# Satellite Image Classification with CNN, Grad-CAM, and Multimodal Explanations

Course: INFO-6147 Deep Learning With PyTorch

Priyanshi Patel

**Abstract**

This capstone project explores the application of deep learning in satellite image classification. A convolutional neural network (CNN) based on ResNet18 architecture is trained on the EuroSAT dataset. To enhance interpretability, Grad-CAM is used to generate visual explanations of the model's decisions, and BLIP, a multimodal large language model (LLM), is integrated to generate natural language descriptions of image content. The study follows all stages of a deep learning pipeline, from data preprocessing to model evaluation and reporting.

# Introduction

This project aims to classify satellite images into 10 land-use categories using a Convolutional Neural Network (CNN) based on ResNet18 architecture. To make the model decisions interpretable, Grad-CAM visualizations were employed to highlight important regions influencing the classification. Additionally, Salesforce's BLIP multimodal Large Language Model (LLM) was incorporated to generate natural language explanations of the images, providing complementary insights.

# Dataset Selection

The satellite image dataset used for this project was downloaded from a GitHub. The dataset consists of RGB images organized into folders by class label. There are 10 classes representing various land use and land cover types: *AnnualCrop, Forest, HerbaceousVegetation, Highway, Industrial, Pasture, PermanentCrop, Residential, River, and SeaLake.* Each class contains several hundred images, resulting in a total of over 27,000 labeled samples.

# Data Preprocessing

To prepare the dataset for training, all images were resized to 64×64 pixels to reduce computational cost while preserving visual features. The preprocessing pipeline for the training set included:

- Random horizontal flipping to make the model invariant to orientation.
- Random rotation up to 15 degrees to simulate different satellite angles.
- Normalization using ImageNet mean and standard deviation: [0.485, 0.456, 0.406] for the mean and [0.229, 0.224, 0.225] for the standard deviation.

For validation and testing, only resizing and normalization were applied to ensure unbiased evaluation. The dataset was split into three subsets using an 80:10:10 ratio:

- 80% for training
- 10% for validation
- 10% for testing

A batch size of 64 was used for loading data during model training and evaluation. The preprocessed images and labels were visualized using `matplotlib` to ensure correct loading and class distribution.

# Model Selection and Architecture

A convolutional neural network using the ResNet18 architecture was selected. The model was customized to output predictions for 10 classes. Key components include:

- Convolutional layers with ReLU activation

- Residual blocks for deep feature extraction
- Dropout for regularization
- Final softmax classification layer

# Model Training

The convolutional neural network used for training was the ResNet-18 architecture from PyTorch's `torchvision.models`, with randomly initialized weights (`weights=None`). The final fully connected (FC) layer was replaced with an output layer corresponding to the number of target classes in the dataset.

The model was trained using the Adam optimizer and the CrossEntropyLoss function. The training was performed for a maximum of 5 epochs. To prevent overfitting, early stopping was implemented with a patience of 1 epoch—training was halted if the validation accuracy did not improve.

During each epoch, both the training loss and training accuracy were recorded. After each epoch, the model was evaluated on the validation set to compute validation accuracy. The model achieving the highest validation accuracy was saved to disk.

Training and validation accuracy, along with training loss, were plotted after each experiment to visualize learning dynamics and convergence.

# Hyperparameter Tuning

A small grid search was performed to identify the optimal learning rate and batch size. The model was trained under four different configurations using combinations of:

- Learning rates: 0.001, 0.0005
- Batch sizes: 32, 64

Each configuration was trained using the same architecture, training loop, and early stopping condition. Based on the validation accuracy results across multiple runs, the best performing hyperparameters were:

- **Learning Rate:** 0.001
- **Batch Size:** 32

This combination achieved the highest validation accuracy of **85.19%**, as shown in the training and validation accuracy plots.

