Image Classification with Convolutional Neural Networks

Carlos Salguero

September 8, 2024

Abstract

This paper presents an analysis of the Intel Image Classification dataset, a collection of approximately 25,000 natural scene images distributed across six categories. The study explores various Convolutional Neural Network (CNN) architectures for multi-class image classification, including custom CNNs and transfer learning approaches using pre-trained models. The dataset's diverse content of buildings, forests, glaciers, mountains, sea, and street scenes provides a challenging benchmark for evaluating the performance of different CNN-based image classification models.

1 Introduction

Image classification remains a fundamental task in computer vision, with applications spanning various domains such as environmental monitoring, urban planning, and autonomous navigation [1]. The Intel Image Classification dataset offers a rich collection of natural scene images, presenting an opportunity to develop and evaluate machine learning models for multi-class image classification.

This dataset, initially published by Intel for an image classification challenge, contains around 25,000 images of size 150x150 pixels, categorized into six classes: buildings, forests, glaciers, mountains, sea, and street scenes. The diversity of these categories provides a comprehensive test bed for assessing the robustness and generalization capabilities of different classification algorithms. In this study, I aim to:

1. Explore the characteristics and distribution of the Intel Image Classification dataset.
2. Implement and compare various CNN architectures for multi-class image classification.
3. Evaluate the performance of these models using standard metrics and analyze their strengths and weaknesses across different image categories.

2 Data

2.1 Dataset Overview

The Intel Image Classification dataset consists of approximately 25,000 color images, each with dimensions of 150x150 pixels. These images are distributed across six categories:

- Buildings
- Forests
- Glacier
- Mountain
- Sea
- Street

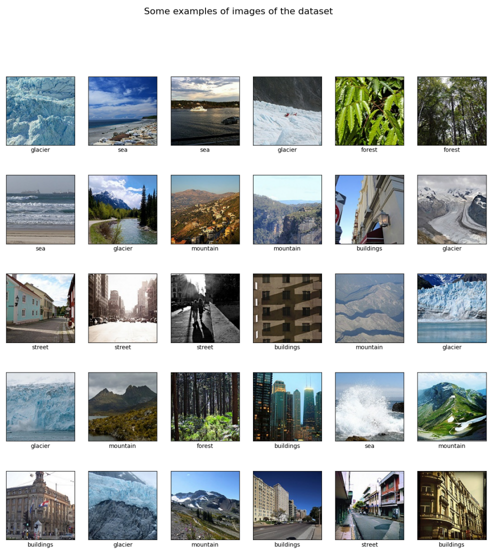


Figure 1: Random images from the dataset

2.2 Data Split

The dataset is divided into three subsets:

- **Training set**: approximately 14,000 images.
- **Testing set**: approximately 3,000 images.
- **Prediction set**: approximately 7,000 images.

This split allows for proper model training, validation, and testing, ensuring a robust evaluation of the classification performance.

2.3 Data preprocessing

Prior to model training, I performed the following preprocessing steps:

1. Image loadng and resizing to ensure consistent dimensiones (150x150 pixels).
2. Normalization of pixel values to the range [0, 1].
3. One-hot encoding of class labels.
4. Data augmentation techniques such as random flips and rotations to increase dataset diversity and prevent overfitting [2].

2.4 Data Characteristics

The dataset presented several challenges and opportunities for image classification:

1. **Intra-class variability**: each category contains a wide range of scenes, lightning conditions, and perspectives.

2. **Inter-class similarity**: some categories, such as mountains and glaciers, may share similar features, potentially leading to classification dificulties.

3. **Balanced classes**: the dataset appears to have a relatively balanced distribution of images across categories, which is beneficial for training unbiased models.

3 Convolutional Neural Networks (CNN)

Convolutional Neural Networks have emerged as the cornerstone of modern image classification systems. These specialized neural networks are designed to automatically and adaptively learn spatial hierarchies of features from input images [3].

3.1 CNN Architecture

The architecture of a CNN is inspired by the organization of the animal visual cortex and is particularly adapted to process grid-like topologies of data, such as images. A typical CNN architecture consists of several key components, as illustrated in Figure 2.

