**AGRICULTURE LAND USE CLASSIFICATION**

**A CAPSTONE PROJECT REPORT**

*Submitted in partial fulfillment of the*

*requirement for the award of the*

*Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

*by*

**BATHULA GURU SUBHASH (21BCE9054)**

**KUNCHALA RAGAVENDRA (21BCE9009)**

**KASIREDDY BHOOMIKA (21BCE9255)**

**YENNE BHAGYA SRI (21BCE9258)**

*Under the Guidance of*

**DR. NAGAJAGADESH BOMMAGANI**



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI- 522237

*NOVEMBER 2024*

**CERTIFICATE**

This is to certify that the Capstone Project work titled “**AGRICULTURE LAND USE CLASSIFICATION**” that is being submitted by **BATHULA GURU SUBHASH (21BCE9056), KUNCHALA RAGAVENDRA (21BCE9009), KASIREDDY BHOOMIKA (21BCE9255), and YENNE BHAGYA SRI (21BCE9258)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of Bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.



Prof NAGAJAGADESH BOMMAGANI



**The thesis is satisfactory / unsatisfactory**

**Internal Examiner1 Internal Examiner2**

**Approved by**

HoD, Department of

School of Computer Science and Engineering

**ACKNOWLEDGEMENTS**

We would like to express our heartfelt gratitude to everyone who supported and guided us throughout the course of this project**, *Agriculture Land Use Classification***.

First and foremost, we are deeply grateful to our guide, **Prof NAGAJAGADESH BOMMAGANI**, for their invaluable insights, encouragement, and unwavering support. Their guidance was instrumental in shaping this project and overcoming challenges along the way.

We extend our thanks to the faculty and staff of the **School of Computer Science and Engineering, VIT-AP University**, for providing us with the resources and a conducive environment to carry out this research.

We would like to acknowledge the support of our peers and colleagues, whose constructive suggestions and camaraderie made this journey more enjoyable and collaborative.

Finally, we are profoundly grateful to our families for their constant encouragement, patience, and understanding, which motivated us to strive for excellence.

This project would not have been possible without the collective efforts and support of all the above-mentioned individuals and institutions. Thank you!

**ABSTRACT**

The project termed as Agriculture Land Use Classification uses AI and overhead views with the aim of solving a real issue/introduction – the problem of properly classifying land for purposes of agriculture. This study delves into approaches including logistic regression, sequential neural networks (SNNs) improving as well as the current breed of Convolutional Neural Networks (CNNs) such as ResNet34 and VGG16, by the use of the EuroSAT dataset that contains nothing but multi-spectral images of different land cover.

Under the FastAI framework, these models are realized with more focus on the data flow processes such as normalization and data augmentation of the model to prevent dynamism challenge and assurance of better performance in unseen data. Using these models, the simplest of them all was logistic regression, SNNs, and CNNs had better extraction of features from the data and performed better in classification. Of the various architectures of CNN, the adaptation of ResNet34 that reintroduces the learned features via skip connections is very effective when applied to the problem of classification with complex satellite images.

The results obtained indicate that CNNs improve the performance of image classification processes when compared to the use of conventional approaches. These findings are extremely important for the activity of agriculture considering that such activities can allow better use of resources, on the monitoring of crops, and on the protection of the environment.

Prospective changes involve the addition of transfer learning, the enlargement of the datasets, and the addition of systems for in the course monitoring at sights. This strives to show how deep learning, combined with satellite imagery, can revolutionize the agricultural landscape in a bid for sustainability and accuracy in land acquisition.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Chapter** | **Title** | **Page Number** |
| **1.** |  | **Acknowledgement** | **2** |
| **2.** |  | **Abstract** | **3** |
| **3.** | **1**  **1.1**  **1.2**  **1.3** | **Introduction**  **Objectives**  **Background and Literature Survey**  **Organization of the Report** | **5**  **6**  **6**  **7** |
| **4.** | **2**  **2.1**  **2.2**  **2.3** | **Chapter Title (Work)**  **Proposed System**  **Working Methodology**  **Software** | **8**  **9-10**  **11** |
| **5.** | **3** | **Results and Discussion** | **12-18** |
| **6.** | **4** | **Conclusion & Future Works** | **19-20** |
| **7.** | **5** | **Appendix** | **21-29** |
| **8.** | **6** | **References** | **30** |

