**Project: Land Type Classification using Sentinel-2 Satellite Images**

**Exploratory Data Analysis (EDA) Report:**

**1. Introduction**

This exploratory data analysis (EDA) performed on the EuroSAT dataset, which consists of Sentinel-2 satellite images categorized into different land use and land cover types. The goal of this project is to classify various land types using deep neural networks.

**2. Dataset Overview**

* **Dataset Name:** EuroSAT (Sentinel-2 RGB Images)
* **Source:** [EuroSAT Dataset](https://madm.dfki.de/files/sentinel/EuroSAT.zip)
* **Dataset Structure:**
  + The dataset is organized into 10 classes: AnnualCrop, Forest, HerbaceousVegetation, Highway, Industrial, Pasture, PermanentCrop, Residential, River, SeaLake.
  + Each class contains approximately 2000–3000 images.
  + A diagram of a data processing process

    AI-generated content may be incorrect.Image dimensions: 64x64 pixels with 3 color channels (RGB).

**3. Data Exploration**

**3.1 Class Distribution**

* The dataset is well-balanced across all 10 classes, with each class containing a comparable number of samples.

A graph of different colored bars

AI-generated content may be incorrect.

**3.2 Sample Images**

* Sample images from each class were visualized to understand visual differences among land types.

A collection of land with different colors

AI-generated content may be incorrect.

**3.3. Data Quality Assessment**

* No missing/corrupted images were detected in the dataset.
* All images are of uniform size (64x64 pixels), ensuring consistency in input dimensions for model training.

**3.4 RGB Pixel Distribution**

* This image shows the RGB pixel intensity distribution of an image, where the blue channel has the highest average frequency around intensity values of 80–100, indicating a dominant presence of blue tones.
* channel shows notable peaks at both low and high intensities, suggesting strong contrast and possible highlights or shadows.

**A graph of different colors

AI-generated content may be incorrect.**

**5. Data Preprocessing**

* **Image Transformation to Tensor:** All images were converted from PIL format to PyTorch tensors using transforms.ToTensor().
* **Normalization:** This step scales the data to have zero mean and unit variance, which helps the model train more efficiently and converge faster.
* **Label Handling**:The dataset was loaded using ImageFolder, which automatically assigns numeric class labels based on folder names (e.g., 0 for "Forest", 1 for "desert", etc.).
* **Data Augmentation:** Techniques such as rotation, flipping, and zooming were applied to increase dataset variability and improve model generalization.
* **Train/Validation/Test Split:** split data ( **Training**: 70%,**Validation**: 20%,**Testing**: 10%)
* Class Distribution After Splitting:

A graph of a number of blue and green bars

AI-generated content may be incorrect.

**PCA & t-SNE for Feature Visualization:**

* **PCA (Principal Component Analysis)** reduces dimensionality → quick visual check of data separability.
* **t-SNE (t-distributed Stochastic Neighbor Embedding)** is better for nonlinear separability.

**A group of colored dots

AI-generated content may be incorrect.**

**6. Summary of Findings**

* The dataset is well-structured and balanced, making it suitable for classification tasks.
* Images show clear distinctions between different land types, supporting effective model training.
* No missing or corrupt images were found, and preprocessing steps have been applied to ensure data consistency.
* High overlap in PCA & t-SNE means the dataset does not have well-separated clusters (not linearly separable).
* Simple classifiers like logistic regression or linear SVM may struggle, this tells us that even with a non-linear method, the data does not have naturally distinct groups.
* Further steps include implementing CNN-based models (e.g., ResNet, VGG) for classification and optimizing performance through hyperparameter tuning.

**7. Next Steps**

* Develop a baseline CNN model (ResNet) for classification.
* Perform model evaluation and hyperparameter tuning.
* Deploy the trained model using Flask/FastAPI for real-world applications.

**EDA Team:**

Ziad Mohamed shawky Youssif Mohamed abd Elkader Mostafa Mahmoud Mohamed

**Modelling:**

**Convolutional Neural Networks (CNN Model):**

**Convolutional Neural Network (CNN)** is a deep learning model designed for image processing tasks. It automatically extracts features from images using convolutional layers.