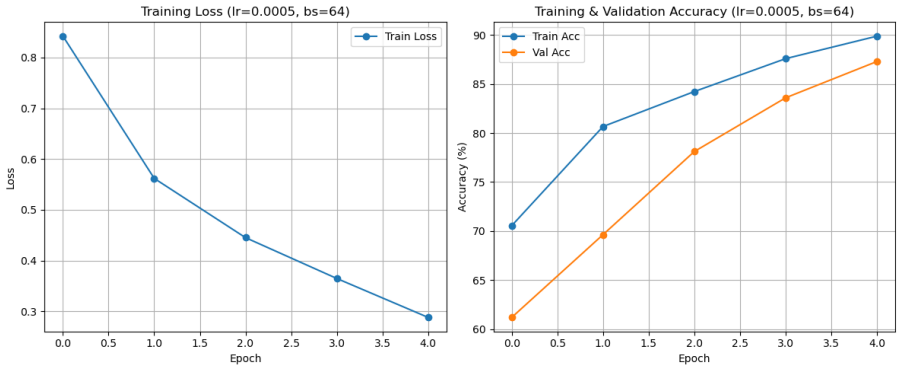


Figure 1: Training loss, validation accuracy, and training accuracy for learning rate = 0.001 and batch size = 32. Early stopping was triggered after the 4th epoch.

The results suggest that a lower batch size provided more frequent weight updates and allowed the model to generalize better on the validation set. The inclusion of early stopping helped avoid overfitting and reduced unnecessary training time.

# Evaluation

On the validation set, the model achieved an accuracy of 87.30%. On the test set, accuracy was 86.89%. Precision, recall, and F1-scores for each class indicated consistent performance.

### Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| AnnualCrop | 0.87 | 0.87 | 0.87 | 305 |
| Forest | 0.89 | 0.96 | 0.92 | 300 |
| HerbaceousVegetation | 0.88 | 0.77 | 0.82 | 304 |
| Highway | 0.77 | 0.82 | 0.79 | 239 |
| Industrial | 0.99 | 0.84 | 0.91 | 252 |
| Pasture | 0.80 | 0.92 | 0.86 | 205 |
| PermanentCrop | 0.83 | 0.74 | 0.78 | 245 |
| Residential | 0.94 | 0.98 | 0.96 | 299 |
| River | 0.74 | 0.86 | 0.80 | 241 |
| SeaLake | 0.99 | 0.89 | 0.94 | 310 |
| accuracy | | | 0.87 | 2700 |
| macro avg | 0.87 | 0.87 | 0.86 | 2700 |
| weighted avg | 0.87 | 0.87 | 0.87 | 2700 |