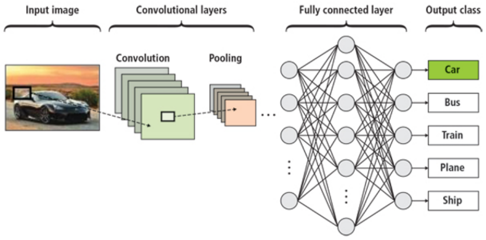


Figure 2: Typical architecture of a Convolutional Neural Network (CNN) for image classification, showing the sequence of convolutional layers, pooling layers, and fully connected layers leading to class predictions [4].

The main components of a CNN architecture include:

- **Convolutional layers**: extract features from the input image.
- **Activation functions**: introduce non-linearity into the model.
- **ooling layers**: reduce spatial dimensions and computational load.
- **Fully connected layers**: compile learned features into class scores.

3.1.1 Convolutional layers

The core building block of CNNs. This layer performs a convolution operation, where a small matrix of numbers (kernel or filter) slides across the input data, performing element-wise multiplication and adding the result.

Mathematically for a 2D input I and a kernel K, it can be expresssed as:

$$(I \times K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \quad (1)$$

Figure 3 illustrates the convolution operation [4]. The center element of the kernel is placed over each source pixel in turn. The source pixel is then replaced with a weighted sum of itself and nearby pixels. This process creates a feature map that indicates the locations and strengths of detected features.

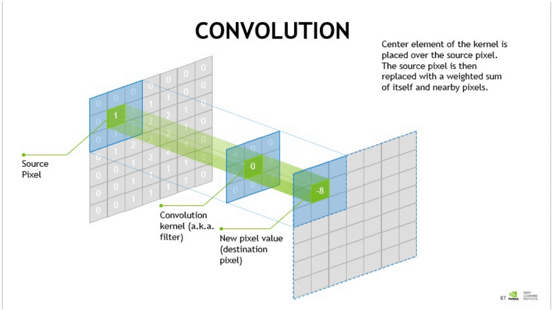

CONVOLUTION
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
Source Pixel
Convolution kernel (a.k.a. filter)
New pixel value (destination pixel)

Figure 3: Illustration of the convolution operation. The convolution kernel (filter) slides over the input image, computing element-wise multiplications and summing to produce the output feature map [4].

3.1.2 Activation Functions

Following each convolutional layer, a non-linear activation function is applied element-wise. The Rectified Linear Unit (ReLU) is a popular choice:

$$f(x) = \max(0, x) \quad (2)$$

ReLU introduces non-linearity into the model, allowing it to learn complex patterns. It also helps mitigate the vanishing gradient problem in deep networks [5].

3.1.3 Pooling Layers

Pooling layers reduce the spatial dimensions of the data, decreasing the computational load and helping to prevent overfitting. Max pooling, which takes the maximum value in each pooling window, is commonly used:

$$y_{ij} = \max_{(a,b) \in R_{ij}} x_{ab} \quad (3)$$

where R_{ij} is a local neighborhood around position (i,j) in the input.

3.1.4 Fully Connected Layers

The final layers of a CNN are typically fully connected layers that compile the features learned by previous layers into class scores. If z is the input to a fully connected layer with weights W and biases b, its output is:

$$y = Wz + b \tag{4}$$

3.2 Implemented Models

3.2.1 Custom Residual Network

The model is a custom residual network inspired by the ResNet architecture [1]. It consists of:

- An initial convolutional layer with 32 filters, followed by batch normalization, LeakyReLU activation, and max pooling
- Three residual blocks, each followed by max pooling:
  - First block with 64 filters
  - Second block with 128 filters
  - Third block with 256 filters
- A flattening operation
- A dense layer with 256 units, followed by batch normalization, LeakyReLU activation, and dropout
- A final dense layer with softmax activation for classification

Each residual block in this model consists of:

- Two convolutional layers with batch normalization and LeakyReLU activation
- A skip connection that adds the input to the output of the second convolution
- An additional convolution on the skip connection if the input and output dimensions don't match

The residual blocks allow for better gradient flow and enable the training of deeper networks. The use of LeakyReLU activations (with a negative slope of 0.1) helps mitigate the ”dying ReLU” problem often encountered in deep networks. L2 regularization is applied to all convolutional and dense layers to prevent overfitting.

