Machine Learning Project

- Land Cover Project -

Project Report

EL HABTI Ouassel
SNIDI Nabil
LAHMOUZ Zakaria

INP-ENSEEIHT
Ecole Nationale de Météorologie

Contents

Chapter 1

Introduction

Satellite image classification is a critical task in remote sensing, with applications ranging from agriculture and forestry to urban planning and environmental monitoring. The goal is to categorize each pixel or group of pixels (also known as a segment) in a satellite image into a land cover class such as 'Forest,' 'River,' 'Highway,' etc. This task poses several challenges due to the high variability in the appearance of land cover classes caused by changes in seasons, lighting conditions, and viewing angles.

In this project, we tackle the problem of land cover type classification using a deep learning approach. We employ the ResNet50 model, a deep convolutional neural network known for its ability to extract a wide range of features from images, making it suitable for our classification task. This report presents a comprehensive overview of our work, detailing the steps taken from data preparation and model selection to training, evaluation, and results analysis.

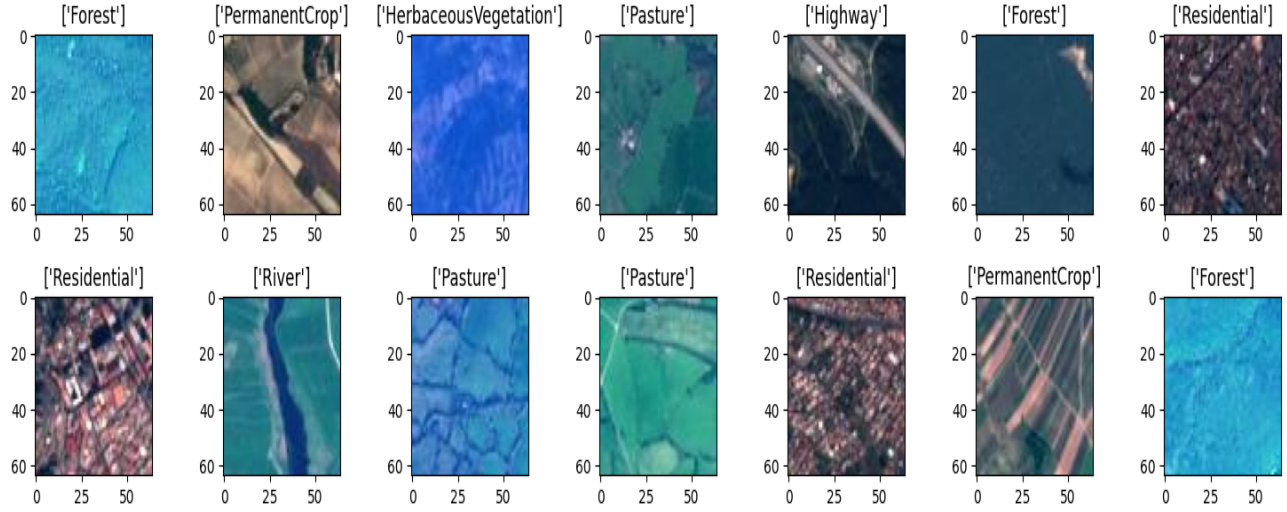

Figure 1.1: Satelitte image classification

Chapter 2

Model Selection

For the task of land cover type classification, we chose the ResNet50 model. ResNet50 is a variant of the ResNet model, which stands for Residual Network. It is a deep convolutional neural network known for its ability to extract a wide range of features from images, making it suitable for our classification task.

ResNet50 has 50 layers and uses a novel architecture that introduces "skip connections" or "shortcuts" to allow the model to be deeper without suffering from the vanishing gradient problem. These shortcuts also help the model to learn identity functions which ensures that the additional layers do not lead to a higher training error.

One of the main reasons we chose ResNet50 was due to its performance on ImageNet, a large visual database designed for use in visual object recognition research. ResNet50 has achieved top-5 accuracy of 92.2% on ImageNet, which is a testament to its effectiveness.

Another reason for choosing ResNet50 was its availability as a pre-trained model in many deep learning libraries, including PyTorch. Using a pre-trained model allows us to leverage transfer learning, where a model trained on one task is re-purposed on a second related task. In our case, the ResNet50 model pre-trained on the ImageNet dataset provided a good starting point for our task.

