Comparative Study of CNNs and Transfer Learning Using ResNet50 for Image Classification

Student Name : Sri Lekha Thorlapati
Student ID : 24091865
Word Count : 1942
GitHub Link : https://github.com/Thorlapati3108/Comparative-Study-of-CNNs-and-Transfer-Learning-Using-ResNet50-for-Image-Classification.git

Abstract

This tutorial explains how deep learning can be applied to image classification using two approaches: a custom-built Convolutional Neural Network (CNN) and Transfer Learning using a fine-tuned ResNet50 model. The Intel Image Classification dataset is used to demonstrate how convolutional layers learn visual patterns and how pre-trained deep networks outperform models trained from scratch. The tutorial includes training curves, confusion matrices, performance metrics, and real prediction samples. By completing this tutorial, the reader will understand how to construct CNN models, evaluate performance, and apply transfer learning to improve classification accuracy.

Introduction

Image classification is a key task in machine learning and computer vision, where a model assigns a label to an image based on its content. It is widely used in applications such as medical diagnostics, facial recognition, autonomous driving, and remote sensing. Unlike traditional methods that rely on manually designed features, deep learning enables models to learn useful patterns directly from raw image data.

The most common deep learning model used for this task is the Convolutional Neural Network (CNN), which learns features such as edges, textures, and shapes using layered convolution operations. However, CNNs trained from scratch require large datasets and significant computing power, which makes performance weaker when data is limited.

Transfer learning addresses this limitation by reusing models trained on massive datasets such as ImageNet. One widely used transfer learning model is ResNet50, a 50-layer deep network that uses residual connections to improve training stability and accuracy.

In this tutorial, a CNN is first built as a baseline using the Intel Image Classification dataset, followed by a fine-tuned ResNet50 model. The goal is to teach how both approaches work and demonstrate why transfer learning leads to better performance in real-world image classification tasks.

What You Will Learn

By following this tutorial, you will learn how to:

- Build and train a CNN model
- Apply image normalisation and preprocessing
- Detect overfitting using accuracy and loss curves
- Interpret confusion matrices and classification reports
- Use transfer learning to improve performance
- Fine-tune a ResNet50 model safely
- Compare models and explain feature learning differences

Dataset Introduction

The Intel Image Classification dataset contains six balanced outdoor scene categories: **buildings, forest, glacier, mountain, sea, and street**. These categories were selected to represent both natural landscapes and urban environments, allowing the model to learn diverse visual patterns.

The dataset includes over **14,000 training images** and **3,000 validation images**. To ensure consistency between models, all images were resized to **224 × 224 pixels**, which is the required input size for the ResNet50 architecture. Resizing also helps standardise the scale of visual information so that networks learn fairly across classes.

Building a CNN From Scratch

This section introduces how a Convolutional Neural Network (CNN) learns features from images before applying transfer learning techniques.

Understanding CNN Components

A CNN is built from specialised layers that progressively extract visual features from images:

- **Convolution layers** detect low-level and high-level features such as edges, textures, shapes, and object parts.
- **Pooling layers** reduce the spatial size of feature maps, making the model computationally efficient and less sensitive to small shifts.
- **Dense layers** combine all learned features to make final predictions.
- **Dropout** randomly disables neurons during training to prevent overfitting and improve generalisation.

CNN Architecture Overview

The CNN developed in this tutorial consists of three convolution blocks followed by a classification stage. Each convolution block increases the number of filters, enabling the network to learn more detailed patterns at each stage.

The final softmax layer outputs six probabilities, one for each image category.

Although simple in design, this model contains over **11 million parameters**, largely due to the large dense layer. Such a large number of parameters makes the CNN **compute intensive** and more prone to overfitting on small datasets.

Training the CNN

Figure 1: CNN Accuracy Curve

Training accuracy increases steadily, but validation accuracy peaks earlier - indicating overfitting after epoch 3.

