Adam optimizer and the "categorical_crossentropy" loss function over 50 epochs. The model obtained a 98.33% accuracy with the training data and 87.20% accuracy with the test data. Validation data was not used, and thus, graphs of the evolution in training over the epochs were not provided, but by the accuracy values of the training and test data alone, we can conclude that the model is overfitted since the gap between this two accuracies is big.

In the end, the trained model is tested using some images and the results seem to be as expected for a model with an 87.20% of accuracy. The author concludes that better and more complex networks such as VGG16, VGG19 and ResNets are worth trying and should give better accuracy than the one obtained in this research.

III. Intel Image Classification

A. Data Description

The Intel Image Classification dataset, which can be found on the Kaggle website, contains around 25k images of size 150x150. These images are divided into six classes: buildings, forest, glacier, mountain, sea, and street [2] (Figure 7).

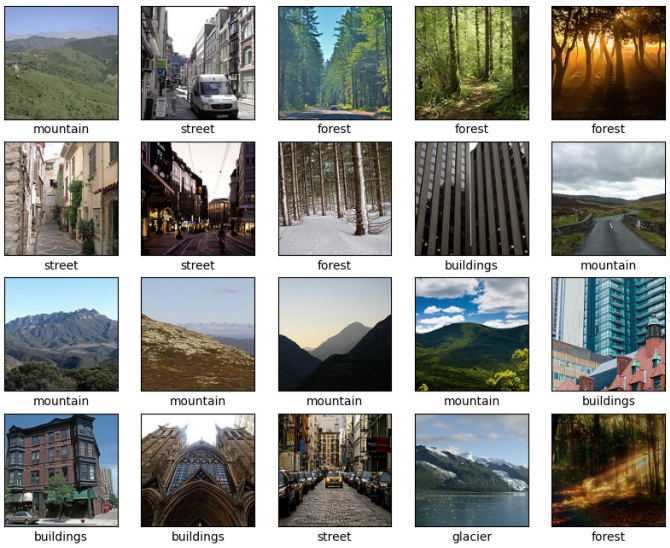

Fig. 7. Example of 20 Images from the Training Dataset

The folders containing the images are one for the training dataset, with around 14,000 images, the test dataset folder with around 3,000 images, and the prediction folder containing about 7,000 images. The images under the prediction folder do not have any labels (Figure 8).

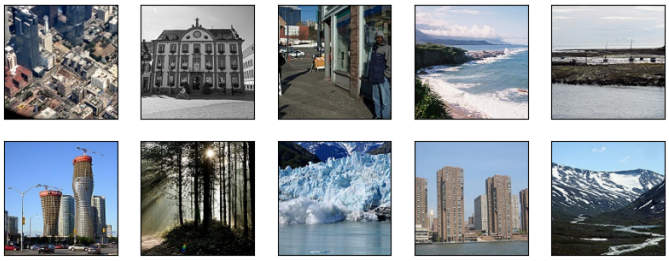

Fig. 8. Examples from the Prediction Dataset

B. Statistical Analysis

The data distribution is uniform in both the training and test datasets, meaning each class is equally represented, as confirmed in Figures 9 and 10. This uniformity ensures there is little variance among classes, which is perfect for efficient training of the model without being biased. Consequently, the model learns more effectively, with balanced updates to its parameters, and delivers reliable and fair predictions across all classes.

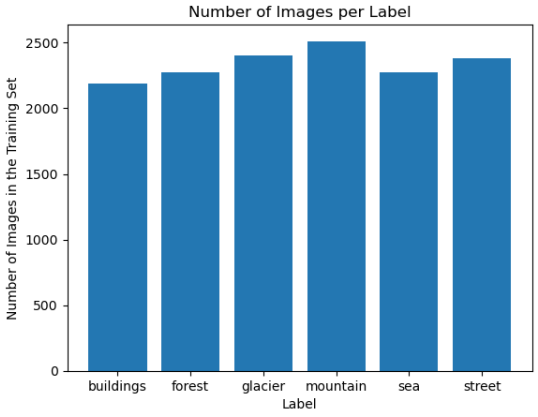

Fig. 9. Distribution of Images per Class in the Training Dataset.

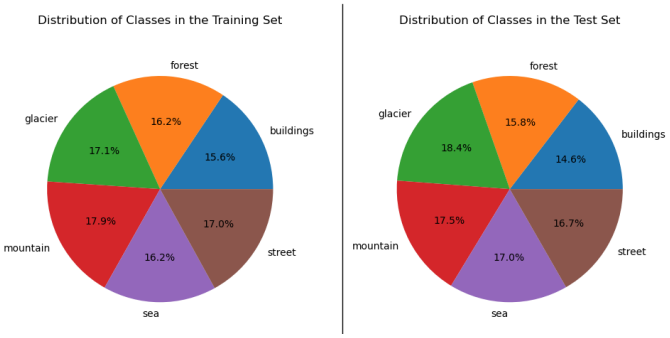

Fig. 10. Distribution of Classes (%) in Training Dataset (left) and Test Dataset (right).

C. Data Preprocessing

For data preprocessing, once the data is uniform across all classes, both in the training and test sets as shown in the

previous section, the only necessary preprocessing step is data normalization.

The images are used with pixel values that range between 0 and 255, and this wide range of values can cause issues during model training. Therefore, each value was divided by 255 to fit inside the range between 0 and 1. In this way, training the model will increase its scalability, leading to improved and more stable performance.

IV. Machine Learning Algorithms

A. CNN

To address the task of scene image classification, we begin with a straightforward approach by implementing a Simple Convolutional Neural Network (CNN). The Simple CNN model serves as our baseline, providing a foundational understanding of how well basic network architectures can perform on the Intel Image Classification dataset.

We developed three versions of the CNN to explore how variations in network complexity and training parameters affect classification performance. Each version differs in its architecture and hyperparameters, allowing us to systematically evaluate the impact of these changes on the model's ability to classify the different scenes accurately.

1) Version 1:

The first version of our Convolutional Neural Network (CNN) was designed with a straightforward architecture aimed at establishing a baseline performance metric. The model consists of two convolutional layers, each followed by a max-pooling layer, and culminates in a fully connected dense layer before the output layer. The specific architecture is as follows:

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_2 (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_3 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_3 (MaxPooling2D)	(None, 36, 36, 64)	0
flatten_1 (Flatten)	(None, 82944)	0
dense_2 (Dense)	(None, 64)	5,308,480
dense_3 (Dense)	(None, 6)	390

Total params: 5,328,262 (20.33 MB)

Trainable params: 5,328,262 (20.33 MB)

Non-trainable params: 0 (0.00 B)

Fig. 11. CNN Version 1 Architecture

The model was compiled using the Adam optimizer with its default learning rate of 0.001. The Adam optimizer is well-suited for this task due to its adaptive learning rate capabilities. The loss function used is "categorical_crossentropy", which is appropriate for multiclass classification problems, and the performance metric tracked during training was accuracy.

The performance of the CNN Version 1 was evaluated using training, validation, and test datasets. The model demonstrated a high accuracy on the training data but showed a significant drop in accuracy on the validation and test data, indicating potential overfitting.

The model obtained 99.74% accuracy with the training data, 77.48% accuracy with the validation data and 76.53% accuracy with the test data. The discrepancy between the training and validation/test accuracies suggests that while the model can learn well from the training data, it struggles to generalize to unseen data. This is further evidenced by the loss values observed during training and validation.

The Figure 12 graphs illustrate the accuracy and loss for both training and validation data over the 10 epochs:

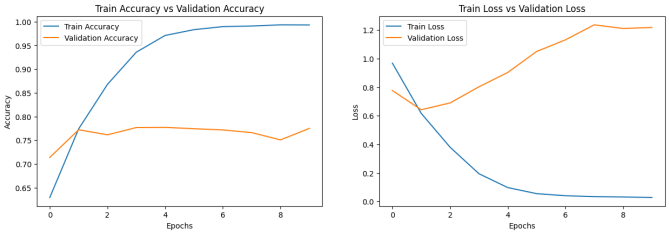

Fig. 12. Accuracy and Loss of the Training and Validation Data for CNN Version 1

These graphs show that while the training accuracy continues to improve and the training loss decreases, the validation accuracy does not improve much since the start of the training and the loss increases, reinforcing the observation of overfitting.

To gain more detailed insights into the model's performance, we generated a classification report and confusion matrix for the validation data. The classification report provides precision, recall, and F1-score for each scene class, while the confusion matrix highlights the specific classes where misclassifications occur most frequently.

	precision	recall	f1-score	support
0	0.69	0.72	0.70	437
1	0.97	0.86	0.91	474
2	0.73	0.75	0.74	553
3	0.70	0.67	0.69	525
4	0.66	0.71	0.68	510
5	0.77	0.78	0.78	501
accuracy			0.75	3000
macro avg	0.76	0.75	0.75	3000
weighted avg	0.75	0.75	0.75	3000

Fig. 13. Classification Report for CNN Version 1

As we can see in Figure 13, the model performs exceptionally well in classifying Forest (1) scenes, with a precision of 0.91, recall of 0.96, and F1-score of 0.93, however, the rest of the scenes do not have such high scores. Glacier (2) and Sea (3) scenes have the lowest precision and recall values, indicating that these categories are more challenging for the model to classify accurately.

