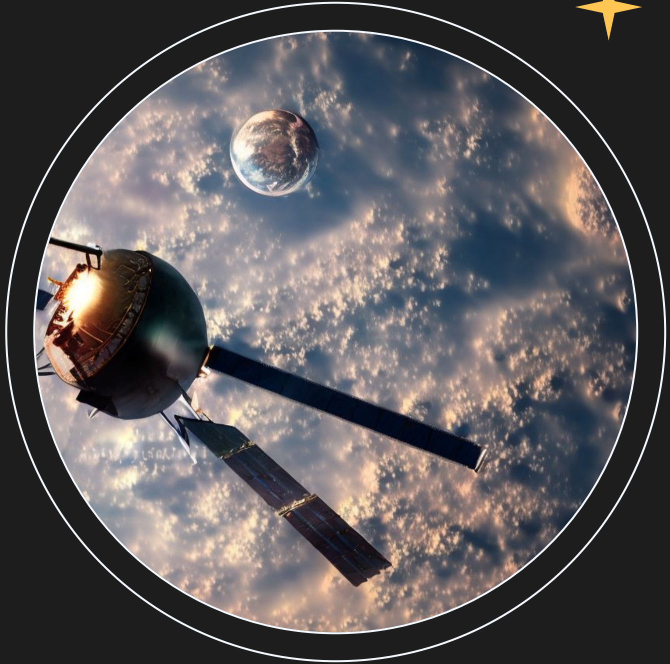
Object Detection in Satellite Imagery

Google/Aftershoot



Challenge Advisors: Hrishikesh Garud and Juliana Chyzhova

By: Aishu Vinod, Mantra Burugu, Isabelle Wang, Mina Yang, Ananya Kadwe

Meet the Team!

Aishu Vinod
Northeastern University

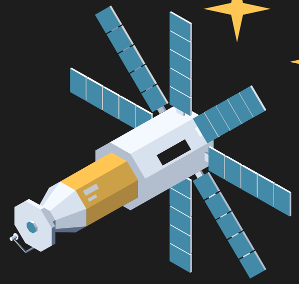
Ananya Kadwe
Brandeis University

Mina Yang
Amherst College

Mantra Burugu
University of Massachusetts

Isabelle Wang
Smith College

Presentation Agenda



1. Introduction and Project Overview
2. Data Preparation and Pre-Processing
3. Model Selection and Evaluation
    a. Support Vector Machine
    b. Convolutional Neural Network
4. Results
5. Challenges
6. Next Steps
7. Questions

Introduction: The Role of ML in Satellite Imagery Analysis

- Our Goal: Develop a machine learning model for object detection and classification in satellite imagery.
- Detect and classify buildings and land cover types using neural networks.
- Use Cases:
  - NASA, ESA, and JAXA
  - Urban Planning
  - Disaster Response
  - Environmental Monitoring

Impact to companies/Business impact

aftershoot

Transform your photography workflow with AI assisted Culling & Editing

"Google's mission is to organize the world's information and make it universally accessible and useful."

Google

1. **Google**

- Improves Google's geospatial services (e.g., Google Earth, Google Maps)
- Supports Google's mission to organize and information universally accessible and useful

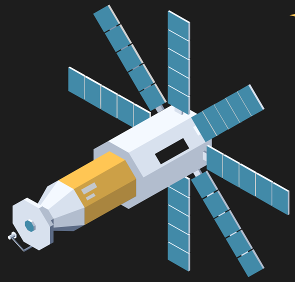
2. **Aftershoot**

- Reinforces commitment to creativity and cutting-edge AI solutions in photography
- Broaden reach into fields like satellite image analysis

Tools

- VSCode
  - Code is written in Python
  - Used Tensorflow Keras for CNN Model
- Google Colab
- Google Gemini
- Stackoverflow