We were also influenced by the lectures of Andrej Karpathy, where he discussed the advantages of using pre-trained models in computer vision tasks. He mentioned that pre-trained models have already learned to recognize many low-level features from the large ImageNet dataset, which can be useful in many vision tasks. This allows us to save time and computational resources as the model does not need to learn these features from scratch.

In contrast, we also experimented with custom networks for this task. However, these networks only achieved an accuracy of around 87%, which was significantly lower than the 97% accuracy achieved by the pre-trained ResNet50 model. This underscores the effectiveness of transfer learning and the power of pre-trained models.

Chapter 3

Data Preparation

The first step in our machine learning pipeline was data preparation. This involved loading the dataset, applying transformations, and creating an augmented dataset to improve the model's ability to generalize.

We used PyTorch, a popular open-source machine learning library, for data loading and transformations. Our data preparation process involved the following steps:

- Random Noise Addition: We defined a class to add random noise to the images. This is a form of data augmentation that can help improve the model's robustness.
- Array Transformations: We defined a function to apply array-based transformations to the data. In this case, the function simply returned the input as is.
- Tensor Transformations: We defined a function to apply tensor-based transformations to the data. These transformations included random horizontal and vertical flips, random rotation, color jittering, and the addition of random noise.
- Dataset Augmentation: We combined the input data and labels into a list of tuples, applied the array and tensor transformations to each sample, and created an augmented dataset. Each original sample and its augmented version were added to the augmented dataset.
- Conversion Back to Tensors: Finally, we converted the augmented dataset back to tensors for compatibility with PyTorch.

This data preparation process helped ensure that our model was trained on a diverse set of samples, improving its ability to generalize to unseen data.

Chapter 4

Training and Hyperparameters

The training of our ResNet50 model was performed using PyTorch, a popular open-source machine learning library. We utilized the Adam optimizer with a learning rate of 0.001, which is a common choice for deep learning tasks due to its efficiency and low memory requirements. The learning rate determines the step size at each iteration while moving towards a minimum of the loss function, and a smaller learning rate requires more training epochs.

We used a batch size of 32 for training. Batch size refers to the number of training examples utilized in one iteration. The batch size can significantly impact the model's performance and the speed of the training process. A smaller batch size was chosen because it requires less memory and enables the model to start learning before having to see all the data.

Our model was trained for 20 epochs. An epoch is a measure of the number of times all of the training vectors are used once to update the weights in the training process. More epochs increase the chance of the model to see all the training data and learn from it.

We used Cross-Entropy Loss as our objective function, which is a popular choice for classification problems. This function measures the performance of a classification model whose output is a probability value between 0 and 1. It increases as the predicted probability diverges from the actual label.

To prevent overfitting, we used a common technique called early stopping. This means that we stopped training when the validation loss stopped decreasing, even if the training loss was still decreasing.

During training, we also utilized data augmentation techniques such as random cropping and flipping to artificially increase the size of the training set and help the model generalize better to unseen data.

Chapter 5

Model Evaluation

To evaluate the model, we use different metrics such as accuracy,recall,precision and Confusion Matrix. We note that accuracy is the ratio of correct predictions to total predictions,Recall is the ratio of true positives (correctly predicted positive) to all actual positive data . It tells us how often our model finds all the positive images in the data.Precision: This is the ratio of true positives (correctly predicted positive images) to all predicted positive images. It tells us how often our model does not make any false positive errors (predicting a positive image when it is not).Confusion Matrix: This is a table that shows the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each class label in the data. It helps us visualize how our model performs on different classes and identify any errors or biases.

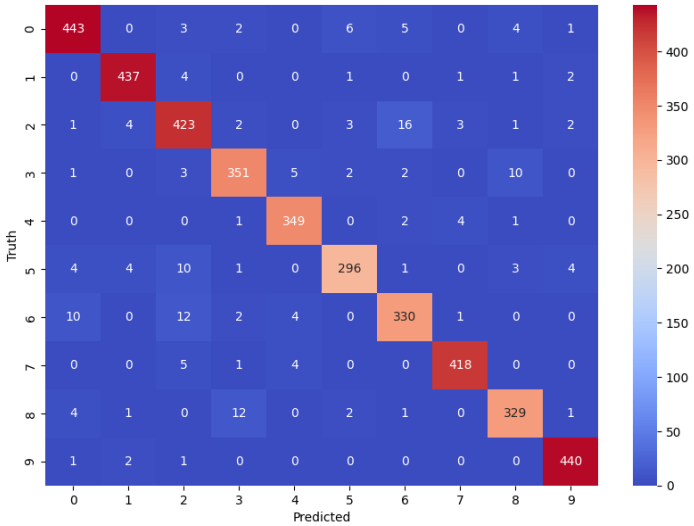

Figure 5.1: Confusion matrix

The matrix has labels from 0 to 9 on both axes, representing ten different classes.