Overall accuracy on the validation data is 77%, with macro and weighted averages for precision, recall, and F1-score all at 0.77, suggesting a balanced performance across different classes.

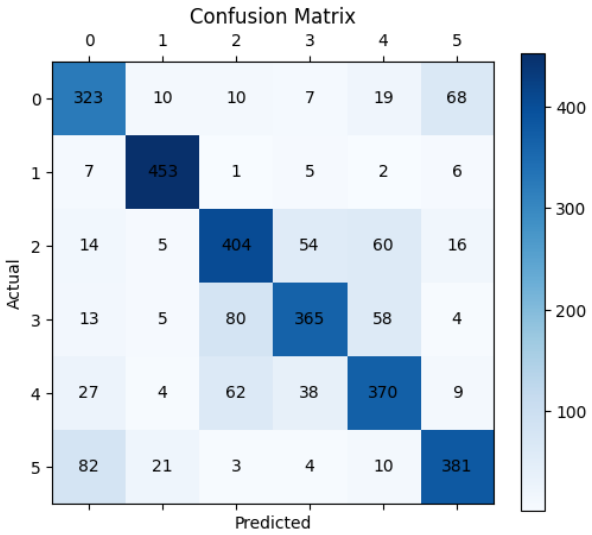

Fig. 14. Confusion Matrix for CNN Version 1

Forests (1) are classified very accurately, with the majority of predictions falling in the correct category. Meanwhile, Buildings (0) are often misclassified as Streets (5), with 68 misclassifications, and Streets (5) are sometimes confused with Buildings (0), showing a notable 82 misclassifications, indicating a challenge in distinguishing these two classes. On the same note, Mountains (3) and Glaciers (2) frequently get confused with each other, reflecting their visual similarities (80 mountains as glaciers and 54 glaciers as mountains).

The ROC (Receiver Operating Characteristic) curves for the six scene classes provide an overall view of the model's ability to distinguish between different types of scenery. Each ROC curve plots the true positive rate against the false positive rate, with the Area Under the Curve (AUC) representing the model's performance.

As it can be seen in Figure 15, only class 1 (forest), had a good AUC Value, while the rest of the scenes came short. This emphasises the results obtained with the classification report (Figure 13) and the confusion matrix (Figure 14).

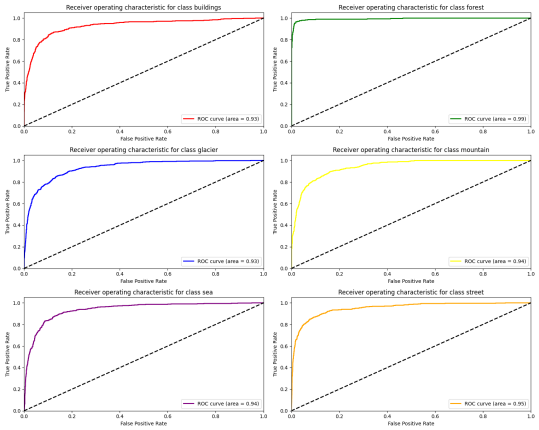

Fig. 15. ROC Curve for CNN Version 1

TABLE I
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR CNN VERSION 2

Train Set	Validation Set	Test Set
99.7%	77.5%	76.5%

2) Version 2:

The second version of our Convolutional Neural Network (CNN) introduced several enhancements aimed at improving the model's performance and addressing the overfitting observed in Version 1. This version featured a more complex architecture, including additional convolutional layers, batch normalization, and dropout, which collectively help in regularization and improving generalization. The architecture of this updated CNN can be seen in Figure 16.

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 150, 150, 128)	9,728
max_pooling2d_7 (MaxPooling2D)	(None, 75, 75, 128)	0
batch_normalization_3 (BatchNormalization)	(None, 75, 75, 128)	512
conv2d_8 (Conv2D)	(None, 75, 75, 64)	73,792
max_pooling2d_8 (MaxPooling2D)	(None, 37, 37, 64)	0
batch_normalization_4 (BatchNormalization)	(None, 37, 37, 64)	256
conv2d_9 (Conv2D)	(None, 37, 37, 32)	18,464
max_pooling2d_9 (MaxPooling2D)	(None, 18, 18, 32)	0
batch_normalization_5 (BatchNormalization)	(None, 18, 18, 32)	128
flatten_3 (Flatten)	(None, 10368)	0
dense_6 (Dense)	(None, 256)	2,654,464
dropout_1 (Dropout)	(None, 256)	0
dense_7 (Dense)	(None, 6)	1,542

Total params: 2,758,886 (10.52 MB)

Trainable params: 2,758,438 (10.52 MB)

Non-trainable params: 448 (1.75 KB)

Fig. 16. CNN Version 2 Architecture

In this version, the model consists of three convolutional layers with increasing filter sizes, each followed by a max-pooling layer and batch normalization to stabilize and accelerate the training process. After the convolutional blocks, the model includes a dense layer with 256 units and a dropout layer with a rate of 0.5 to prevent overfitting, followed by the output layer.

The model was compiled using the Adam optimizer with an initial learning rate of 0.001. A learning rate scheduler, ReduceLROnPlateau, was applied to dynamically adjust the learning rate based on the validation loss, thus enabling finer adjustments as training progresses. The loss function used is "categorical_crossentropy", and the accuracy was the metric tracked during training.

The performance of CNN Version 2 was assessed on training, validation, and test datasets. Significant improvements over Version 1 were observed, particularly in validation and test accuracy, indicating enhanced generalization. The model achieved an accuracy of 99.85% on the training data, 86.14% on the validation data, and 84.77% on the test data. These improvements suggest that enhancements in this version have helped to mitigate the overfitting issue observed in Version 1, although a notable gap between training and validation accuracies remains.

The following graphs (Figure 17) show the accuracy and loss for both training and validation data over 30 epochs:

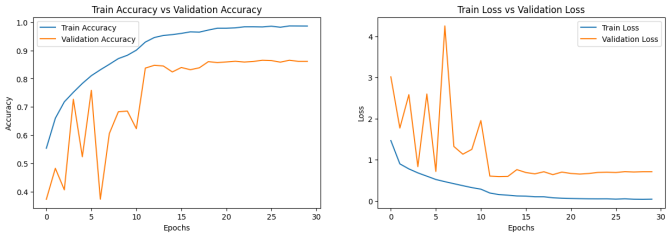

Fig. 17. Accuracy and Loss of the Training and Validation Data for CNN Version 2

Observing these graphs, we notice that the accuracy and loss values for the validation data exhibit significant variation in the initial epochs, only stabilizing around the 15th epoch. After this, we can see that the values are more closely aligned than in Version 1, suggesting a better fit to the validation data.

To further evaluate the model's performance, a classification report and confusion matrix for the test data were generated.

	precision	recall	f1-score	support
0	0.84	0.85	0.85	437
1	0.96	0.97	0.96	474
2	0.82	0.80	0.81	553
3	0.79	0.83	0.81	525
4	0.88	0.87	0.88	510
5	0.89	0.86	0.87	501
accuracy			0.86	3000
macro avg	0.86	0.86	0.86	3000
weighted avg	0.86	0.86	0.86	3000

Fig. 18. Classification Report for CNN Version 2

As shown in Figure 18, the model improved in classifying every scene compared to Version 1. The precision, recall, and F1-scores values increased for all classes, indicating a more balanced and accurate performance. The first version was already good at predicting the Forest (1) scenes, but now the values have increased even more, indicating that the model is very good at identifying forests. The worst scenes are now Glacier (2) and Mountain (3), with Sea (4) receiving very good scores when compared to the first CNN.

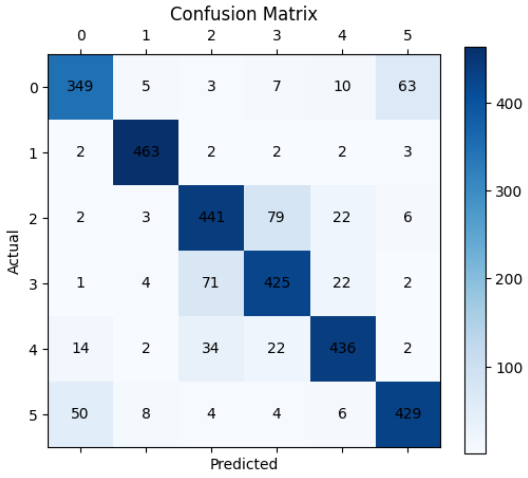

Fig. 19. Confusion Matrix for CNN Version 2

The confusion matrix (Figure 19) reveals fewer misclassifications compared to Version 1. We can see that Forests (1) are classified very accurately, with 463 out of 500 being classified correctly. However, the model still seems to struggle with distinguishing between Buildings (0) and Streets (5). 63 buildings were misclassified as streets, and a higher number, 50 streets were misclassified as buildings. Similarly, there is still confusion between Mountains (3) and Glaciers (2). Although the same misclassifications occur, the number is slightly lower when compared to version 1.

Finally, ROC curves for each scene were plotted. As shown in Figure 20, the AUC value for the Forest scene, which was already high, has further improved. The AUC values for the other classes have also increased, with Buildings showing

the most significant improvement from 0.93 to 0.98, while Glaciers remain the lowest, but now, at 0.96.