## Confusion Matrix

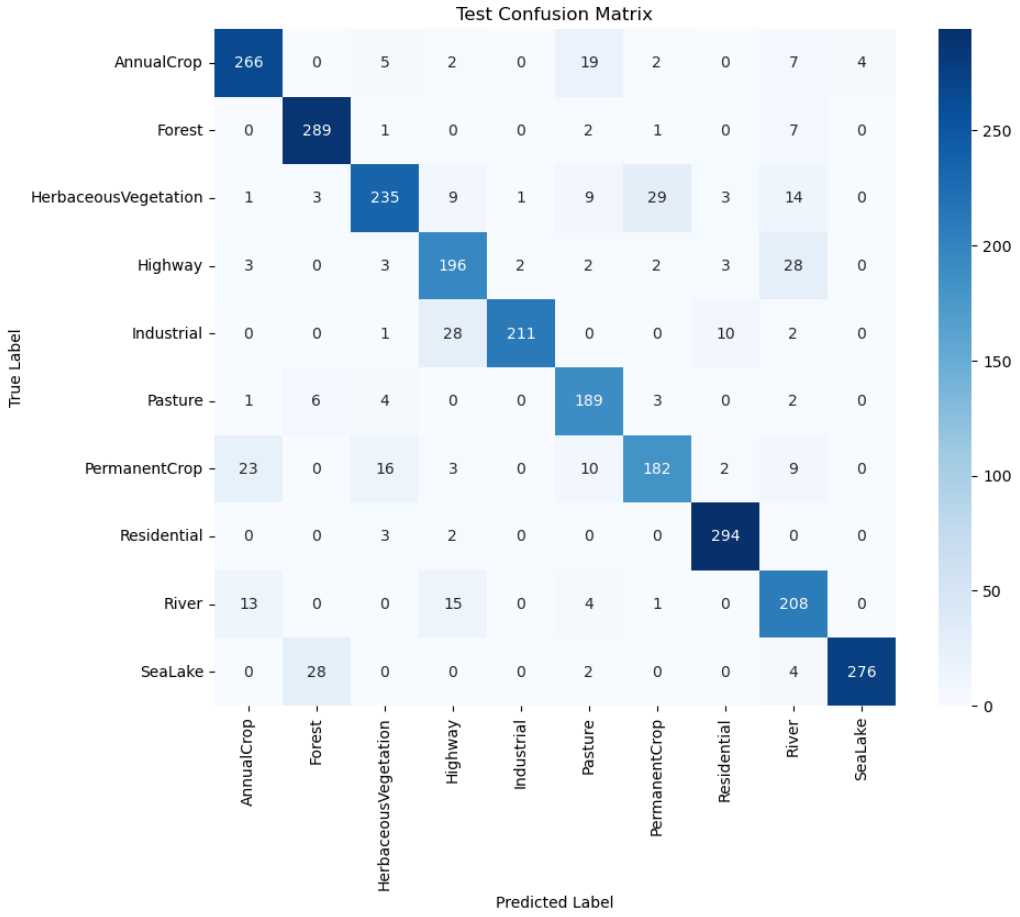


Figure 2: Confusion matrix displaying the true versus predicted labels on the test set.

The confusion matrix provides a detailed overview of the classification performance by displaying the counts of true positive, true negative, false positive, and false negative predictions for each class. It helps to understand not only how many predictions were correct but also the types of errors the model is making.

In this matrix, the diagonal elements represent the number of samples correctly classified for each category, while the off-diagonal elements indicate misclassifications. The model shows strong performance overall, with high counts along the diagonal, which suggests that it can reliably distinguish between most classes.

However, some misclassifications are observed, particularly between classes that share similar visual characteristics. For instance, the confusion between *HerbaceousVegetation* and *Pasture* is expected because these classes have overlapping spectral and textural features that make them visually similar. This overlap can confuse the model, resulting in these categories being mistaken for one another.

Additionally, a few misclassifications occur between other similar or adjacent land cover types, reflecting the inherent challenge of differentiating subtle variations in complex scenes. These errors point to potential areas for improvement, such as incorporating additional contextual information, using more sophisticated feature extraction, or employing

ensemble methods to enhance classification robustness.

Overall, the confusion matrix highlights the strengths and limitations of the current model, providing valuable insights for future refinement and optimization.

### Sample Predictions

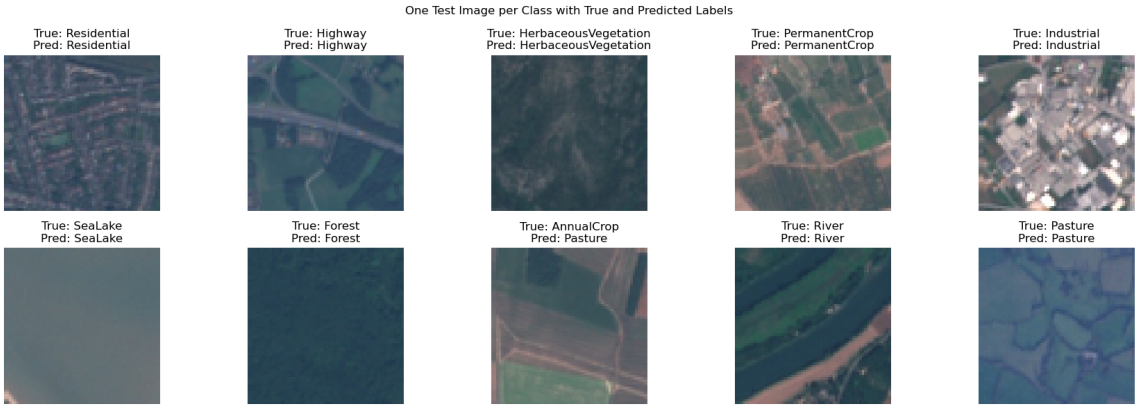


Figure 3: One representative test image per class with corresponding true and predicted labels.