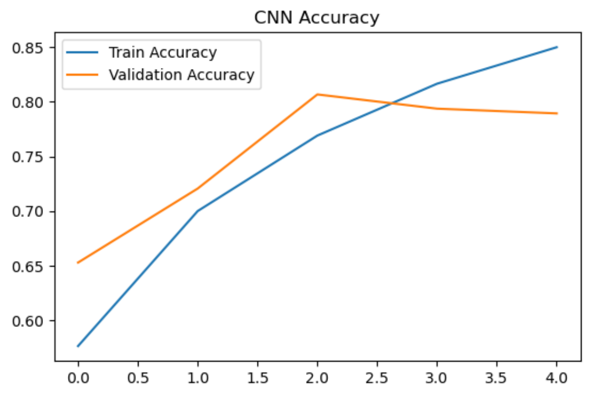
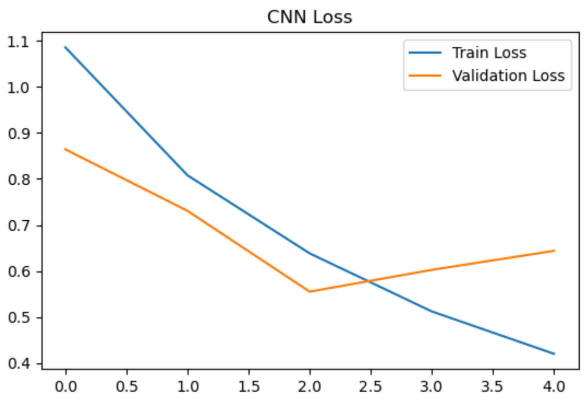


Figure 2: CNN Loss Curve

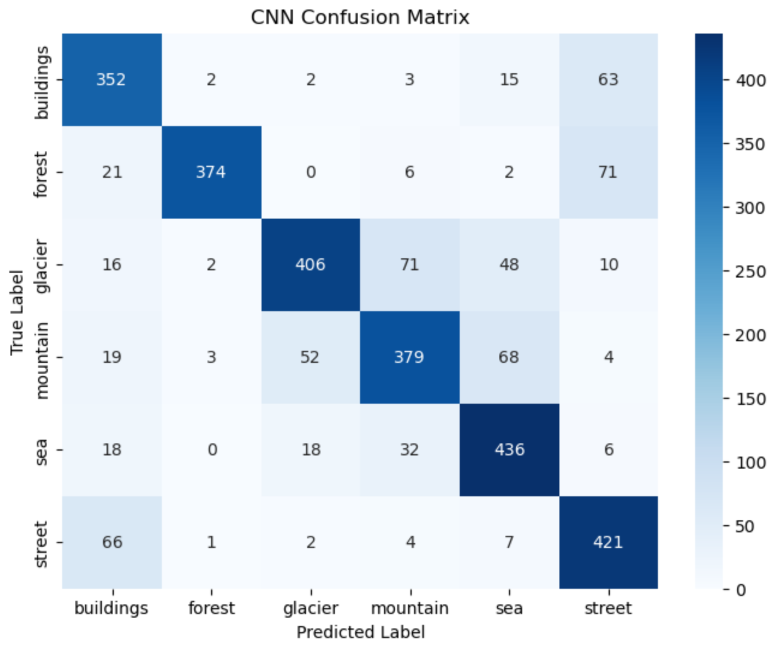


Training loss continues to reduce while validation loss rises, confirming poor generalisation.

These trends demonstrate a classic overfitting pattern. The model becomes highly specialised in recognising training images but is less successful when predicting new unseen data.

CNN Confusion Matrix

Figure 3: CNN Confusion Matrix



True \ Predicted	buildings	forest	glacier	mountain	sea	street
buildings	352	2	2	3	15	63
forest	21	374	0	6	2	71
glacier	16	2	406	71	48	10
mountain	19	3	52	379	68	4
sea	18	0	18	32	436	6
street	66	1	2	4	7	421

The CNN frequently confuses glacier and mountain classes due to visual similarity.

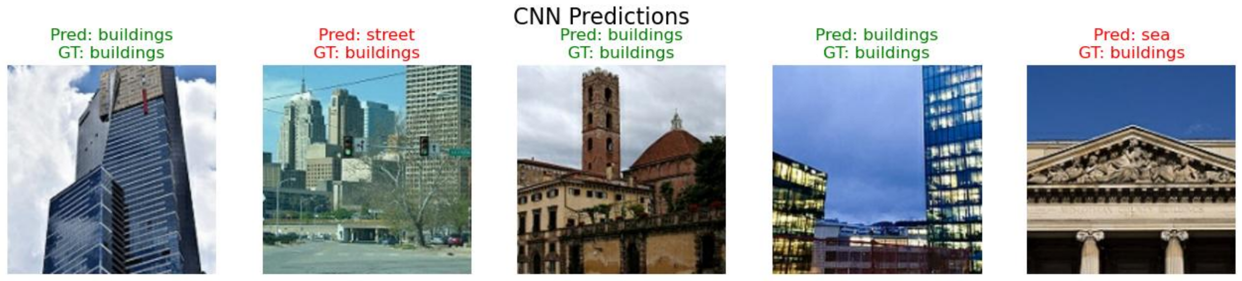
The confusion matrix reveals where predictions fail. From the matrix:

- **Glacier and mountain** images are regularly mixed due to similar textures.
- **Buildings and street** are confused because of overlapping architectural features.
- **Forest** has the highest accuracy due to its distinct colour and texture.