**CHAPTER 1**

**INTRODUCTION**

Agriculture holds a vital position in achieving food security along with being the beyond backbone of economies in various parts of the world. Nonetheless, with the growing global population and varying climatic conditions, sustainable management of agricultural resources has become a necessity. The process of understanding and categorizing agricultural land use is crucial for the proper distribution of resources, effective crop monitoring, and the formulation of long-term plans. The task in this project, the Agriculture Land Use Classification, is to provide an accurate land cover classification that uses machine learning techniques.

The approach utilizes satellite imagery from the EuroSAT database, which contains a wide range of multi-spectral images from viols to slums, among other types of land use. Furthermore, the exploitation of deep learning, allows modelling and testing of various models, including logistic regression, sequential neural networks, and CNNs based on actual CNNs like ResNet34 and VGG16. Using FastAI library from PyTorch, the cutting-edge models are developed by conducting advanced preprocessing, and data augmentation techniques to increase accurateness and robustness.

Deliverables from the project are manifested by the provision of precise land classification insights which are expected to lead to the use of sustainable agricultural practices. Through insight into well-informed decision-making processes, what could be the ripple effect could be the changing of how agricultural land is constantly tracked and managed, thus ensuring the sector remains efficient and sustainable.

* 1. **Objectives**

The following are the objectives of this project:

* Classify agricultural land use using satellite imagery from the EuroSAT dataset.
* Analyze and compare the performance of logistic regression, sequential neural networks, and CNNs like ResNet34 and VGG16.
* Utilize the FastAI framework for implementing machine learning models.
* Apply preprocessing and data augmentation techniques to improve model performance.
* Provide accurate land classification insights to support sustainable agricultural practices.
* Enhance resource management and contribute to precision agriculture.
* Demonstrate the potential of machine learning in addressing agricultural and environmental challenges.
  1. **Background and Literature Survey**

Agricultural land use designations are important for both the management and monitoring of natural resources. Growing population, hunger, and environmental challenges require proper land use classification which in return can assist in decision making, resource management and farming policies for improving food production. Machine learning and remote sensing technologies can be employed with a great potential to meet these issues. Because of its capacity to deliver high quality and wide scope data, satellite images have also become a vital resource in the study of land use.

The result obtained from EuroSat data set which was derived from Sentinel-2 satellite images shows a large selection of multi spectral images in 10 classes comprising of agricultural and Non agricultural land types. This is a dataset that has been used a great deal in research for its relevance and diversity to machine learning operations. Research that has been performed with the aid of EuroSAT shows improvements in the coverage of earth’s surface when using classic or even modern described procedures. Previously, logistic regression has been deployed as a standard approach to determine benchmark results for several classification problems. On the other hand, this model does not address complexity.

Preprocessing techniques such as resizing, normalization, and data augmentation have proven essential in improving model performance and generalization. Techniques like random rotations, flipping, and contrast adjustments enhance the diversity of the training dataset, reducing overfitting and boosting robustness. Such methods have been widely employed in research to improve the effectiveness of machine learning models in agricultural and environmental studies.In previous works, researchers have highlighted the importance of combining satellite imagery with deep learning techniques for agricultural applications. Studies have explored the use of CNNs to detect crop types, monitor soil health, and predict yield. The findings underscore the potential of these technologies in enabling precision agriculture and sustainable farming practices. However, challenges remain, such as optimizing computational resources, improving model interpretability, and addressing region-specific variations in agricultural practices.

This further investigates intensive studies about the EuroSAT dataset and logistic regression, sequential neural networks, and advanced CNNs for agricultural land use classification. What this project does is cover preprocessing, data augmentation, and performance assessment in this endeavor to enrich the expanding pool of knowledge in sustainable agriculture and machine-learning applications. It also aims to keep doing best for the land classification work to improve the accuracy of classification and facilitate intelligent decision-making in resource management and agricultural planning.