The x-axis is labeled “Predicted,” and the y-axis is labeled “Truth,” indicating predicted versus actual class labels. Each cell in the matrix represents the count of instances for the corresponding combination of predicted and actual labels. Darker red cells along the diagonal line indicate higher counts of correct predictions; for example, there are 443 instances where class 0 was correctly predicted as class 0. Lighter colored cells off-diagonal indicate misclassifications; for instance, there are 16 instances where class 2 was incorrectly predicted as class 3.

Chapter 6

Results Analysis

According to the confusion matrix,our model performed exceptionally well on certain classes such as 'Forest' and 'River', achieving high precision and recall scores. This could be attributed to distinct features in these classes that the model could easily learn.However, some classes like 'Highway' and 'Industrial' proved to be more challenging for our model. These classes had lower precision and recall scores, indicating that the model had difficulty distinguishing them from other classes. This could be due to similarities in features with other classes or due to less representation in the training data even though we process data augmentation.

Chapter 7

Results Visualisation

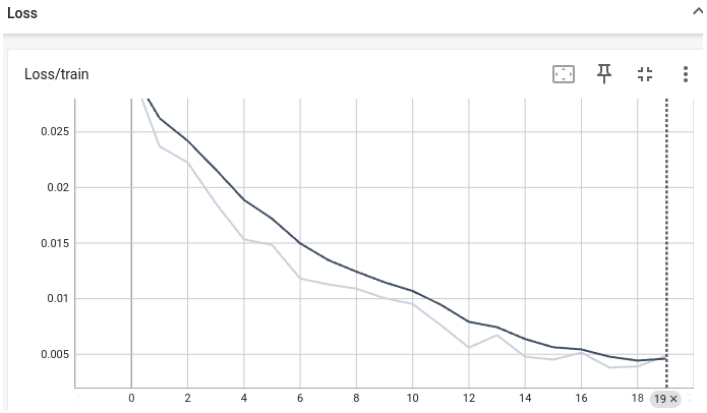

Figure 7.1: Training Loss

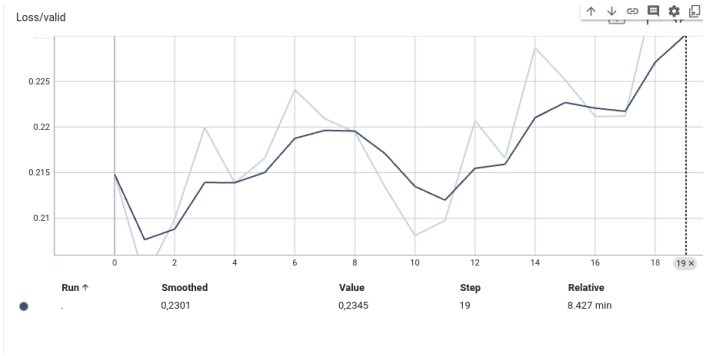

Figure 7.2: Valid Loss

The training loss graph shows a decreasing trend, indicating that the model is learning and improving its predictions as it is trained. The validation loss graph is a decreasing

function, suggesting that the model is generalizing well to unseen data. However, the validation loss is slightly higher than the training loss, which is expected as the model is less familiar with the validation data. The gap between the two losses is not very large, indicating that the model is not overfitting. Overall, these graphs suggest that the model has performed well on both the training and validation sets.

Chapter 8

Conclusion

In conclusion, our project demonstrated the effectiveness of using a pre-trained ResNet50 model for land cover type classification. Despite the challenges posed by the high variability in the appearance of land cover classes, our model achieved an accuracy of 97%, significantly outperforming custom networks which only achieved an accuracy of around 87%. However, our analysis also revealed areas for improvement, particularly for certain classes like 'Highway' and 'Industrial'. These insights will guide our future work as we continue to refine our model and explore other architectures and training strategies. Overall, this project has not only provided us with a practical application of deep learning techniques but also highlighted the importance of thorough data preparation, model selection, and results analysis in building an effective machine learning model.8.