The confusion matrix is a vital diagnostic tool that reveals weaknesses not visible through accuracy alone.

CNN Predictions

Figure 4: CNN Prediction Examples



Several predictions fail due to shared visual characteristics across classes.

Common error patterns:

- Urban structures confuse buildings and streets.
- Lighting and colour overlap cause sea-sky confusion.

These errors highlight that the model struggles when visual clues are subtle or overlapping.

Why Use Transfer Learning?

Training a CNN from scratch can be inefficient when datasets are small or medium-sized. Large models require thousands of images per class and long training time to achieve meaningful accuracy. When trained on small datasets, CNNs tend to overfit because they learn noise instead of useful features.

Transfer learning solves this problem by reusing models trained on large datasets. ResNet50 is a widely used model trained on the ImageNet dataset containing over 14 million images. As a result, the network has already learned powerful feature extractors such as:

- Edges
- Shapes
- Object outlines
- Textures
- Visual patterns

Instead of learning everything from the beginning, transfer learning allows us to adapt this knowledge to a new task.

In this tutorial, only the final classification layers are retrained, while the earlier layers remain unchanged. This significantly improves performance and reduces the risk of overfitting.

Implementing ResNet50

Loading the Model

The ResNet50 architecture is loaded without its original output layer. The original classification layer was designed for 1000 classes, which does not match the Intel dataset. By removing it, the model becomes a generic feature extractor.

A new classification head is added to predict the six classes in this dataset.

Freezing Early Layers

Early layers in ResNet50 detect fundamental visual patterns such as:

- Horizontal and vertical edges
- Brightness gradients
- Corners and shapes

These low-level features apply universally to most images. Therefore, freezing these layers prevents the model from forgetting useful information learned during ImageNet training.

Only the later layers are fine-tuned to adapt the network to the Intel dataset. This technique ensures:

- Faster convergence
- Reduced overfitting
- Higher accuracy

Data Augmentation

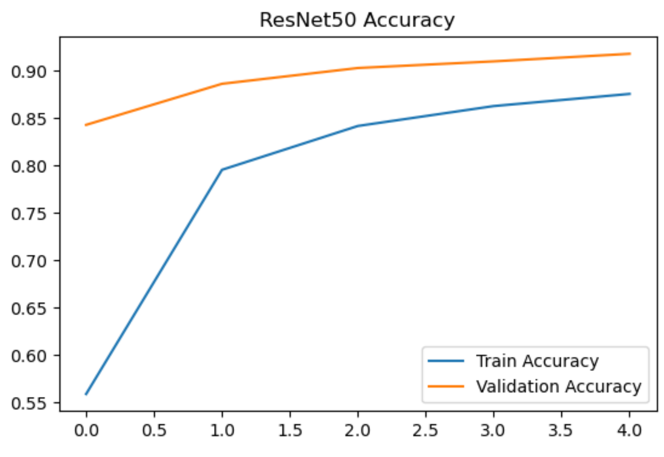
Real-world images vary by orientation, lighting and scale. To simulate these variations, data augmentation is applied:

- Random horizontal flip
- Small rotations
- Random zoom

These transformations help the model recognise objects even when conditions change, improving robustness and generalisation.

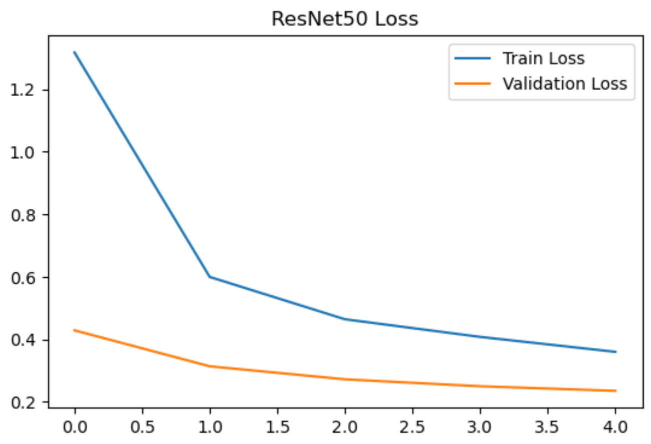
Analysing ResNet50 Performance

Figure 5: ResNet50 Accuracy Curve



Validation accuracy exceeds 91%, showing strong generalisation.