The model uses an input shape of (150, 150, 3), suitable for color images, and is designed to classify into 10 different classes. The architecture progressively increases the number of filters in deeper layers while reducing spatial dimensions through max pooling, allowing the network to capture increasingly complex features.

3.2.2 Simplified CNN Model

The second model is a more straightforward CNN architecture, consisting of:

- Three convolutional layers with ReLU activations, each followed by max pooling
- A flattening operation
- Two dense layers, the final one using softmax activation for classification

This simplified model serves as a baseline to compare against the more complex residual network.

3.3 Model Training

Both models are trained using the Adam optimizer [6] with a learning rate of 1e-3, with categorical cross-entropy as the loss function and monitored validation accuracy to prevent overfitting.

4 Implementation

4.1 Custom Residual Network

Defines a convolutional neural network with residual blocks, designed for image classification tasks. The model accepts input images of shape (150, 150, 3). It begins with an initial Conv2D layer of 32 filters, followed by batch normalization and LeakyReLU activation (alpha = 0.1). The network then deepens through a series of residual blocks, each increasing in complexity (64, 128, and 256 filters respectively). Each residual block contains two Conv2D layers with batch normalization and LeakyReLU activations, and employs a skips connection to facilitate gradient flow. The skip connection is adjusted when the input and output dimensions don't match. MaxPooling2D is applied after the initial convolution each and residual block to reduce spatial dimensions. The final stages of the network includes a Flatten operation, followed by a Dense layer with 256 units, BatchNormalization, LeakyReLU activation, and Dropout (rate = 0.5) for regularization. The network concludes with another Flatten operation and a final Dense layer with softmax activation, outputting probabilities for each of the n_classes.

4.1.1 Model Parameters

Parameter type	Number	Size
Total Params	6,523,206	24.88 MB
Trainable Params	6,519,942	24.87 MB
Non-trainable Params	3,264	12.75 KB

Table 1: Parameter summary

4.1.2 Model Summary

Layer Type	Details
Input	Shape: (150, 150, 3)
Conv2D	32 filters, 3x3 kernel, same padding, L2 regularization
BatchNorm + LeakyReLU	$\alpha = 0.1$
MaxPooling2D	2x2 pool size
Residual Block 1	64 filters, 3x3 kernel, L2 regularization
MaxPooling2D	2x2 pool size
Residual Block 2	128 filters, 3x3 kernel, L2 regularization
MaxPooling2D	2x2 pool size
Residual Block 3	256 filters, 3x3 kernel, L2 regularization
MaxPooling2D	2x2 pool size
Flatten	-
Dense	256 units, L2 regularization
BatchNorm + LeakyReLU	$\alpha = 0.1$
Dropout	Rate: 0.5
Flatten	-
Dense (Output)	n_classes units, Softmax activation

Table 2: Model Architecture Summary

4.2 Simplified CNN

A Sequential Convolutional Neural Network (CNN) designed for image classification tasks. It takes input images with a shape of (150, 150, 3), representing RGB images of 150x150 pixels. The architecture consists of three convolutional layers, each followed by ReLU activation. The first Conv2D layer has 32 filters, while the second and third have 64 filters each, all using 3x3 kernels. Max pooling with a 2x2 pool size is applied after the first two convolutional layers, the model flattens the output and passes it through two dense layers. The first dense layer has 64 units with ReLU activation, serving as a fully connected hidden layer. The final dense layer acts as the output layer, with the number of units equal to the number of classes (n_classes) and softmax activation for multi-class classification.