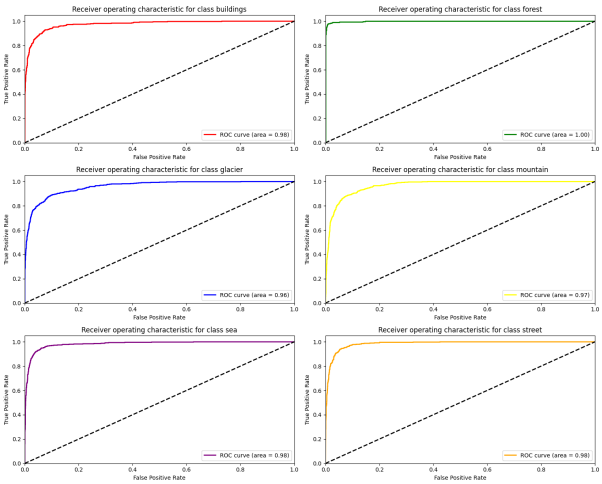

Fig. 20. ROC Curve for CNN Version 2

Overall, the enhancements in CNN Version 2 have resulted in a more robust model with improved generalization to unseen data, as evidenced by higher validation and test accuracies and more balanced performance across different scene classes. However, the discrepancy between the training and validation accuracies remains higher than desired.

TABLE II
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR CNN VERSION 2

Train Set	Validation Set	Test Set
99.8%	86.1%	84.8%

3) Version 3:

The third version of our Convolutional Neural Network (CNN) incorporated additional regularization techniques and data augmentation to further enhance the model's performance and generalization capabilities. This version includes dropout layers within the convolutional blocks, more batch normalization, and data augmentation to increase the robustness of the model against overfitting. The architecture of our final CNN attempt can be seen in Figure 21.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 150, 150, 128)	9,728
batch_normalization_3 (BatchNormalization)	(None, 150, 150, 128)	512
max_pooling2d_3 (MaxPooling2D)	(None, 75, 75, 128)	0
conv2d_4 (Conv2D)	(None, 75, 75, 64)	73,792
dropout_2 (Dropout)	(None, 75, 75, 64)	0
batch_normalization_4 (BatchNormalization)	(None, 75, 75, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 38, 38, 64)	0
conv2d_5 (Conv2D)	(None, 38, 38, 32)	18,464
batch_normalization_5 (BatchNormalization)	(None, 38, 38, 32)	128
max_pooling2d_5 (MaxPooling2D)	(None, 19, 19, 32)	0
flatten_1 (Flatten)	(None, 11552)	0
dense_2 (Dense)	(None, 512)	5,915,136
dropout_3 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 6)	3,078

Total params: 6,021,094 (22.97 MB)

Trainable params: 6,020,646 (22.97 MB)

Non-trainable params: 448 (1.75 KB)

Fig. 21. CNN Version 3 Architecture

In this version, the model comprises three convolutional layers with batch normalization following each layer to normalize the activations and improve stability. Dropout is used after the second convolutional layer to randomly disable a fraction of neurons, preventing co-adaptation and thus reducing overfitting. Max-pooling is applied after each convolutional block to reduce spatial dimensions. A dense layer with 512 units and a dropout rate of 0.3 is added before the output layer.

The model was compiled using the Adam optimizer with a default learning rate of 0.001. A learning rate scheduler, ReduceLROnPlateau, was employed to adjust the learning rate based on the validation accuracy. The loss function used is "categorical_crossentropy", and the accuracy was the primary metric tracked during training.

Data augmentation was applied using the ImageDataGenerator to generate more diverse training samples through random transformations, including rotations, zooms, and shifts. This further enhances the model's ability to generalize to unseen data and thus, minimize overfitting.

The model achieved an accuracy of 92.86% on the training data, 86.32% on the validation data, and 86.27% on the test data. The closer alignment of training and validation accuracies suggests that the regularization techniques and data augmentation effectively mitigated overfitting.

The following graphs (Figure 22) show the accuracy and loss for both training and validation data over 30 epochs:

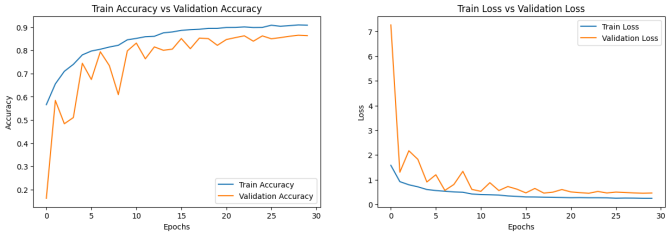

Fig. 22. Accuracy and Loss of the Training and Validation Data for CNN Version 3

Comparing Figure 22 with the previous two versions, we observe a much closer alignment of the validation data to the training data over the epochs. This, along with the improved validation accuracy and decreased validation loss, indicates more effective learning and better generalization.

Observing the classification report (Figure 23) and the confusion matrix (Figure 24), it can be seen that although some improvements in the classification report, the values are still pretty similar. On the other hand, by observing the confusion matrix, the number of misclassifications has decreased once more, especially in the classes that were being misclassified by one another (Buildings (0) with Streets (5), and Glaciers (2), with Mountains (3)).

	precision	recall	f1-score	support
0	0.82	0.87	0.85	437
1	0.92	0.99	0.95	474
2	0.86	0.76	0.81	553
3	0.85	0.73	0.78	525
4	0.76	0.95	0.84	510
5	0.90	0.81	0.86	501
accuracy			0.85	3000
macro avg	0.85	0.85	0.85	3000
weighted avg	0.85	0.85	0.85	3000

Fig. 23. Classification Report for CNN Version 3

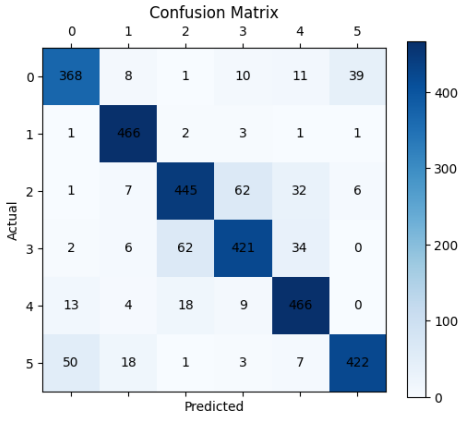

Fig. 24. Confusion Matrix for CNN Version 3

Lastly, to gain further insight, ROC curves for each class were plotted (Figure 25). Similar to Version 2, the AUC values showed minimal improvement. Forests (1) and Streets (5) have very good AUC values, while Glaciers, despite being still the lowest, have improved from 0.96 to 0.97.

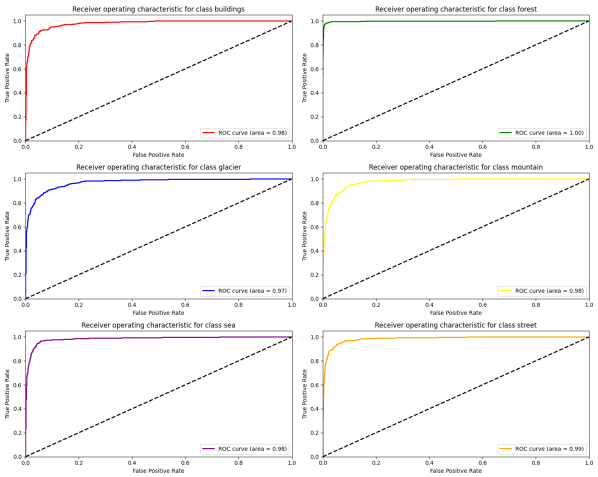

Fig. 25. ROC Curve for CNN Version 3

Due to the incorporation of batch normalization, dropout, and data augmentation in CNN Version 3, the model has become more robust and has mitigated the overfitting seen in the previous models.

TABLE III
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR CNN VERSION 2

Train Set	Validation Set	Test Set
92.9%	86.3%	86.3%

B. AlexNet

AlexNet revolutionized the field of deep learning and computer vision by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a substantial margin. This convolutional neural network (CNN) architecture significantly advanced the performance of neural networks on image classification tasks. AlexNet demonstrated that deep learning models, trained with large datasets and enhanced computational power, could surpass traditional machine learning methods in accuracy and efficiency.

This was the first architecture that used GPU to boost the training performance. AlexNet consists of 5 convolution layers, 3 max-pooling layers, 2 Normalized layers, 2 fully connected layers and 1 SoftMax layer. Each convolution layer consists of a convolution filter and a non-linear activation function called "ReLU" (Figure 26). The pooling layers are used to perform the max-pooling function and the input size is fixed due to the presence of fully connected layers [1].

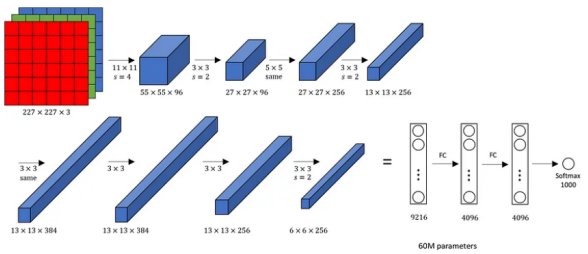

Fig. 26. AlexNet Base Architecture

1) Version 1 - AlexNet:

Firstly, we tried to implement an AlexNet architecture, adapting the original architecture to the intel image classification dataset. To do so we started by defining the architecture of the model (Figure 27).