Ethical Considerations



1. Privacy and Surveillance
2. Bias in Data and Algorithms
3. Environmental Impact
4. Dual-Use Implications

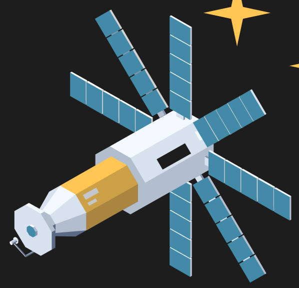
Dataset Overview

- ★ EuroSAT Dataset
  - ○ Public dataset for remote sensing and geospatial analysis.
- ★ Satellite images in JPEG format
- ★ 27,011 images with a total size of 91.9 MB.
- ★ The dataset is largely balanced, with each category containing around 2,000 to 3,000 images.
- ★ Ten categories: Forest, Annual Crop, Sea/Lake, Herbaceous Vegetation, Residential, Permanent Crop, River, Highway, Industrial, Pasture





Our approach



- ★ Our problem is a supervised learning classification problem.
  - Multiclass
- ★ Define label and features
- ★ Start with a simple Support Vector Machine Model
- ★ Final model will be a Convolutional Neural Network Model

Label and features

Features:

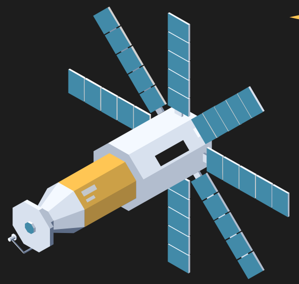
- individual pixels of images
- Total of 12,288 pixels per example

pixel_0	pixel_1	pixel_2	pixel_3	pixel_4	pixel_5	pixel_6	pixel_7	pixel_8	pixel_9	pixel_10	pixel_11	pixel_12
43	65	79	43	65	79	47	64	80	47	64	80	44
41	69	81	41	69	81	43	71	82	44	72	83	43
30	54	66	30	54	66	30	54	66	29	53	65	30
44	66	80	45	67	81	44	65	82	49	70	87	48
40	59	76	41	60	77	42	63	80	39	62	78	36
32	54	67	32	54	67	32	51	65	31	53	66	26
45	70	75	46	71	76	40	67	76	40	67	76	39
40	65	70	38	66	70	37	64	71	36	65	71	38
41	80	75	41	80	75	46	82	80	43	77	78	36
43	70	77	43	70	77	44	68	78	43	68	75	42
34	56	67	34	56	67	33	57	69	33	57	69	32

Labels

- Ten categories, one-hot encoded from 0-9
  - 0 Annual Crop
  - 1 Forest
  - 2 Herbaceous Vegetation
  - 3 Highway
  - 4 Industrial
  - 5 Pasture
  - 6 Permanent Crop
  - 7 Residential
  - 8 River
  - 9 SeaLake

Data preparation and pre-processing



- ★ All images resized to 64x64 pixels using OpenCV.
- ★ Pixel values will be normalized to a range of 0 to 1 by dividing by 255.0.
- ★ Ensured uniformity in color representation.
- ★ Images are flattened into one-dimensional arrays, each representing the pixel values.
  - 12288 pixels per image
- ★ Ensure correct labeling for training.
  - Ten categories labeled as 0-9
- ★ One hot encoded target variables
- ★ Split data into training and test split
  - Proportionate for each category
  - 80/20 split

Challenges - Data Processing

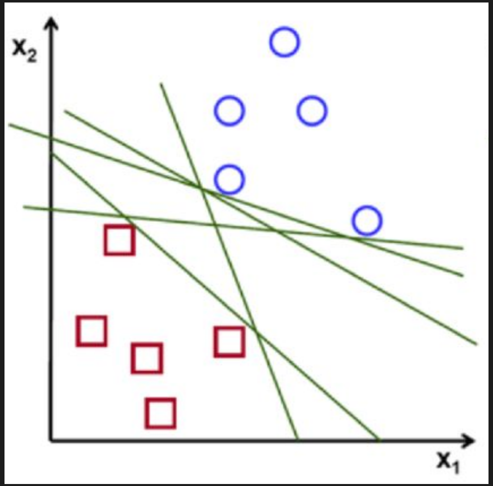
★ High-dimensional image data in a CSV format