4.2.1 Model Parameters

Parameter type	Number	Size
Total Params	4,791,750	18.28 MB
Trainable Params	4,791,750	18.28 MB
Non-trainable Params	0	0 MB

Table 3: Parameter summary

4.2.2 Model Summary

Layer	Details
Input	Shape: (150, 150, 3)
Conv2D	32 filters, (3, 3) kernel, ReLU activation
MaxPooling2D	(2, 2) pool size
Conv2D	64 filters, (3, 3) kernel, ReLU activation
MaxPooling2D	(2, 2) pool size
Conv2D	64 filters, (3, 3) kernel, ReLU activation
Flatten	-
Dense	64 units, ReLU activation
Dense (Output)	n_classes units, Softmax activation

Table 4: Simplified Model Architecture Summary

4.3 Hyperparameters

The hyperparameters configured for the models are:

- **Epochs**: 300
- **Batch size**: 32
- **Learning rate**: 0.0001
- **Regularization L2**: 0.01
- **Dropout rate**: 0.5

5 Results

5.1 Residual CNN Network

5.1.1 Model Accuracy

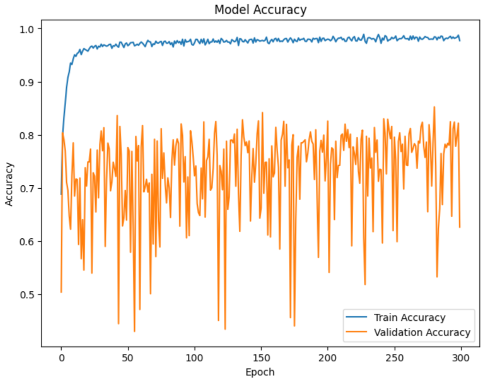


Figure 4: Training and validation accuracy over 300 epochs for the residuals-based CNN model, showing significant overfitting

Figure 4 displays the training and validation accuracy over epochs:

- Training accuracy (blue line) increases rapidly and then steadily, reaching nearly 100% by the end.
- Validation accuracy (orange line) is much more volatile and consistently lower than training accuracy.
- The large gap between training and validation accuracy confirms significant overfitting.
- Validation accuracy shows no clear upward trend after the initial rise, suggesting the model isn't generalizing well to new data.

5.1.2 Model Loss

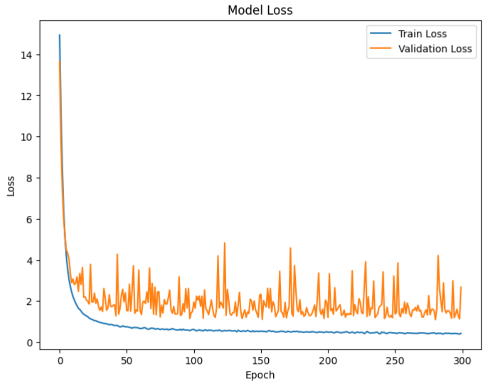


Figure 5: Training and validation loss over 300 epochs for the residuals-based CNN model, indicating a widening gap between training and validation performance

Figure 5 shows the training and validation loss over epochs:

- Both training and validation loss decrease rapidly in the early epochs, indicating quick initial learning.
- The trainling loss (blue line) continues to decrease steadily, reaching very low values.
- The validation loss (orange line) is more volatile and generally higher than the training loss, suggesting some overfitting.
- The gap between training and validation loss widens over time, further indicating overfitting.
- Despite fluctuations, the validation loss shows a general downward trend, suggesting that the model is still learning useful features.

5.1.3 Confusion Matrix

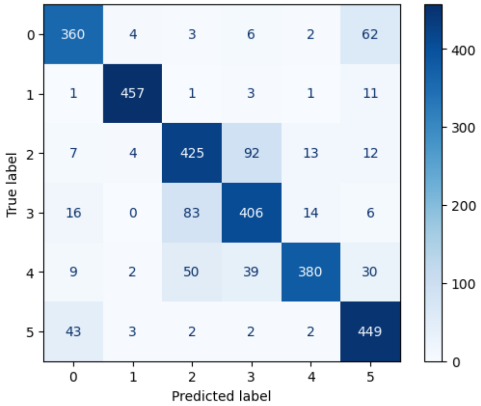


Figure 6: Confusion matrix of the residuals-based CNN model, displaying classification performance across six classes

The diagonal elements represent correct predictions, which are generally high, indicating good overall performance. Class 1 (forests) has the highest correct predictions (457), suggesting that the model performs best on this class. The model seems to perform worst on class 4 (sea), with relatively more misclassifications accross other classes.