Layer (type)	Output Shape	Param #
conv0 (Conv2D)	(None, 35, 35, 96)	34,944
bn0 (BatchNormalization)	(None, 35, 35, 96)	384
max0 (MaxPooling2D)	(None, 17, 17, 96)	0
conv1 (Conv2D)	(None, 17, 17, 256)	614,656
bn1 (BatchNormalization)	(None, 17, 17, 256)	1,024
max1 (MaxPooling2D)	(None, 8, 8, 256)	0
conv2 (Conv2D)	(None, 8, 8, 384)	885,120
bn2 (BatchNormalization)	(None, 8, 8, 384)	1,536
conv3 (Conv2D)	(None, 8, 8, 384)	1,327,488
bn3 (BatchNormalization)	(None, 8, 8, 384)	1,536
conv4 (Conv2D)	(None, 8, 8, 256)	884,992
bn4 (BatchNormalization)	(None, 8, 8, 256)	1,024
max2 (MaxPooling2D)	(None, 3, 3, 256)	0
flatten (Flatten)	(None, 2304)	0
fc0 (Dense)	(None, 4096)	9,441,280
fc1 (Dense)	(None, 4096)	16,781,312
fc2 (Dense)	(None, 6)	24,582

Total params: 29,999,878 (114.44 MB)

Trainable params: 29,997,126 (114.43 MB)

Non-trainable params: 2,752 (10.75 KB)

Fig. 27. AlexNet Version 1 Architecture

The architecture was implemented following the Base AlexNet Architecture (Figure 26), only changing the input and output layers to adapt to the dataset in use.

The model was compiled using the Adam optimizer with a default learning rate of 0.001. The loss function used is ”categorical_crossentropy”, and the accuracy was the primary metric tracked during training.

Figure 28 shows the results for the accuracy and loss comparing the performance of both the training set and the validation set obtained during the training of the model.

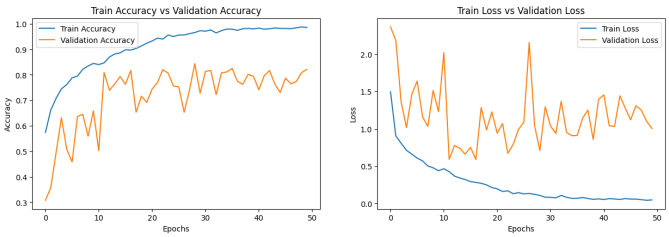

Fig. 28. Accuracy and Loss of the Training and Validation Data for AlexNet Version 1

The graphs (Figure 28) show that the values of the validation data are very inconsistent, not following the training data as they are supposed to. In complement, the model is slightly overfitted, as can be seen by the noticeable gap between the training and validation accuracy values.

After these observations, a classification report (Figure 29) and confusion matrix (Figure 30) were built to display the model's performance across the 6 classes.

Here we can see that this model performed worse than the previous one (CNN Version 3), having the same characteristics in terms of the best and worst performing classes. The best being the Forests (1) and the worst being Buildings (0) misclassified with Streets (5) (and vice-versa) and Glaciers (2) misclassified with Mountains (3) (and vice-versa).

	precision	recall	f1-score	support
0	0.84	0.69	0.76	437
1	0.89	0.97	0.93	474
2	0.72	0.77	0.75	553
3	0.68	0.82	0.75	525
4	0.85	0.73	0.78	510
5	0.86	0.79	0.83	501
accuracy			0.80	3000
macro avg	0.81	0.80	0.80	3000
weighted avg	0.80	0.80	0.80	3000

Fig. 29. Classification Report for AlexNet Version 1

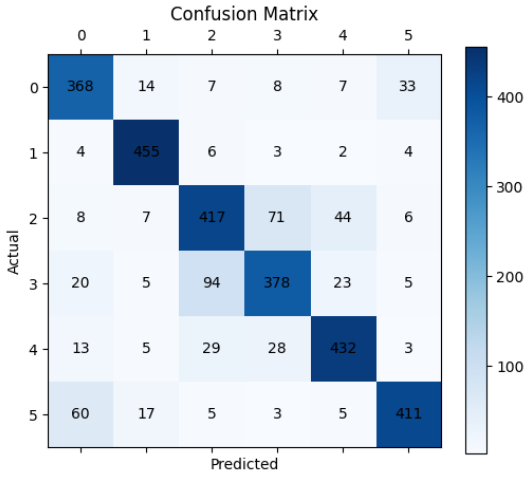

Fig. 30. Confusion Matrix for AlexNet Version 1

To finalize, ROC curves were plotted for each class, to gain some more insight into the model. Once again the only class with a very good AUC value was the Forest with 0.99, and the worst were the Glaciers and Mountains with a value of 0.94.

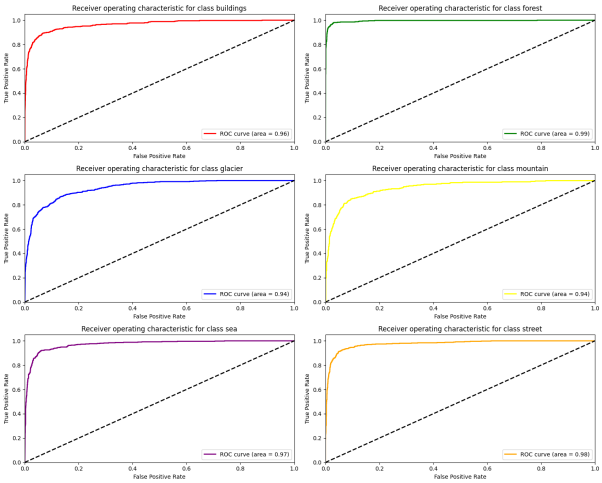

Fig. 31. ROC Curve for AlexNet Version 1

TABLE IV
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR ALEXNET VERSION 1

Train Set	Validation Set	Test Set
97.8%	82.1%	82.0%

2) Version 2 - AlexNet with Fine-tuning:

In a last effort to obtain satisfying results with the AlexNet CNN Architecture, we decided to add two dropout layers, that randomly drop neurons during training, along with applying data augmentation to some of the images in the training set, which randomly rotates the images with a 10-degree range, randomly zooms and shifts horizontally and vertically, and randomly flips the images horizontally. These additions had the intention of constructing a more robust model and also preventing the overfitting detected in the first version of the AlexNet.

The architecture (Figure 32) uses the base AlexNet architecture presented before but now uses the two added dropout layers.

Layer (type)	Output Shape	Param #
conv0 (Conv2D)	(None, 35, 35, 96)	34,944
bn0 (BatchNormalization)	(None, 35, 35, 96)	384
max0 (MaxPooling2D)	(None, 17, 17, 96)	0
conv1 (Conv2D)	(None, 17, 17, 256)	614,656
bn1 (BatchNormalization)	(None, 17, 17, 256)	1,024
conv2 (Conv2D)	(None, 17, 17, 384)	885,120
bn2 (BatchNormalization)	(None, 17, 17, 384)	1,536
conv3 (Conv2D)	(None, 17, 17, 384)	1,327,488
bn3 (BatchNormalization)	(None, 17, 17, 384)	1,536
conv4 (Conv2D)	(None, 17, 17, 256)	884,992
bn4 (BatchNormalization)	(None, 17, 17, 256)	1,024
max1 (MaxPooling2D)	(None, 8, 8, 256)	0
flatten_2 (Flatten)	(None, 16384)	0
fc0 (Dense)	(None, 4096)	67,112,960
dropout_4 (Dropout)	(None, 4096)	0
fc1 (Dense)	(None, 4096)	16,781,312
dropout_5 (Dropout)	(None, 4096)	0
fc2 (Dense)	(None, 6)	24,582

Total params: 87,671,558 (334.44 MB)

Trainable params: 87,668,806 (334.43 MB)

Non-trainable params: 2,752 (10.75 KB)

Fig. 32. AlexNet Version 2 Architecture

The model was compiled using the Adam optimizer with a learning rate of 0.0001. A learning rate scheduler, ReduceLROnPlateau, was employed to adjust the learning rate based on the validation accuracy. The loss function used is ”categorical_crossentropy”, and the accuracy was the primary metric tracked during training. Data augmentation was applied using the ImageDataGenerator and it applies the augmentations referred to above.

Figure 33 compares the accuracy and loss of the training and validation sets over the 30 epochs.

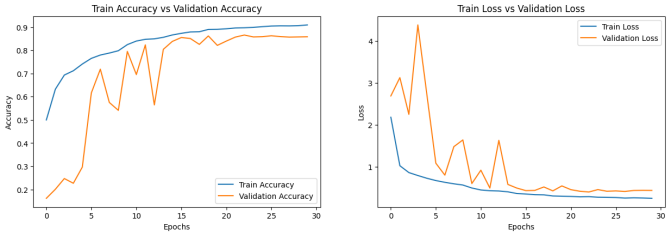

Fig. 33. Accuracy and Loss of the Training and Validation Data for AlexNet Version 2

As observed in these graphs (Figure 33), the model was able to mitigate the overfitting present in the previous AlexNet model, presenting a small gap between the training and validation accuracy and loss values.