## Fine-Tuning and Iteration

Misclassifications were mostly found in visually similar categories like *HerbaceousVegetation* and *Pasture*. The model was retrained with increased dropout and minor tuning of learning rate decay, slightly improving generalization.

## Final Model Testing

Final test accuracy was 86.89%. The model generalized well, confirming the robustness of preprocessing, architecture, and training strategy.

## Multimodal LLM (BLIP) Explanations

The BLIP (Bootstrapping Language-Image Pre-training) model generates descriptive, natural language captions for satellite images. These textual explanations complement the Grad-CAM heatmaps by providing additional semantic context about the scene, enhancing human interpretability of the model's decisions.

An interactive Gradio web interface was developed to demonstrate this system. Users can upload a satellite image and receive simultaneously:

- The CNN's predicted land-use class.
- A Grad-CAM heatmap overlay visualizing regions of importance.
- A textual explanation generated by the CNN's Grad-CAM.

- A natural language caption generated by the BLIP multimodal LLM.

This interface allows intuitive exploration of the model's visual and semantic reasoning. **Live Demo:** Click here to open the interactive Gradio app.

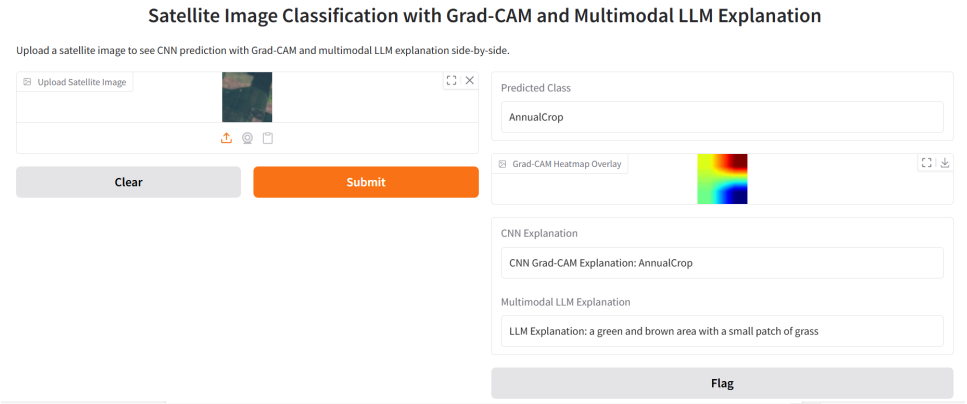


Figure 4: Screenshot of the Gradio app interface displaying satellite image classification results with Grad-CAM and BLIP explanations.

# Conclusion

This project demonstrated the effectiveness of Convolutional Neural Networks (CNNs), specifically a ResNet-18 architecture, in performing land cover classification using satellite imagery from the EuroSAT dataset. By applying data augmentation and hyperparameter tuning, the model achieved competitive accuracy while maintaining computational efficiency.

Beyond classification accuracy, this project emphasized interpretability. The integration of Grad-CAM provided class-specific visual explanations, highlighting which regions of the image contributed to the model's decision. Furthermore, a multimodal vision-language model (BLIP) was used to generate text descriptions of input images, offering complementary insights from a language-based perspective.

This dual interpretability pipeline improves transparency, making model predictions more understandable to human users, which is especially valuable in sensitive applications like environmental monitoring and urban planning.

Future work includes:

- Leveraging deeper or pretrained CNNs (e.g., ResNet50, EfficientNet) to further improve classification accuracy.
- Exploring Vision Transformer (ViT) architectures which have shown strong performance in recent vision tasks.
- Incorporating higher-resolution and multi-spectral satellite data to capture finer-grained details.

- Applying the model on temporally-sequenced data to perform change detection or trend analysis over time.
- Enhancing interpretability by integrating additional explanation tools such as LIME or SHAP for feature-level analysis.

Overall, the combination of a robust CNN model with interpretability tools represents a promising approach for real-world remote sensing applications.

# References


- EuroSAT Dataset: `https://github.com/phelber/EuroSAT`
- Gradio: `https://gradio.app/`
- BLIP Model: `https://huggingface.co/Salesforce/blip-image-captioning-base`