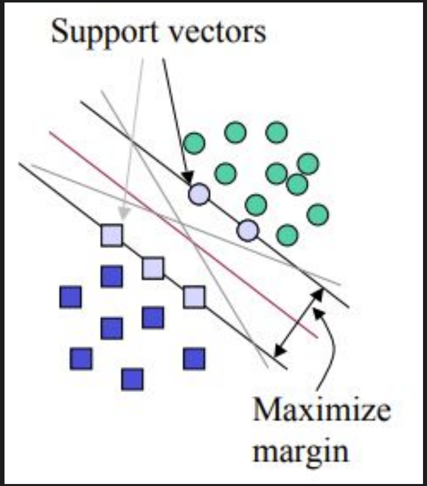
★ Large file slowed down VSCode/Colab

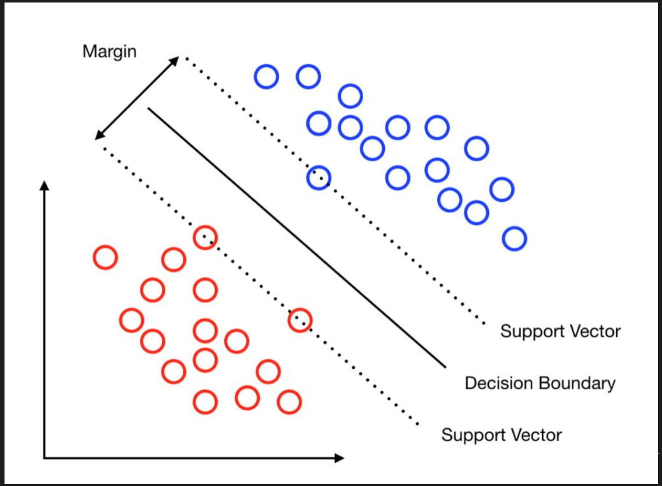
★ Unstructured representation of data made it difficult to assess accuracy of pre-processing steps

Support Vector Machine

- ★ First model selection using the Support Vector Machine Library (svm.SVC)
- ★ Support Vector Machine (SVM): A type of supervised learning method that uses a hyperplane or line for classification



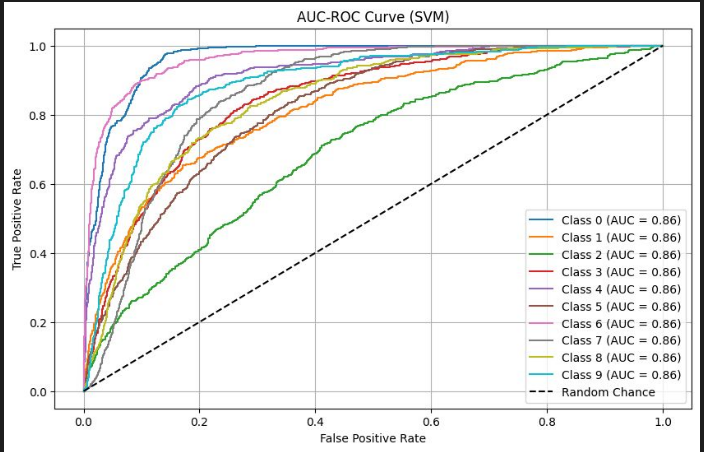




Training the SVM

- ★ Label is the categories ranged 0-9 for each of the category
- ★ Features are the individual 12288 pixels
- ★ Initial hyperparameters of the model:
  - kernel = ‘linear’
  - Regularization parameter: C = 5.0
  - PCA: 100
- ★ Train test split parameters: 80/20, and stratified = “y”
- ★ Accuracy Score: 0.4548

SVM - Metrics



Confusion Matrix and Classification Report

	Class	Precision	Recall	F1-Score
0	0	0.65	0.72	0.68
1	1	0.38	0.39	0.38
2	2	0.24	0.2	0.22
3	3	0.35	0.4	0.37
4	4	0.53	0.62	0.57
5	5	0.45	0.43	0.44
6	6	0.73	0.64	0.68
7	7	0.42	0.5	0.45
8	8	0.32	0.22	0.26
9	9	0.39	0.33	0.35