To further evaluate the model's performance, we generated a classification report (Figure 34) and a confusion matrix (Figure 35) for the test data.

	precision	recall	f1-score	support
0	0.90	0.81	0.85	437
1	0.91	0.98	0.94	474
2	0.89	0.71	0.79	553
3	0.76	0.88	0.82	525
4	0.84	0.92	0.88	510
5	0.89	0.87	0.88	501
accuracy			0.86	3000
macro avg	0.87	0.86	0.86	3000
weighted avg	0.86	0.86	0.86	3000

Fig. 34. Classification Report for AlexNet Version 2

As shown in Figure 34, the model improved in classifying almost every scene compared to Version 1. The precision, recall, and F1-scores values increased for almost all classes, indicating a more balanced and accurate performance. The best values are as expected, in the Forest (1) class, but now also presents good values in the Sea (4) and Street (5) classes. On the other hand, the recall and F1-score values for Glacier (2) are still low, 0.71 and 0.79 respectively, indicating that the model has problems in correctly classifying this class.

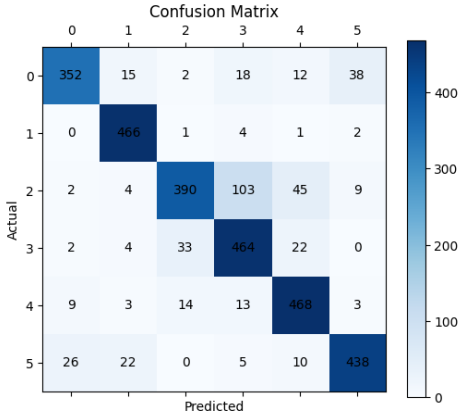

Fig. 35. Confusion Matrix for AlexNet Version 2

The confusion matrix (Figure 35) reveals an improvement in almost every class, with an exception. The Glaciers (2) are being misclassified as Mountains (3) more than any previously developed model, indicating once again that the model is having problems correctly classifying Glaciers.

Finally, ROC curves for each scene were also generated. As shown in Figure 36, the AUC values for the Forest, Sea and Street classes are very high, while the AUC values of the Glacier class are the lowest at 0.97, once again proving that this class is not being well interpreted and classified by our model.

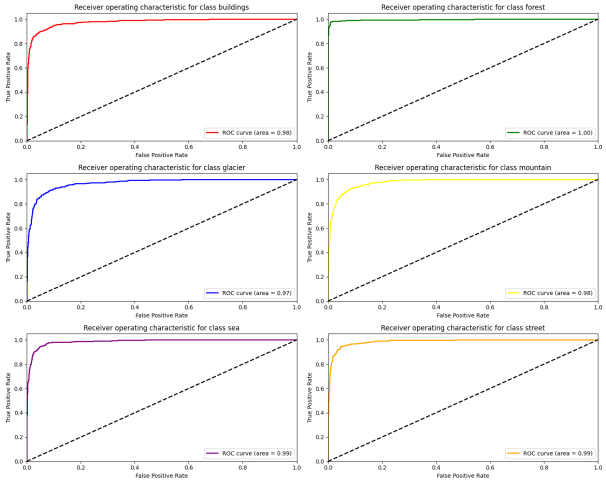

Fig. 36. ROC Curve for CNN Version 2

TABLE V
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR ALEXNET VERSION 2

Train Set	Validation Set	Test Set
92.0%	85.8%	85.9%

C. MobileNet V2

MobileNet v2 is a neural network architecture for computer vision, mainly image classification, on devices with limited resources. It employs an inverted residual block structure together with depthwise convolutions (Figure 37) [9]. These significantly enhance computational efficiency and reduce the number of required calculations while maintaining good accuracy. This makes it an efficient and powerful model for image classification by joining architecture optimized for efficiency with good performance regarding precision.

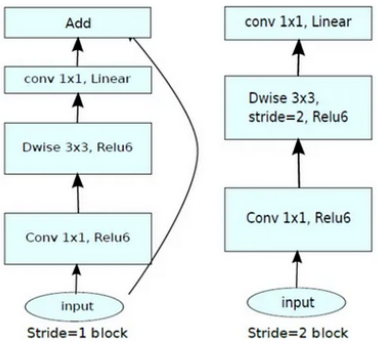

Fig. 37. MobileNetV2 Convolutional Blocks

1) Version 1 - Model Based MobileNetV2 with Frozen Convolutional Layers:

Initial work on MobileNetV2 used a pre-trained model where all convolutional layers were frozen, making only the final layers trainable. This would allow for faster adaptation to this new dataset while keeping most of the features learned from the previous dataset.

Therefore, pre-trained convolutional layers, global pooling, and a dense output layer with softmax activation are used in the model architecture to provide multi-class classification into 6 classes (Figure 38).

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_150 (Functional)	(None, 5, 5, 1280)	2,257,984
global_average_pooling2d_3 (GlobalAveragePooling2D)	(None, 1280)	0
dense_9 (Dense)	(None, 6)	7,686

Total params: 2,265,670 (8.64 MB)

Trainable params: 7,686 (30.02 KB)

Non-trainable params: 2,257,984 (8.61 MB)

Fig. 38. MobileNetV2 Version 1 Architecture

The model was compiled with the RMSprop optimizer using its default hyperparameters and the ”categorical_crossentropy” loss function, which is suitable for multiclass classification problems. It was then trained for 15 epochs using a training set of 11,227 examples, and its performance was monitored with a validation set of 2,807 examples to track how well it performed on unseen data.

Figure 39 shows the results for accuracy and loss comparing the performance on the training set with the validation set obtained during the training of this model.

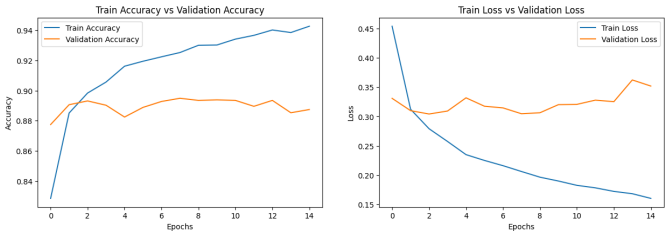

Fig. 39. Accuracy and Loss of the Training and Validation Data for MobileNetV2 Version 1

These graphs (Figure 39) show that the model was slightly overfitted, as there was a visible gap between the training accuracy and loss values in comparison with validation accuracy and loss values.

The classification report (Figure 40) and confusion matrix (Figure 41)were generated using the test data. From these results, it can be clearly seen that model performance for classes 2 and 3 (glacier and mountain) is not as good, returning relatively lower values in terms of precision, recall, and the F1-score compared to what was obtained for other classes.

	precision	recall	f1-score	support
0	0.94	0.86	0.90	437
1	0.98	0.99	0.98	474
2	0.86	0.75	0.80	553
3	0.76	0.85	0.80	525
4	0.88	0.90	0.89	510
5	0.89	0.94	0.91	501
accuracy			0.88	3000
macro avg	0.89	0.88	0.88	3000
weighted avg	0.88	0.88	0.88	3000

Fig. 40. Classification Report for MobileNetV2 Version 1

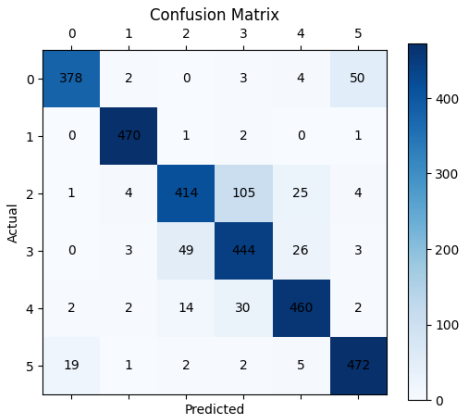

Fig. 41. Confusion Matrix for MobileNetV2 Version 1

To gain further insight into the model, ROC curves were plotted. Since it is a multi-class classification, each class can be taken as the positive class against all other remaining classes. Thereby, a ROC curve for each class can be constructed showing how well the model is at distinguishing that class from others. As it can be seen in Figure 42, classes 0, 1, 4, and 5 had very good AUC values, while 2 and 3 were

poorer, which comes in line with the classification report and confusion matrix results.

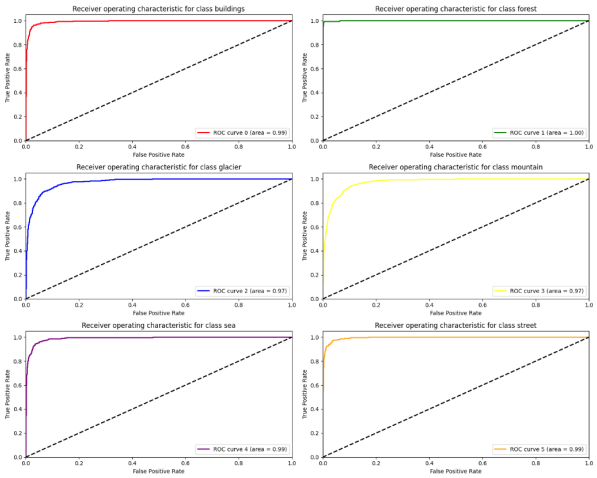

Fig. 42. ROC Curve for each Class for MobileNetV2 Version 1

Finally, the accuracy obtained for the training data, validation data, and test data were compared. Table VI shows the values obtained for each one.