Key Components:

* **Convolutional Layers**: Detect patterns using filters (kernels)
* **Pooling Layers**: Reduce dimensions while preserving features
* **Fully Connected Layers**: Classify the extracted features
* **Activation Functions**: Typically ReLU for non-linearity

Advantages

* Captures spatial relationships in images
* Reduces number of parameters compared to traditional networks
* Suitable for image classification, object detection, and segmentation
* no manual feature extraction & engineering needed.

**First model: (Model Architecture Summary):**

1. Input Layer:

* Shape: (128, 128, 3) -> Accepts RGB images.

2. Convolutional Blocks:

* 5 convolutional blocks with increasing filter sizes: 32, 64, 128, 256, 512.
* Each block includes:
  + Conv2D with 3x3 filters for extract Features
  + BatchNormalization for stabilizing learning
  + LeakyReLU activation (α = 0.1) for non-linearity
  + MaxPooling2D to downsample features
  + Dropout (from the 3rd block onward) to prevent overfitting

3. Feature Aggregation:

* GlobalAveragePooling2D replaces fully connected flattening to reduce overfitting and dimensionality.

4. Fully Connected Layers:

* Dense(256) layer followed by LeakyReLU and Dropout(0.4)
* Final Dense layer with softmax activation to output probabilities across classes

Compilation:

* Optimizer: Adam (learning rate = 0.001)
* Loss Function: Categorical Crossentropy (suitable for multi-class classification)
* Metric: Accuracy

Callbacks:

* EarlyStopping: Stops training when validation loss stops improving (patience = 5)
* ReduceLROnPlateau: Reduces learning rate by half if validation loss stagnates for 3 epochs

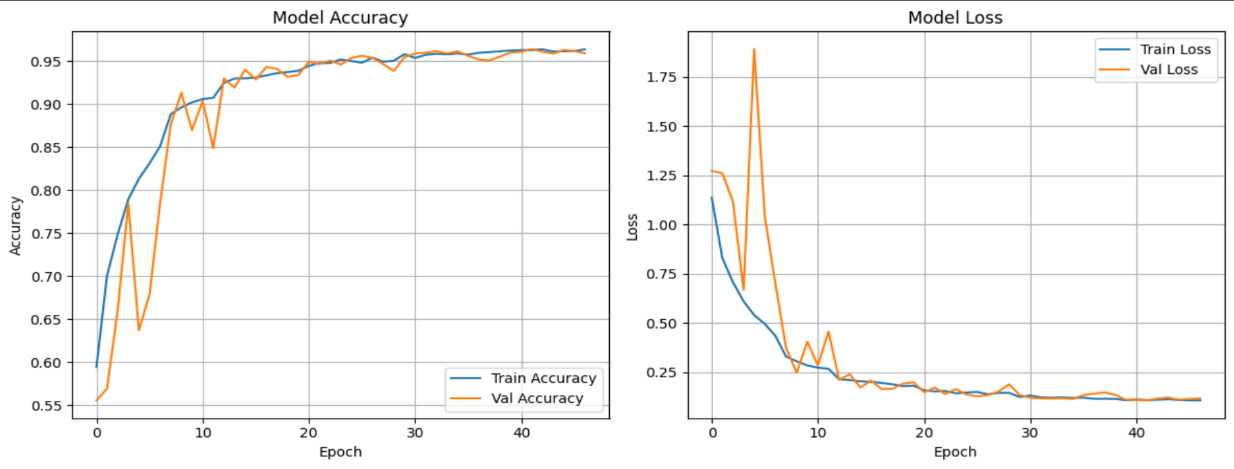
Output:

* Softmax probabilities for classification into len(categories) classes.