Summary Metrics

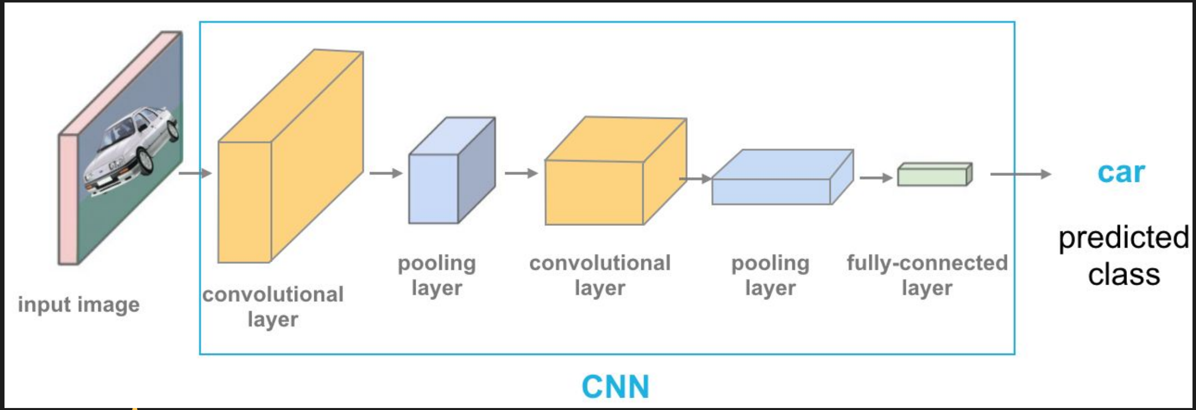
	Metric	Precision	Recall	F1-Score
0	Accuracy	0.45	0.45	0.45
1	Macro Avg	0.44	0.44	0.44
2	Weighted Avg	0.45	0.45	0.45

Challenges - SVM Training

- Flattening 12,288 features made dataset highly dimensional, took a long time to parse through the dataset
- Increased computational requirements to train SVM model
- Longer training times and increased memory usage
- Difficult to quickly experiment with different models or hyperparameters

Convolutional Neural Network

- ★ Convolutional Neural Network (CNN): A type of supervised machine learning algorithm used for processing images.
- ★ Input layer -> hidden layers -> Output layer




```
cnn_model = keras.Sequential()

# 1. Input layer
input_layer = keras.layers.InputLayer(input_shape=(64, 64, 3))
cnn_model.add(input_layer)

# 2. First convolutional block
cnn_model.add(keras.layers.Conv2D(16, (3, 3), padding="same"))
cnn_model.add(keras.layers.BatchNormalization())
cnn_model.add(keras.layers.ReLU())
```

Training the CNN model

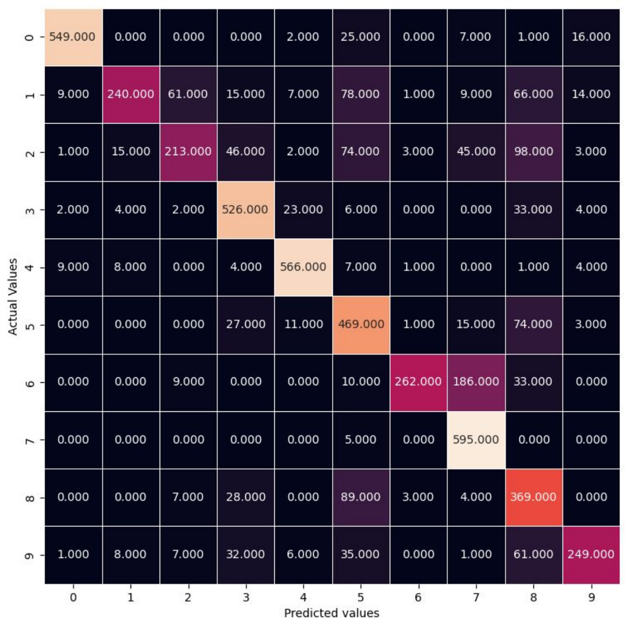
- Our model
  - Dimensions of images are 64 by 64 with 3 channels
  - 80/10/10 split
  - Images normalized by dividing the every pixels by 255.0
  - Activation function: softmax and ReLU
  - Number of filters: 16, 32, 64, 128, 256
  - Optimizer: Stochastic Gradient Descent with learning rate of 0.1
  - Loss function: Sparse Categorical Cross Entropy
  - Metrics: Accuracy
  - Epochs: Trained for 1, 5, 8, 20 epochs with four layers
  - Layers: 4 layers. Also trained for 3 and 5 layers.