TABLE VI
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR MOBILENETV2 VERSION 1

Train Set	Validation Set	Test Set
94.9%	88.7%	87.9%

2) Version 2 - Trainable MobileNetV2 Model with Additional Layers and Hyperparameter Tunnig:

The results with the initial version of MobileNetV2 turned out not to be the best due to overfitting, so another approach was taken. Hyperparameter tuning was conducted to find the best optimizer, learning rate, momentum, and weight decay for the MobileNetV2 model. Also, retraining of the model took place with the addition of extra layers.

The base MobileNetV2 model was then set to trainable, after which a Global Average Pooling layer was added to reduce the feature map spatial dimensions. Two dense layers were then added with neurons 1024 and 512, both of which significantly improved the learning capacity of the model. In addition, two dropout layers are added to prevent the overfitting process by randomly dropping neurons during training. And, finally, a densely connected output layer with 6 neurons and softmax activation has been added for classification into 6 categories. (Figure 43)

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_150 (Functional)	(None, 5, 5, 1280)	2,257,984
global_average_pooling2d_2 (GlobalAveragePooling2D)	(None, 1280)	0
dense_2 (Dense)	(None, 1024)	1,311,744
dropout (Dropout)	(None, 1024)	0
dense_3 (Dense)	(None, 512)	524,800
dropout_1 (Dropout)	(None, 512)	0
dense_4 (Dense)	(None, 6)	3,078

Total params: 4,097,606 (15.63 MB)

Trainable params: 4,063,494 (15.50 MB)

Non-trainable params: 34,112 (133.25 KB)

Fig. 43. MobileNetV2 Version 2 Architecture

This was followed by Bayesian Optimization-based hyperparameter tuning. Bayesian Optimization is a form of Bayesian inference that requires the construction of a probabilistic model for the objective function. This probabilistic model decides which point to explore next to find the best combination of hyperparameters. Twelve different trials were done, varying the values of the hyperparameters within a specified range, as shown in Table VII.

TABLE VII
HYPERPARAMETER SETTINGS FOR BAYESIAN OPTIMIZATION FOR MOBILENETV2 VERSION 2

Hyperparameter	Values Range
Learning Rate	1e-4 - 1e-2
Optimizer	Adam, RMSprop or SGD
Momentum	0.0 - 0.99
Weight Decay	0 - 0.01

With this hyperparameter tuning, the best hyperparameters obtained are shown in Table VIII.

TABLE VIII
BEST HYPERPARAMETERS OBTAINED FROM HYPERPARAMETER TUNING FOR MOBILENETV2 VERSION 2

Learning Rate	Optimizer	Momentum	Weight Decay
0.00053	SGD	0.63080	0.00899

Then, the model was compiled with the best hyperparameters, and training was conducted using 15 epochs.

Figure 44 shows the results for accuracy and loss comparing the performance on the training set with the validation set obtained during the training of this model.

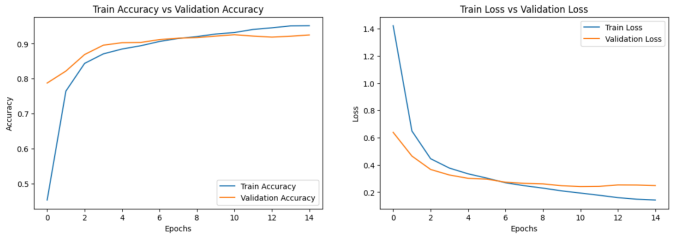

Fig. 44. Accuracy and Loss of the Training and Validation Data for MobileNetV2 Version 2

These graphs (Figure 44) correspond to a well-fitted model for the data, unlike what happened in the first version of MobileNetV2. There is no tendency for this model to overfit or underfit since accuracy and loss values are very similar between the training and validation datasets.

	precision	recall	f1-score	support
0	0.93	0.91	0.92	437
1	0.98	0.99	0.98	474
2	0.86	0.88	0.87	553
3	0.89	0.85	0.87	525
4	0.93	0.95	0.94	510
5	0.92	0.94	0.93	501
accuracy			0.92	3000
macro avg	0.92	0.92	0.92	3000
weighted avg	0.92	0.92	0.92	3000

Fig. 45. Classification Report for MobileNetV2 Version 2

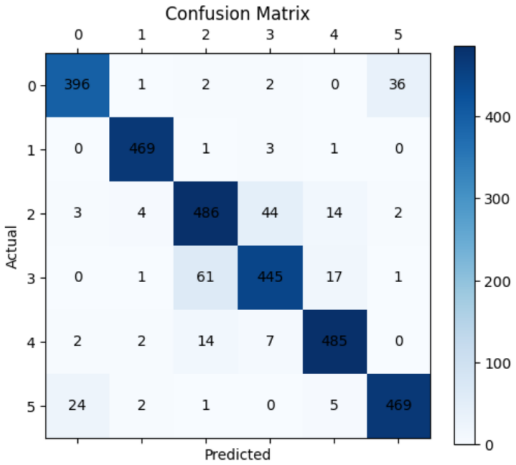

Fig. 46. Confusion Matrix for MobileNetV2 Version 2

The classification report (Figure 45) and confusion matrix (Figure 46) were generated using the test data. From the analysis of these results, it is evident that the model has improved in all classes compared to what was obtained in version one. This model still performed poorer in classes 2 and 3, although the difference from other classes is not as big as in the first version.

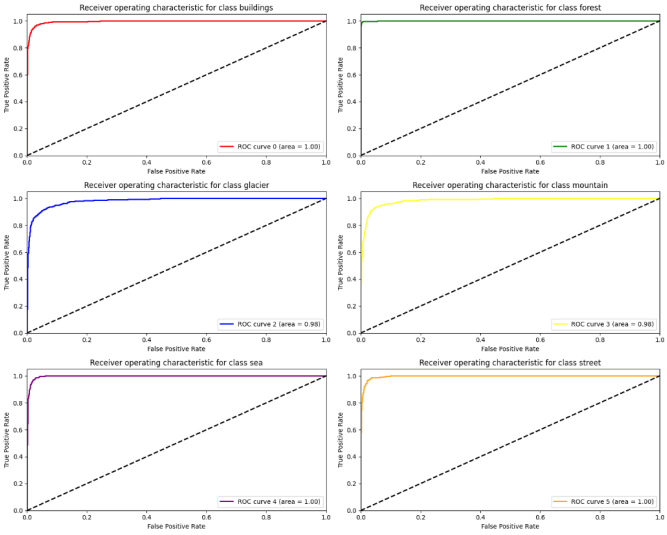

Fig. 47. ROC Curve for each Class for MobileNetV2 Version 1

ROC Curve (Figure 47) analysis was also conducted for this model, which confirms all previous analyses. The AUC obtained for classes 0, 1, 4, and 5 was 1, indicating that the model can perfectly distinguish each of these classes compared to all others. For classes 2 and 3, this value equalled 0.98, meaning not perfect like 1, but still a very reasonable value.

Finally, the accuracy obtained for the training data, validation data, and test data were compared. Table IX shows the values obtained for each one.

TABLE IX
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR MOBILENETV2 VERSION 2

Train Set	Validation Set	Test Set
97.6%	92.5%	91.7%

D. VGG16

VGG-16 is a deep CNN model composed of 16 layers, where there are 13 convolutional layers and 3 fully connected layers. Its simplicity makes it very effective and able to attain very good results in most of the fundamental vision tasks like image classification and object recognition [5]. This model architecture (Figure 48) is characterized by convolutional layers followed by max-pooling layers, which successively increase in depth. Such architecture will let the model learn complex and hierarchical representations of visual features and, hence, lead to accurate and robust predictions.

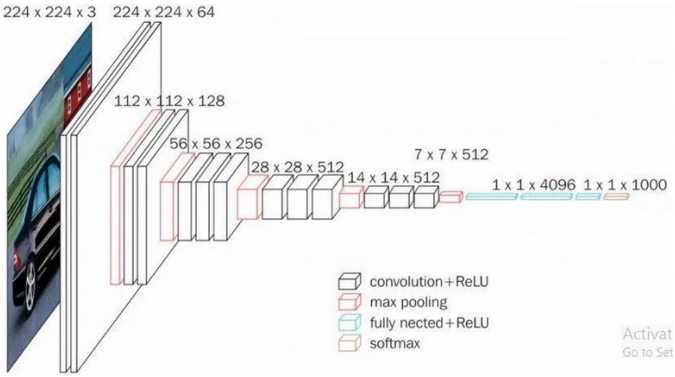

Fig. 48. VGG16 Architecture

1) Version 1 - Transfer Learning with VGG16:

The first technique that was used for the study of the VGG16 model was Transfer Learning, an advanced deep learning technique that utilizes a pre-trained model, VGG16, as a starting point for new learning tasks. This allows the use of features the model has learned on ImageNet and fine-tuning it on much smaller datasets for specific tasks. Here, the layers of VGG16 have been set as non-trainable to preserve the weights it had learnt. Added a Flatten layer on top of VGG16, followed by a dense layer with 'softmax' activation to classify into 6 categories (Figure 49).