**A screenshot of a computer screen

AI-generated content may be incorrect.A graph with numbers and labels

AI-generated content may be incorrect.Confusion Matrix: Classification Report:**



**Second model: (Model Architecture Summary):**

1. Input Layer:  
Shape: (32, 32, 3) → Accepts smaller RGB images.

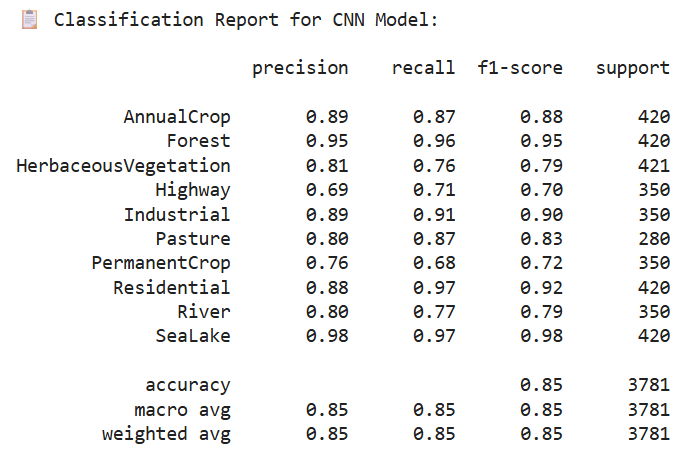
2. Convolutional Blocks:  
• Two convolutional layers to extract low- and mid-level features:  
  o Conv2D(32 filters, 3×3) with ReLU activation → extracts basic edges and patterns  
  o Conv2D(64 filters, 3×3) with ReLU activation → learns more complex features  
• MaxPooling2D(pool\_size=2×2): reduces spatial dimensions (downsampling)  
• Dropout(0.25): randomly deactivates neurons to prevent overfitting

3. Feature Flattening:  
• Flatten(): Converts 3D feature maps into 1D vector for the dense layers

4. Fully Connected Layers:  
• Dense(128) with ReLU activation → learns global patterns from features  
• Dropout(0.5): aggressively reduces overfitting before the final layer  
• Final Dense layer with softmax activation → outputs class probabilities (len(class\_names) classes)

Compilation:  
• Optimizer: Adam  
• Loss Function: Categorical Crossentropy (used for multi-class classification)  
• Metric: Accuracy

Training:  
• Batch Size: 128 → processes 128 images per training step  
• Epochs: 50 → model trains for up to 50 full passes through data  
• Validation: Uses (x\_valid, y\_valid) for monitoring performance during training

A screenshot of a graph

AI-generated content may be incorrect.**Confusion Matrix: Classification Report:**

A graph of a line and a line

AI-generated content may be incorrect.

**Residual Neural Network (ResNet Model):**

**ResNet** **(Residual Neural Network)** is a deep learning architecture specifically designed to train very deep networks efficiently by using skip (shortcut) connections.

Key Components:  
• Residual Blocks: Core units that add shortcut connections to bypass layers  
• Skip Connections: Allow gradients to flow directly through the network, preventing vanishing gradients  
• Batch Normalization: Normalizes activations to stabilize and speed up training  
• Global Average Pooling: Reduces dimensions before the output layer, minimizing overfitting  
• Fully Connected Layer: Final layer that outputs class probabilities or logits

Advantages:  
• Enables training of very deep networks (e.g., 18, 34, 50+ layers)  
• Avoids vanishing gradient problem with identity mappings  
• Improves accuracy by learning residual functions instead of full transformations  
• Outperforms traditional CNNs on many benchmark image classification tasks  
• Modular design – easy to scale by stacking more residual blocks

**Model Architecture Summary**:  
1. Input Layer:  
• Shape: (64, 64, 3) → Accepts resized RGB images  
• Modified First Convolution:  
  o Conv2D(3→64, kernel size=3×3, stride=1, padding=1) replaces default 7×7 to suit smaller images

2. Backbone Architecture:  
• Base Model: ResNet18 (no pre-trained weights used)  
• Composition:  
  o 4 residual blocks with skip connections  
  o Each block contains multiple convolutional layers with batch normalization and ReLU  
  o Downsampling via stride or max pooling inside residual blocks  
• Advantage: Deep architecture with identity mappings enables better gradient flow and deeper learning

3. Classification Head:  
• Global Average Pooling → replaces flattening for reducing overfitting  
• Final Fully Connected Layer:  
  o Linear(in\_features → 10) → Predicts class logits for 10 classes

4. Compilation & Optimization:  
• Loss Function: CrossEntropyLoss → suitable for multi-class classification  
• Optimizers: Randomly selected between:  
  o Adam → adaptive learning  
  o SGD with momentum (0.9) → stable convergence  
• Learning Rate: Randomly selected from [0.001, 0.0005, 0.0001]

5. Training Configuration:  
• Epochs: Randomly selected from [5, 10, 15]  
• Batch Size: Randomly selected from [16, 32, 64]  
• Random Search Strategy:  
  o Performs 5 random combinations of hyperparameters  
  o Tracks best model based on validation accuracy

A screenshot of a graph

AI-generated content may be incorrect.6. Evaluation & Output:  
• During each trial:  
  o Validation accuracy is computed after each epoch  
  o Best model (highest val acc) is stored  
• Final Outputs:  
  o Best hyperparameter configuration  
  o Best model saved as best\_resnet\_model.pth  
  o Highest achieved validation accuracy printed

A screenshot of a graph

AI-generated content may be incorrect.