CNN Model Results

Model	Number of Hidden Layers	Number of Epochs	Test Accuracy Score	Test Log Loss (Cross Entropy)
A	4	1	38.204%	3.337
B	4	4	52.278%	2.360
C	4	5	64.167%	1.315
D	4	6	75.741%	0.713
E	4	8	55.148%	1.554
F	4	20	63.741%	1.347
G	3 (Delete)	1	45.333%	1.926
H	5 (Add)	1	49.222%	2.029
I	5 (Add)	5	29.963%	5.209

Results for our chosen model:

The chosen model we chose is Model D, with an accuracy score of 76% and with 6 epochs and 4 .

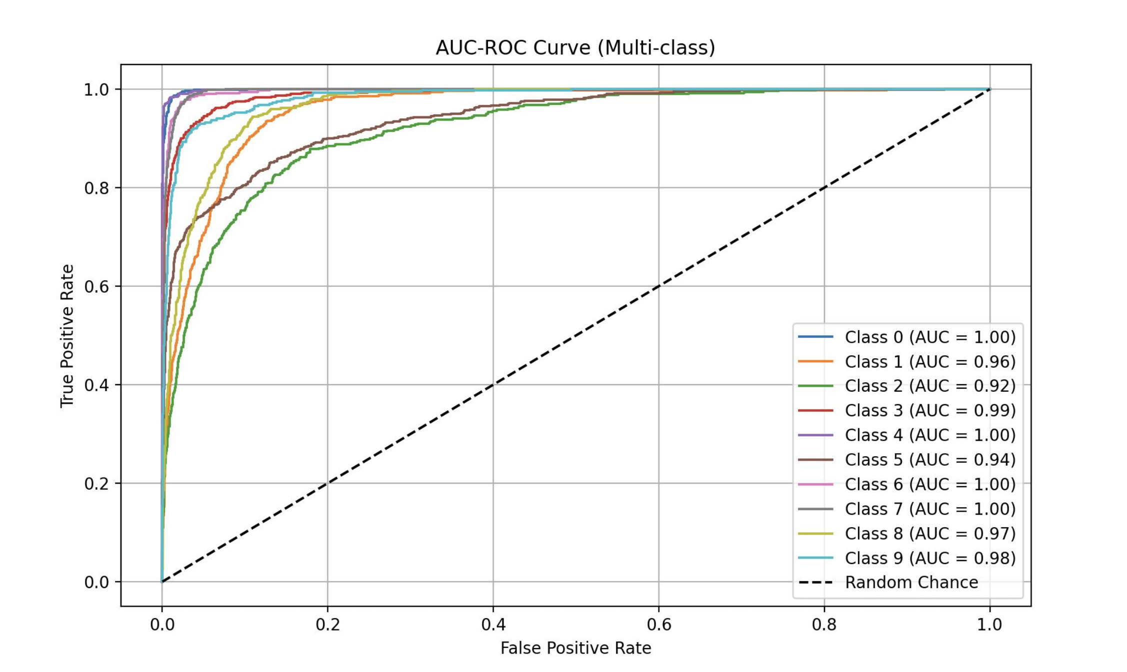


Classification Report:

	precision	recall	f1-score	support
0	0.93	0.97	0.95	600
1	0.69	0.56	0.62	500
2	0.94	0.06	0.12	500
3	0.83	0.86	0.85	600
4	0.94	0.97	0.95	600
5	0.76	0.67	0.71	600
6	0.98	0.71	0.82	500
7	0.91	0.90	0.90	600
8	0.39	0.96	0.56	500
9	0.82	0.81	0.81	400
accuracy			0.76	5400
macro avg	0.82	0.75	0.73	5400
weighted avg	0.82	0.76	0.74	5400

CNN Model Results (Cont.)