Figure 6: ResNet50 Loss Curve



Both losses decrease smoothly - no signs of overfitting.

Unlike the CNN, ResNet50 shows stable convergence with no divergence between training and validation curves. This confirms that the model generalises effectively.

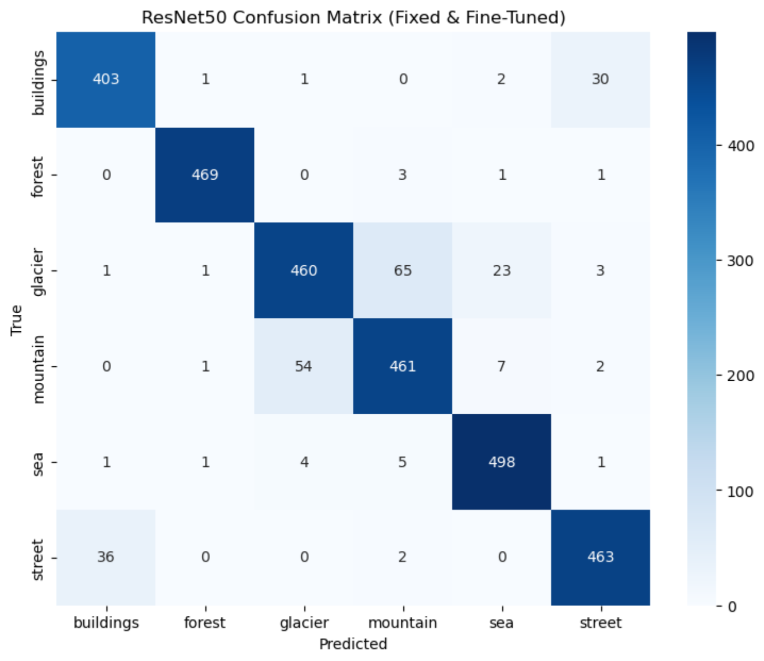
This strong performance occurs due to a combination of:

- Transfer learning
- Batch normalisation
- Lower learning rate
- Image augmentation

Together, these prevent the model from memorising training examples.

ResNet Confusion Matrix

Figure 7: ResNet50 Confusion Matrix



True \ Predicted	buildings	forest	glacier	mountain	sea	street
buildings	403	1	1	0	2	30
forest	0	469	0	3	1	1
glacier	1	1	460	65	23	3
mountain	0	1	54	461	7	2
sea	1	1	4	5	498	1
street	36	0	0	2	0	463

Classification accuracy improves significantly across all classes.

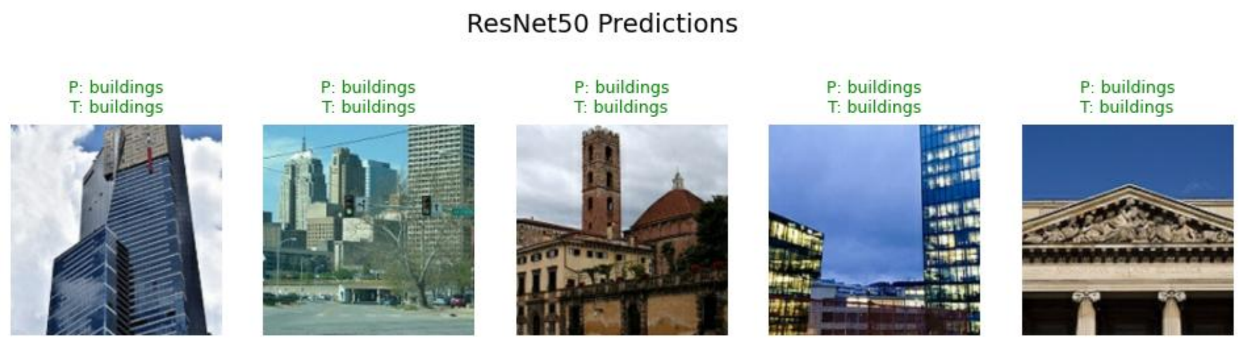
The confusion matrix demonstrates clear improvement over the CNN:

- **Forest and sea** are almost perfectly classified.
- **Glacier and mountain** confusion is rare.
- Urban scenes are distinguished more accurately.