**You Only Look Once (YOLO Model):**

**YOLO** is a real-time object detection model that detects and classifies multiple objects in an image using a single forward pass of a neural network. It divides the image into a grid and simultaneously predicts bounding boxes and class probabilities.

Key Components:  
• Backbone: Feature extractor (e.g., CSPDarknet in YOLOv8) that processes input images  
• Neck: Combines features from different scales (e.g., PANet or FPN)  
• Head: Predicts bounding box coordinates, objectness scores, and class labels  
• Bounding Boxes: Defined by center x/y, width, height — converted to YOLO format  
• YOLO Format: Normalized coordinates for efficient training and inference

Advantages:  
• Real-time object detection (fast inference speed)  
• Unified architecture — detection done in one step  
• High accuracy with fewer false positives  
• Efficient with limited hardware (mobile-friendly models like YOLOv8n)  
• Easy to fine-tune on custom datasets (e.g., EuroSAT used above)

**YOLOv8 Model Architecture Summary**

**YOLOv8** is the latest evolution of the YOLO (You Only Look Once) object detection family. It’s designed for real-time, high-speed detection with improved accuracy and flexibility.

**A diagram of a computer program

AI-generated content may be incorrect.**

1. Backbone (Feature Extraction):  
   • CSPDarknet (Cross Stage Partial Network)  
   • Extracts rich spatial and semantic features from the input image  
   • Uses a combination of convolutional blocks, bottlenecks, and residual-like connections
2. Neck (Feature Aggregation):  
   • BiFPN (Bidirectional Feature Pyramid Network)  
   • Combines features from different levels (scales) of the backbone  
   • Enhances context awareness for both small and large objects
3. Head (Prediction Layer):  
   • Anchor-free detection head  
   • Directly predicts:  
   – Class probabilities  
   – Objectness score (confidence)  
   – Bounding box coordinates (x, y, w, h)  
   • Uses one detection layer per scale
4. Activation Functions:  
   • SiLU (Swish) activation used throughout for non-linearity  
   • Improves gradient flow and generalization
5. A screenshot of a computer screen

   AI-generated content may be incorrect.A graph with blue and orange lines

   AI-generated content may be incorrect.Output Format:  
   • For each bounding box:  
   class\_id, x\_center, y\_center, width, height (all normalized)  
   • Supports multi-class, multi-object predictions in one image

**Model Team:**

Yasmine Yasser Abdelwadood Rana Alaa Ahmed Ahmed

Seif Eldeen khaled Nabil Mostafa Mahmoud Mohamed

* A diagram of a target with arrows

  AI-generated content may be incorrect.**A screenshot of a computer

  AI-generated content may be incorrect.Model Deployment:** (Platform: HuggingFace)

The website is allows users to upload satellite images of land. Once an image is uploaded through the frontend interface, it is sent to the backend via a **REST API**. The backend processes the image and passes it to a pre-trained deep learning model, which analyzes the content and classifies the land type (e.g., agricultural, urban, desert, water). The classification result, along with a confidence score, is sent back to the frontend and displayed to the user. Optionally, the system can store the image and classification results in a database for future reference.

A river with a city and buildings

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.**Our Deployment** [**Link**](https://huggingface.co/spaces/YasmineOribi/type_land_classification)

**Deployment Team:**

Yasmine Yasser Abdelwadood Rana Alaa Ahmed Ahmed

Seif Eldeen khaled Nabil Youssif Mohamed abd Elkader

**MLflow (**for first CNN model)

To facilitate model tracking, reproducibility, and future deployment.

A screenshot of a computer

AI-generated content may be incorrect.

**Role Member**: Mostafa Mahmoud Mohamed

**Project Team Members:**

**● Yasmine Yasser Abdelwadood**

**● Rana Alaa Ahmed Ahmed**

**● Seif Eldeen khaled Nabil**

**● Youssef Mohamed Abdelkader**

**● Mostafa Mahmoud Mohamed**

**● Ziad Mohamed shawqi**

Thank you