There's some confusion between classes, particularly:

- Class 0 (buildings) is sometimes misclassified as class 5 (street) in 43 instances.
- Class 2 (glacier) is sometimes misclassified as class 3 (mountains) in 83 instances.
- Class 3 (mountains) is sometimes misclassified as class 2 (glacier) in 92 instances.

5.1.4 Initial conclusions

1. The model achieves good performance on the training set, but struggles to generalize to the validation set, indicating overfitting.
2. The final test accuracy of 82.57% is reasonably good, but there's room for improvement.

3. The model performs better on some classes than others.

5.2 Fine Tuning Residual CNN Network

5.2.1 Model Accuracy

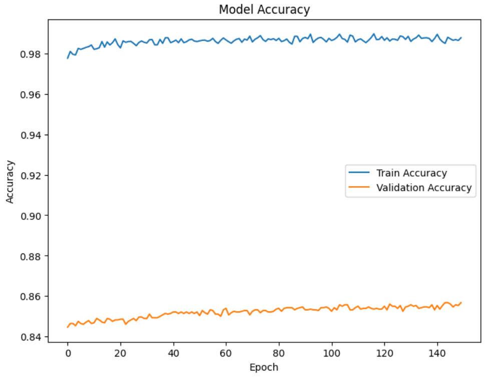


Figure 7: Training and validation accuracy over 150 epochs for the fine-tuned residuals-based CNN model, showing improved generalization and consistent performance improvement

Figure 7 displays the following:

- Training accuracy (blue line) is consistently high (98 - 99%).
- Validacion accuracy (orange line) shows a steady increase over epochs, reaching about 85 - 86%.
- The gap between training and validation accuracy has decreased compared to the pre-fine-tunning results (figure 4), indicating improved generalization.
- The validation accuracy shows a clear upward trend, unlike the pre-fine-tuning graph.

5.2.2 Model Loss

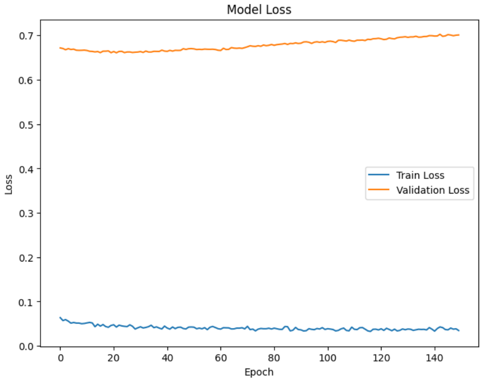


Figure 8: Training and validation loss over 150 epochs for the fine-tuned residuals-based CNN model, demonstrating reduced overfitting and more stable training

Figure 8 shows:

- The training loss (blue line) is consistently lower than the validation loss (orange line), indicating some overfitting still exists.
- Both training and validation losses are much lower and more stable compared to the figure 5.
- The gap between training and validation loss is smaller, suggesting reduced overfitting.
- The validation loss shows a slight upward trend over epochs, which might indicate that the model is starting to overfit towards the end of the training.

5.2.3 Confusion Matrix

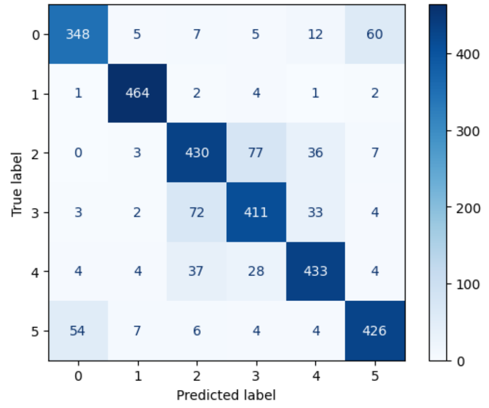


Figure 9: Caption

Figure 9 indicates that the diagonal elements are higher compared to the pre-fine-tuning matrix (figure 6), indicating improved performance. Class 1 (forest) still has the highest correct predictions (464), showing consistent performance. Class 4 (sea) shows improved performance with fewer misclassifications across other classes.