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 150, 150, 3)	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1,792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36,928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73,856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147,584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295,168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590,080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590,080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1,180,160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2,359,808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2,359,808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0
flatten (Flatten)	(None, 8192)	0
dense (Dense)	(None, 6)	49,158

Total params: 14,763,846 (56.32 MB)

Trainable params: 49,158 (192.02 KB)

Non-trainable params: 14,714,688 (56.13 MB)

Fig. 49. VGG16 Version 1 Architecture

Next, the model was compiled with categorical cross-entropy as the loss, Adam as the optimizer, and accuracy as the main metric to evaluate the model's performance. This configuration was chosen because it is commonly used in multi-class classification problems, balancing training efficiency with prediction accuracy.

After that, the model was trained for 15 epochs, obtaining the accuracy and loss values on the training and validation sets as shown in Figure 50.

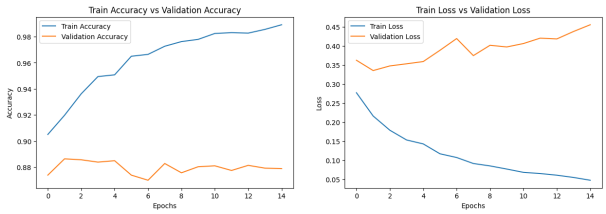

Fig. 50. Accuracy and Loss of the Training and Validation Data for VGG16 Version 1

The graphs of accuracy and loss values of the training data and validation data (Figure 50) prove this model is not ideal. On checking Model Accuracy, it was noticed that there was overfitting because the model accuracy reached 99% on the training data and remained at 87% for the validation data, which is quite a big difference between the training set accuracy and the test accuracy. Also for the loss, it has very different values between the training and validation dataset and for this latter, it tends to increase which isn't good.

	precision	recall	f1-score	support
0	0.92	0.86	0.89	437
1	0.98	0.96	0.97	474
2	0.89	0.71	0.79	553
3	0.75	0.90	0.81	525
4	0.87	0.91	0.89	510
5	0.89	0.93	0.91	501
accuracy			0.88	3000
macro avg	0.88	0.88	0.88	3000
weighted avg	0.88	0.88	0.87	3000

Fig. 51. Classification Report for VGG16 Version 1

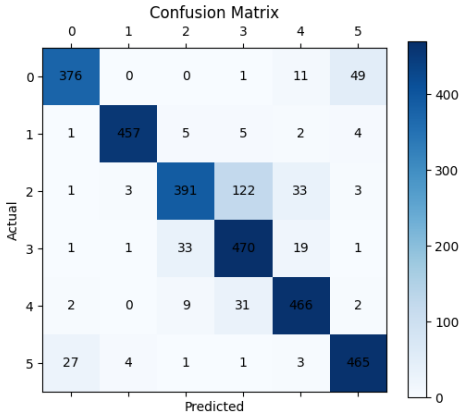

Fig. 52. Confusion Matrix for VGG16 Version 1

The classification report (Figure 51) and confusion matrix (Figure 52) show the model performed fairly well in evaluating the classes with the test data. However, class 3 has low precision compared to other classes. Also, there is a low recall for class 2 and low F1 scores for classes 2 and 3.

The ROC curves (Figure 53) once again demonstrate the accuracy of the results from other performance metrics, showing a low AUC for classes 2 and 3 and a perfect AUC for class 1.

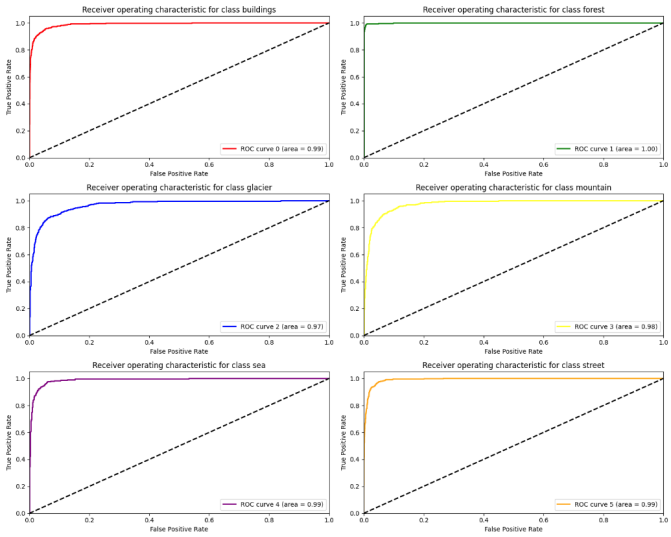

Fig. 53. ROC Curve for each Class for VGG16 Version 1

Finally, the accuracy obtained for the training data, validation data, and test data were compared. Table X shows the values obtained for each one.

TABLE X
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR VGG16 VERSION 1

Train Set	Validation Set	Test Set
98.9%	87.9%	87.5%

2) Version 2 - Fine-tuning VGG16 and Hyperparameter Tuning for Classification:

To combat the overfitting observed in version 1, a fine-tuning approach was adopted. This process involves adjusting a pre-trained model, VGG16, for a specific task by modifying its final layers or adding new layers to tailor the model to the needs of the new task. Moreover, hyperparameter tuning was performed in search of the best values of these hyperparameters.

First, the pre-trained VGG16 model is loaded, and its last layer is removed to adapt it to a new classification task. Then, a new sequential model is created with the output of VGG16 connected to a Flatten layer to prepare the data. Next, it adds a dense layer with 250 neurons and ReLU activation, followed by Dropout to reduce overfitting. Finally, classification into the six classes involved a dense layer with 6 neurons accompanied by a softmax activation function. The strategy followed in this training was that all layers in VGG16 at the start of training were non-trainable except block5_conv1, the first convolutional layer, which would adjust itself to learn special features of the new task. (Figure 54)

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14,714,688
flatten_1 (Flatten)	(None, 8192)	0
dense_2 (Dense)	(None, 250)	2,048,250
dropout_1 (Dropout)	(None, 250)	0
dense_3 (Dense)	(None, 6)	1,506

Total params: 16,764,444 (63.95 MB)

Trainable params: 9,129,180 (34.83 MB)

Non-trainable params: 7,635,264 (29.13 MB)

Fig. 54. VGG16 Version 2 Architecture

After that, Bayesian Optimization-based methods were applied for hyperparameter tuning. Eight trials have been made while varying the values within the intervals defined for every hyperparameter shown in Table XI.

TABLE XI
HYPERPARAMETER SETTINGS FOR BAYESIAN OPTIMIZATION FOR VGG16 VERSION 2

Hyperparameter	Values Range
Learning Rate	1e-4 - 1e-2
Optimizer	Adam, Adagrad or Nadam

With this hyperparameter tuning, the best hyperparameters obtained are shown in Table XII.

TABLE XII
BEST HYPERPARAMETERS OBTAINED FROM HYPERPARAMETER TUNING FOR VGG16 VERSION 2

Learning Rate	Optimizer
0.0001	Adam

The model was then compiled with these values for the hyperparameters, and following that, was trained for 15 epochs.

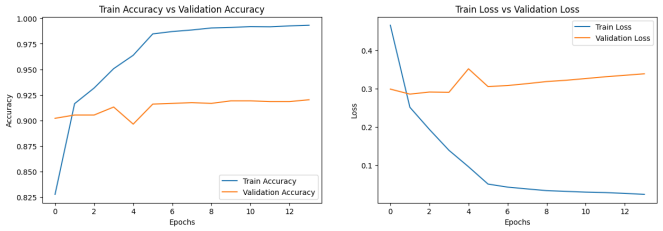

Fig. 55. Accuracy and Loss of the Training and Validation Data for VGG16 Version 2

Comparing the accuracy and loss graphs of the training and validation data (Figure 55) obtained with this model to those obtained in the first version studied, these results show improvements. The training data accuracy has been maintained at 99%, while the validation data accuracy has increased from 87% to 92%. In relation to loss, improvements are also observed, achieving a lower loss on the validation data compared to the first version, however, there is still a tendency for this loss to increase

	precision	recall	f1-score	support
0	0.94	0.92	0.93	437
1	0.98	1.00	0.99	474
2	0.90	0.85	0.87	553
3	0.87	0.89	0.88	525
4	0.94	0.96	0.95	510
5	0.93	0.95	0.94	501
accuracy			0.93	3000
macro avg	0.93	0.93	0.93	3000
weighted avg	0.93	0.93	0.92	3000

Fig. 56. Classification Report for VGG16 Version 2

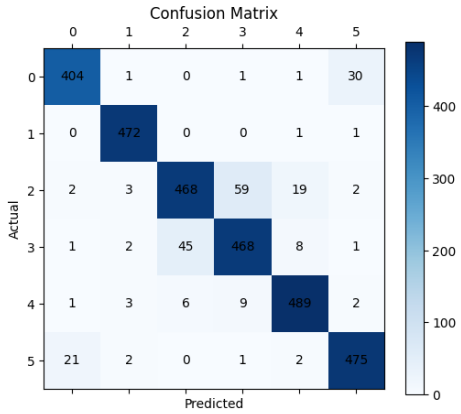

Fig. 57. Confusion Matrix for VGG16 Version 2

Analyzing the classification report (Figure 56) and confusion matrix (Figure 57) obtained from the test data, it is evident that there has been an improvement in the performance of classifying all classes compared to the results obtained in the previous study of the VGG16 model. At this point, none of the parameters have a score below 85%.