ROC-AUC



Challenges with the CNN Model

- Run time:
  - The 20 epochs with four layer took 2 hours (each epoch roughly 10 minutes)
- Finding the best hyperparameters
  - There were a myriad of hyperparameters that contributed to the accuracy and performance of the model. We weren't able to find the best value for each hyperparameter, so we chose the most important parameters first and focused on tuning that–epochs and layers.

Model comparison

Model Name	Description	Results	Pros	Cons
Support Vector Machine (SVM)	Classical ML method used for supervised learning that finds best hyperplane separating classes	Accuracy: 45.48%	Easier to implement, less prone to overfitting than CNN	- Time consuming to train for nonlinear (poly and rfb) kernels
Convolutional Neural Network (CNN)	Neural network with hidden layers and used for supervised learning	Accuracy: 75.741%	Automatic feature extraction, suitable for large datasets	- Time consuming to train for higher epochs

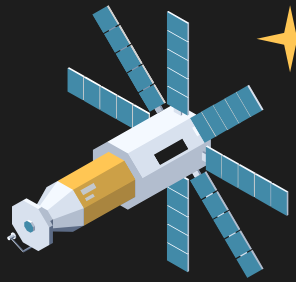
Next Steps...

★ **Continue Tuning and Validating**

- ★ Experiment with further hyperparameter tuning
- ★ Validate the model on unseen datasets

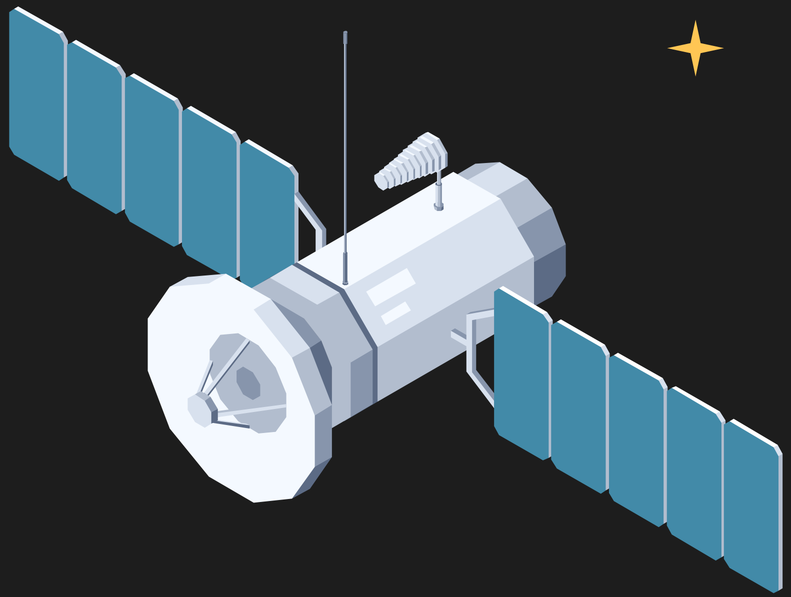
★ **Deployment**

- ★ Save model
- ★ Create a REST API to serve the model in production
- ★ Flask App
- ★ Implement monitoring to track the model's performance in production, checking for signs of model drift or performance degradation over time. This could involve periodic retraining as new data becomes available.



Acknowledgements

Thank you to Break Through Tech Boston @MIT, our TA Helen Bang, and our Challenge Advisors Hrishikesh Garud and Juliana Chyzhova!



Thank You!