There's reduced confusion between classes, particularly:

- Class 0 (buildings) misclassifications as class 5 (street) increased from 43 to 54 instances.
- Class 2 (glacier) misclassifications as class 3 (mountains) decreased from 83 to 72 instances.
- Clas 3 (mountains) misclassifications as class 2 (glacier) decreased from 92 to 77 instances.

5.2.4 Fine tuning conclusions

- The fine-tuning has led to better overall performance, with test accuracy increasing from 82.57% to 83.73%.
- While some overfitting still exists, the gap between training and validation metrics has decreased significantly, indicating better generalization.
- The confusion matrix shows improved performance across most classes, with reduced misclassifications.
- While there's clear improvement, there's still a gap between training and validation performance, suggesting potential for further optimization.

5.3 Simplified CNN Network

5.3.1 Model Accuracy

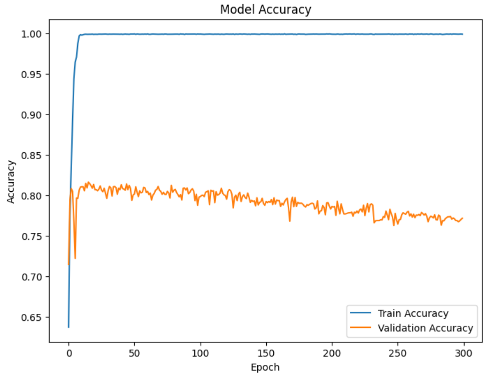


Figure 10: Training and validation accuracy over 300 epochs for the simplified CNN model, showing perfect training accuracy but declining validation accuracy

Figure 10 displays the following:

- Training accuracy (blue line) quickly reaches and mantains nearly 100%, which is a sign of overfitting.
- Validation accuracy (orange line) peaks early (around 81 - 82%) and then shows a gradual decline.
- There's a substantial gap between training and validation accuracy, further confirming overfitting.
- The declining validation accuracy suggests that the model is memorizing the training data rather than learning generalizable features.

5.3.2 Model Loss

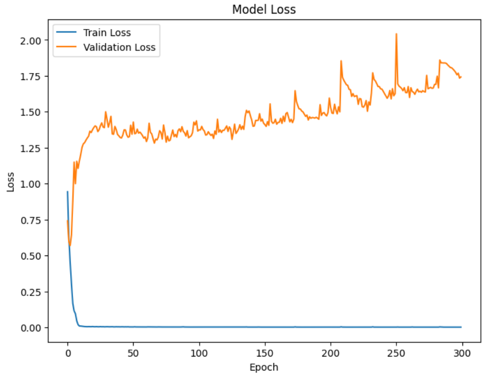


Figure 11: Training and validation loss over 300 epochs for the simplified CNN model, illustrating severe overfitting and unstable validation performance

Figure 11 shows:

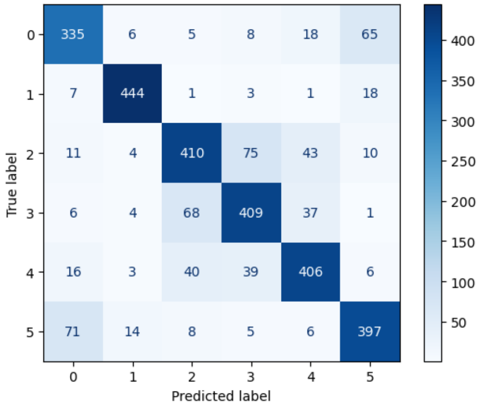
- The training loss (blue line) decreases rapidly and stays very low throughout training.