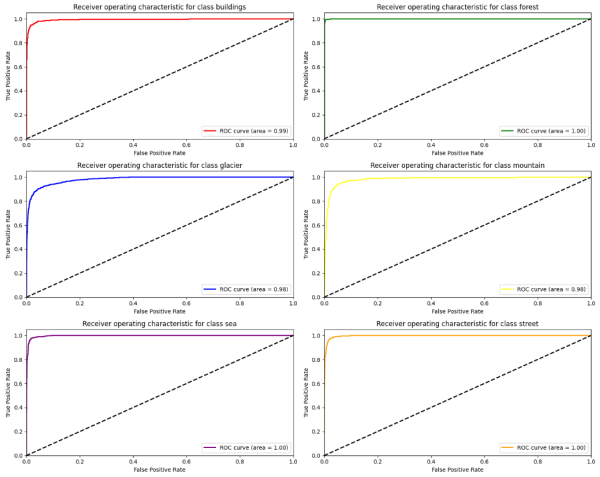

Fig. 58. ROC Curve for each Class for VGG16 Version 2

The ROC curves (Figure 58) show that this model can almost perfectly distinguish each class when compared to all others.

Finally, the accuracy obtained for the training data, validation data, and test data were compared. Table XIII shows the values obtained for each one.

TABLE XIII
COMPARISON TABLE OF ACCURACY ON TRAINING, VALIDATION, AND TEST SET FOR VGG16 VERSION 2

Train Set	Validation Set	Test Set
99.6%	92%	92.5%

V. RESULTS

This section presents the results in terms of the accuracy of the different models previously studied, over the training, validation, and test datasets. All these models are explained in more detail in Section IV: Machine Learning Algorithms.

TABLE XIV
COMPARISON OF MODELS ACCURACY ON TRAINING, VALIDATION, AND TEST DATA

Models	Train Set	Validation Set	Test Set
CNN Version 1	99.7%	77.5%	76.5%
CNN Version 2	99.8%	86.1%	84.8%
CNN Version 3	92.9%	86.3%	86.3%
AlexNet Version 1	97.8%	82.1%	82%
AlexNet Version 2	92%	85.8%	85.9%
MobileNetV2 Version 1	94.9%	88.7%	87.9%
MobileNetV2 Version 2	97.6%	92.5%	91.7%
VGG16 Version 1	98.9%	87.9%	87.5%
VGG16 Version 2	99.6%	92%	92.5%

It can be seen from Table XIV that the accuracy varied across models in the training, validation, and test sets. And

most of the models indicate possible overfitting. For instance, CNN Version 1 and 2 had the highest accuracy on the training set, which is 99.7% and 99.8%, but it drastically drops at validation and the test sets.

Comparing these results with previous state-of-the-art studies, the previous study on MobileNet indicates that it attained an accuracy of 99% for a learning rate of 0.01 and 94% at a learning rate of 0.09 on the training set. Our study on MobileNetV2 Version 2 attained an accuracy of 97.6% on the training set and 92.5% on the validation set. The previous study did not provide validation data results, so a direct comparison is not possible.

One of the previous studies reported achieving an accuracy of 86% on validation data, another reached 92.4%. In our studies, VGG16 Version 1 has reached an accuracy of 88.7%, while Version 2 has reached an accuracy of 92% on the validation data. Results on the loss were identical to previous studies and ours.

Relatively to the AlexNet research presented, it obtained an accuracy of 98.33% on the training data and 87.20% on the test data, it did not use validation data and thus a direct comparison is not possible. The models that we developed were not able to achieve a higher test accuracy than this with 82.1% for Version 1 and 85.8% for Version 2, but the overfitting present in the research was mitigated in the AlexNet Version 2 that uses two Dropout layers and data augmentation.

For CNN, the previous study achieved only 76% accuracy on the validation data, while our three versions of CNN achieved better accuracy than that, with the best being the third version with 86.3% accuracy.

Overall, our studies on these models achieved better results and performance than the state-of-the-art studies reviewed.

To conclude, our top-performing model, MobileNetV2 Version 2, confirms the literature specifying how well this architecture does its job concerning image classification. Moreover, fine-tuning and adjusting hyperparameters resulted in improved accuracy across all models. These findings suggest that while the deeper AlextNet, VGG16 and MobileNetV2 often benefit from fine-tuning, the simpler models, in this case, baseline CNNs, are faced with challenges of overfitting.

VI. Conclusions

A. Discussion

This project provided several valuable insights into various machine learning model performances on the task of image classification. The comparative analysis performed using AlexNet, MobileNetV2, VGG16, and baseline CNNs indicates the dissimilarities in accuracy and robustness on different datasets.

AlexNet was a performance breakthrough in image classification with deep convolutional layers. In this work, version 1 of AlexNet achieved about 97.8% accuracy on the training set. However, it exhibited lower performance on the validation and test sets, suggesting potential issues with generalization. Then, fine-tuning was applied to address these challenges, resulting in a slight improvement in generalization.

The architecture of MobileNetV2 is quite light, efficient, and most appropriate to be run on anything from mobile and embedded devices. For our work, an enhanced version of MobileNetV2 with more layers and the hyperparameters optimized, revealed surprising improvements where the accuracy improved at 97.6% in the training set and 92.5% in the validation set, thus proving robustness across changing computational constraints.

VGG16 is a very deep network and offers excellent feature extraction capabilities, making it the base for most computer vision applications. Our experiments with VGG16 Version 1 and Version 2 did involve finetuning and changes in hyperparameters with notable increases in accuracy.

In contrast to predefined deeper models like VGG16 and MobileNetV2, baseline simple CNNs showed competitive performance in some cases. However, we noticed that baseline CNNs could suffer from overfitting, posting very high training accuracy but low validation and test set accuracy. This behaviour was improved by the introduction of regularization techniques in Version 2 and Version 3, where the last one reduced a substantial amount of the gap between the training and validation accuracy.

B. Challenges

One of the major challenges encountered during this study was the overfitting problems. While models initially show quite competitive results on the training set, their performances turned radically bad on both validation and test sets. This demonstrates that efficient methods for regularization and careful tuning of hyperparameters are essential in order to strengthen generalization capability.

Another challenge we overcame had to do with computational power. Initially, we tried running these models on our local computers, but the models were very computationally intensive, making it hard for our machines to handle them effectively, often taking excessively long training times. To mitigate this issue, we resorted to Kaggle, where we could make use of more powerful computation simply by running a notebook and running the models there. It was this transition that helped us overcome the computational limitations quite considerably and proceed with our experiments effectively.

C. Future Iterations

Future studies can be done in many ways to further refine the results of this work. First, studying more advanced regularization methods may avoid overfitting problems, which frequently occur in baseline CNNs and even in highly complex models like AlexNet. Secondly, ensemble techniques or transfer learning strategies might be able to use effectively the different strengths of the implemented models, including AlexNet, MobileNetV2, and VGG16, and possibly achieve high classification accuracy by combined learning. Lastly, exploring non-CNN models on this dataset could provide insights into alternative approaches to image classification, expanding the scope of model comparison and evaluation.

References

[1] *AlexNet Architecture Explained*. Last accessed 19 June 2024. 2022. URL: https://medium.com/@siddheshb008/alexnet-architecture-explained-b6240c528bd5.

[2] Puneet Bansal. *Intel Image Classification*. Last accessed 8 June 2024. 2019. URL: https://www.kaggle.com/datasets/puneet6060/intel-image-classification.

[3] Waleed Ijaz. *Intel Image classification - VGG16 - Val_acc: 92.4%*. Last accessed 5 June 2024. 2024. URL: https://www.kaggle.com/code/waleedejaz/intel-image-classification-vgg16-val-acc-92-4.

[4] Arshleen Kaur et al. "Fine-tuned EfficientNet and MobileNetV2 Models for Intel Images Classification". In: *2024 3rd International Conference for Innovation in Technology (INOCON)*. 2024, pp. 1–5. DOI: 10.1109/INOCON60754.2024.10512279.

[5] Pawangfg. *VGG-16 — Modelo CNN*. Last accessed 19 June 2024. 2024. URL: https://www.geeksforgeeks.org/vgg-16-cnn-model/.

[6] Mohamed Soudy, Yasmine Afify, and Nagwa Badr. "RepConv: A novel architecture for image scene classification on Intel scenes dataset". In: *International Journal of Intelligent Computing and Information Sciences* (Apr. 2022), pp. 1–11. DOI: 10.21608/ijicis.2022.118834.1163.

[7] Divyanshu Srivastava. *Intel CNN vs VGG16 vs MobileNet*. Last accessed 5 June 2024. 2024. URL: https://www.kaggle.com/code/divyanshuxyz/intel-cnn-vs-vgg16-vs-mobilenet.

[8] Keshav Tangri. *Multi-Class Image Classification using Alexnet Deep Learning Network implemented in Keras API*. Last accessed 20 June 2024. 2020. URL: https://medium.com/analytics-vidhya/multi-class-image-classification-using-alexnet-deep-learning-network-implemented-in-keras-api-c9ae7bc4c05f.

[9] Sik-Ho Tsang. *Review: MobileNetV2 — Light Weight Model (Image Classification)*. Last accessed 19 June 2024. 2019. URL: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c.