This shows that ResNet50 has learned more meaningful representations of image structure.

ResNet Sample Predictions

Figure 8: ResNet50 Prediction Examples



All predictions shown are correct showing improved classification confidence.

The ResNet model:

- Handles variation better
- Interprets texture accurately
- Understands spatial relationships

Compared with CNN predictions, ResNet50 is more reliable in difficult scenarios such as poor lighting, complex backgrounds or overlapping features.

Model Comparison

Feature	**CNN**	**ResNet50**
Accuracy	79%	91.8%
Training efficiency	Moderate	Faster convergence
Overfitting	Present	None
Generalisation	Weak	Strong
Feature learning	From scratch	Transfer learning

Although the CNN was able to learn useful patterns from the dataset, it struggled to generalise well to unseen images. This is largely due to its reliance on limited training data and a large number of trainable parameters. Errors were common in visually similar classes such as glacier and mountain.

In contrast, ResNet50 demonstrated significantly better performance across all evaluation metrics. Because it was pretrained on the ImageNet dataset, it already possessed a strong understanding of image structure and composition. Fine-tuning only the later layers allowed the model to adapt itself efficiently to the Intel dataset while preserving this prior knowledge.

Learning Checkpoints

1. Why does freezing layers improve generalisation?
Freezing early layers prevents the model from overwriting generic visual features such as edges and shapes. This reduces overfitting and maintains useful representations learned from large datasets.

2. Why does validation accuracy exceed training accuracy in ResNet50?
This occurs when regularisation techniques such as data augmentation and batch normalisation make training examples harder. The model becomes more robust and generalises better to unseen images.

3. What happens if all layers are unfrozen?
Training the entire network may lead to overfitting and unstable learning, especially when the dataset is small. Fine-tuning too many parameters can also destroy useful pretrained knowledge.

Conclusion

This tutorial demonstrated how image classification can be performed effectively using deep learning by comparing a Convolutional Neural Network (CNN) trained from scratch with a transfer learning approach using ResNet50. The CNN model achieved a validation accuracy of **79%**, showing that it was capable of learning useful visual patterns. However, signs of overfitting were observed in the training curves, where validation performance began to stagnate while training accuracy continued to improve. This highlighted one of the primary limitations of training deep networks with limited data.

In contrast, the ResNet50 transfer learning model achieved a substantially higher validation accuracy of **91.8%**. By leveraging pretrained weights learned from millions of images in the ImageNet dataset, the model required less training time and generalised better to unseen data. The improved confusion matrix showed fewer misclassifications, particularly in difficult categories such as glacier and mountain, demonstrating stronger feature learning and improved robustness.

From this comparison, several important conclusions can be drawn:

- Transfer learning significantly improves accuracy and generalisation.
- Pretrained models outperform shallow architectures on small datasets.
- Data augmentation reduces overfitting and increases model robustness.
- Confusion matrices provide critical insights beyond accuracy alone.

For real-world image classification tasks, especially when labelled data is limited, pretrained architectures such as ResNet50 should be prioritised over models trained from scratch. Transfer learning offers a reliable, cost-effective and high-performing solution for building accurate image classification systems with minimal data.

References

- TensorFlow Documentation.
  https://www.tensorflow.org/
- Intel Image Classification Dataset (Kaggle).
  https://www.kaggle.com/datasets/puneet6060/intel-image-classification
- He, K., Zhang, X., Ren, S., & Sun, J. (2016).
  Deep Residual Learning for Image Recognition.
  Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
  https://arxiv.org/abs/1512.03385
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012).
  ImageNet classification with deep convolutional neural networks.
  Advances in Neural Information Processing Systems (NeurIPS).
  https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- Simonyan, K., & Zisserman, A. (2015).
  Very Deep Convolutional Networks for Large-Scale Image Recognition.
  International Conference on Learning Representations (ICLR).
  https://arxiv.org/abs/1409.1556
- Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014).
  How transferable are features in deep neural networks?
  Advances in Neural Information Processing Systems (NeurIPS).
  https://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf
- Pan, S. J., & Yang, Q. (2010).
  A survey on transfer learning.
  IEEE Transactions on Knowledge and Data Engineering.
  https://ieeexplore.ieee.org/document/5288526
- Howard, A. G., et al. (2017).
  MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.
  arXiv preprint.
  https://arxiv.org/abs/1704.04861