- The validation loss (orange line) is significantly higher than the training loss and shows an overall increasing trend.

- There's a large and growing gap between training and validation loss, indicating severe overfitting.

- The validation loss becomes more volatile in later epochs, suggesting the model is struggling to generalize.

5.3.3 Confusion Matrix



	0	1	2	3	4	5
0	335	6	5	8	18	65
1	7	444	1	3	1	18
2	11	4	410	75	43	10
3	6	4	68	409	37	1
4	16	3	40	39	406	6
5	71	14	8	5	6	397

Figure 12: Confusion matrix of the simplified CNN model, displaying classification performance and inter-class confusion across six classes

The diagonal elements (correct predictions) are generally high, indicating decent overall performance. Class 1 (forest) has the highest correct predictions (444), showing strong performance for this class. Classes 3 (mountains) and 4 (sea) show relatively good performance with fewer misclassifications.

There's noticeable confusion between some classes:

- Class 0 (buildings) is often misclassified as class 5 (streat) in 71 instances.

- Class 2 (glacier) is often misclassified as class 3 (mountain) in 68 instances and class 4 (sea) in 40 instances.

- Class 5 (street) is sometimes called misclassified as class 0 (buildings) in 65 instances.

5.3.4 Conclusions

- The simplified CNN model shows severe overfitting, as evidenced by the large gap between training and validation metrics in both loss and accuracy graphs.

- The test accuracy of 80.03% is lower than the fine-tuned model from the previous analysis

(83.73%), indicating that the simplified model is less effective.

- The confusion matrix suggests some class imbalance or difficulty in distinguishing between certain classes, particularly 0 (buildings) and 5 (streets), and 2 (glaciers) and 3 (mountains).
- While the training metrics quickly stabilize, the validation metrics show increasing instability, especially in later epochs.
- The model's ability to generalize is poor, as shown by the declining validation accuracy over time.
- The simplified CNN might be too simple to capture the complexity of the data, leading to underfitting on the validation set despite overfitting on the training set.

5.4 Fine Tuning Simplified CNN Network

5.4.1 Model Accuracy

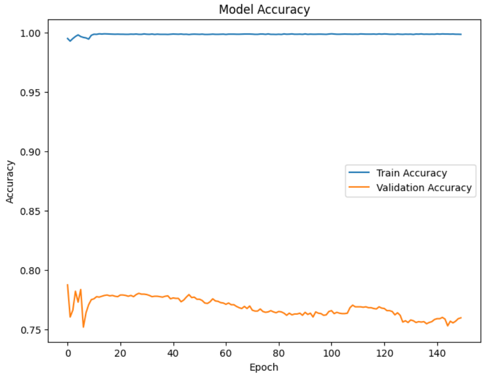


Figure 13: Training and validation accuracy over 150 epochs for the fine-tuned simplified CNN model, showing perfect training accuracy but declining validation accuracy

Figure 13 displays:

- Training accuracy (blue line) quickly reaches and mantains nearly 100%, displaying overfitting.
- Validation accuracy (orange line) shows a gradual decline over epochs, starting around 78% and ending near 75%.
- There's a substantial gap between training and validation accuracy, further confirming overfitting.
- The declining validation accuracy suggests that the model is memorizing the training data rather than learning generalizable features.

5.4.2 Model Loss

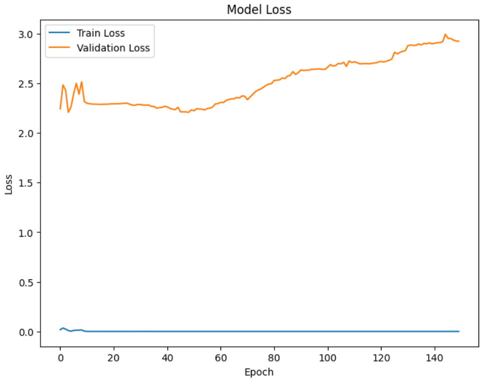


Figure 14: Training and validation loss over 150 epochs for the fine-tuned simplified CNN model, illustrating persistent overfitting and unstable validation performance

Displayed by figure 14

- The training loss (blue line) remains very low and stable throughout the fine-tuning process.
- The validation loss (orange line) is significantly higher than the training loss and shows an increasing trend.
- There's a large and growing gap between training and validation loss, indicating severe overfitting.

- The validation loss becomes more volatile in later epochs, suggesting the model is struggling to generalize.

5.4.3 Confusion Matrix

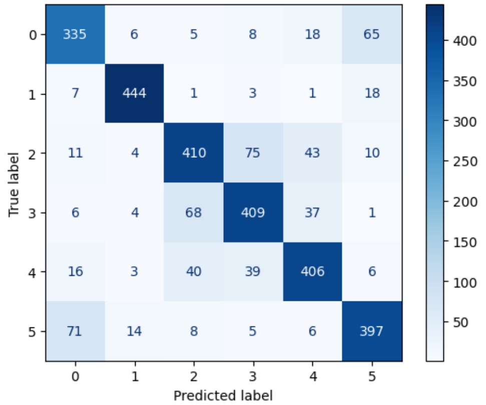


Figure 15: Confusion matrix of the fine-tuned simplified CNN model, showing classification performance and inter-class confusion across six classes

Class 1 (forests) has the highest correct predictions (444), showing strong performance for this class. Classes 3 (mountains) and 4 (sea) show relatively good performance with fewer misclassifications.

- Class 0 (buildings) is often misclassified as class 5 (streat) in 71 instances.
- Class 2 (glacier) is often misclassified as class 3 (mountain) in 68 instances and class 4 (sea) in 40 instances.
- Class 5 (street) is sometimes called misclassified as class 0 (buildings) in 65 instances.

5.5 Fine-tuning Conclusions

- The fine-tuned simplified CNN model still shows severe overfitting, as evidenced by the large gap between training and validation metrics in both loss and accuracy graphs.
- The test accuracy of 80.03% is the same as before fine-tuning, indicating that the fine-tuning process did not improve the model's overall performance.
- The confusion matrix suggests some class imbalance or difficulty in distinguishing between certain classes, particularly 0 (buildings) and 5 (sea), and 2 (glaciers) and 3 (mountains).
- While the training metrics are stable, the validation metrics show increasing instability, especially in later epochs.
- The model's ability to generalize has not improved with fine-tuning, as shown by the declining validation accuracy over time.

6 Conclusion

This study explored the performance of two Convolutional Neural Network (CNN) architectures on the Intel Image Classification dataset, which contains approximately 25,000 natural scene images across six categories. The main findings and conclusions are:

1. **Model comparison**: the custom residual CNN outperformed the simplified CNN, achieving a higher test accuracy with a better generalization and reduced overfitting.
2. **Overfitting**: both models initialy showed signs of overfitting, with high training accuracy but lower validation accuracy. Fine-tuning helped reduce overfitting in the Residual CNN but had minimal impact in the simplified model.
3. **Class performance**: both models performed best on the ”forest” class, consistently achieving high accuracy. There was notable confusion between certain class pairs, such as ”buildings” vs ”street” and ”glacier” vs ”mountains”, suggesting similarities in these categories that challenge the models.
4. **Model complexity**: the superior perforamnce of the Residual CNN suggests that the complexity of the Intel Image Classification dataset requires a more sophisticated model architecture.

The Simplified CNN, while easier to train, may be too simple to capture the nuances of the diverse image categories.

5. **Fine-tunning effects**: fine-tuning significantly improved the Residual CNN's performance, increasing test accuracy and reducing the gap between training and validation metrics. The Simplified CNN did not benefit from fine-tuning, maintaining the same test accuracy and showing persistent overfitting.

In conclusion, while both CNN architectures showed promising results on the Intel Image Classification dataset, the custom Residual CNN demonstrated superior performance and better potential for further improvement.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[4] NVIDIA. What is a convolutional neural network? `https://www.nvidia.com/en-eu/glossary/convolutional-neural-network/`, 2024. Accessed: [Thursday 05, September 2024].

[5] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.