**1.3 Organization of the Report**

The remaining chapters of the project report are described as follows:

* Chapter 2 contains the proposed system, methodology, hardware and software details.
* Chapter 3 gives the cost involved in the implementation of the project.
* Chapter 4 discusses the results obtained after the project was implemented.
* Chapter 5 concludes the report.
* Chapter 6 consists of codes.
* Chapter 7 gives references.

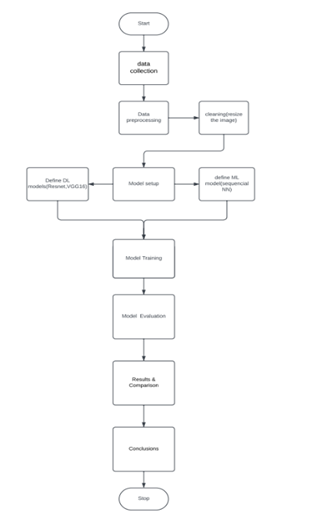
**CHAPTER 2**

This Chapter describes the proposed system, working methodology, software and hardware details.

**2.1 Proposed System**

This project proposes a system that will classify agricultural land use indifferently obtained satellite images, such as European Satellite data. Models to be utilized are machine learning models, including logistic regression, sequential neural networks, and improved models based on convolutional neural networks: ResNet34 and VGG16. Using these models, the fastai framework would be able to analyze land pattern coverages effectively.This would contribute very largelytowards the enhancement of accuracy in the classification of land usage that would provide

results to support sustainable agricultural practice, precision farming, and improved resource management.



**Figure 2. System Block Diagram**

**2.2**  **Working Methodology**

**Dataset:** EuroSAT dataset, which includes multi-spectral satellite images categorized into various land-cover classes.

**Preprocessing:** Data resizing, normalization, and augmentation (e.g., rotation, flipping) to improve model performance and reduce overfitting and Data augmentation.

Data augmentation:

The plan for the augmentation of data was based on the idea of creating some sort of variety in the dataset so as to increase robustness against overfitting through different manipulations done on the training images. For this, it randomly zoomed and rotated up to angles of about 50-degrees, implemented a horizontal flip, then shifted the source images in width and height, and also sheared them. These random augmentations yielded different copies of the original images so that invariant features can learn in the models. Images were further normalized using the rescale and preprocessing functions found in ImageDataGenerator to scale pixel values to 0 and 1 and prep to VGG16 pre-processing requirements.

The aim is to keep the best of pre-trained features but at the same time not leave out task-specific learning on a varying dataset. Combining fine-tuning and data augmentation resulted in improved accuracy as well as robustness, proving the value of applied strategies in transfer learning pipelines.

**Model Selection:**

**Logistic Regression:** Baseline model for performance comparison.

**Sequential Neural Networks:** Intermediate-level model for enhanced feature extraction.

**CNNs (e.g., ResNet34, VGG16):** Advanced models for high accuracy and robust performance.

1.)Model Architectures and Training :

I)Logistic Regression:

As a baseline model, logistic regression was implemented to provide a benchmark for the more complex models. Logistic regression is typically limited with image data, as it lacks hierarchical feature learning, but it offers interpretability and simplicity for comparison.

1. Linear Model:  
   z = β₀ + β₁x₁ + ... + βₙxₙ
2. Sigmoid Function (Probability):  
   σ(z) = 1 / (1 + e^(-z))
3. Prediction:  
   ŷ = {  
   1, if σ(z) ≥ 0.5  
   0, if σ(z) < 0.5  
   }
4. Cost Function:  
   J(β) = -(1/m) Σ [yᵢ log(ŷᵢ) + (1 - yᵢ) log(1 - ŷᵢ)]
5. Gradient Descent (Parameter Update):  
   βⱼ = βⱼ - α ∂J / ∂βⱼ

II) Sequential Neural Network :

A fully connected, multi-layer sequential neural network was constructed with input, hidden, and output layers. While less powerful than CNNs for image classification, this architecture allows comparison of deep learning models without convolutions. Layers were optimized using the ReLU activation function and trained with Cross Entropy Loss.

III)Convolutional Neural Networks (ResNet34 and VGG16):

ResNet34: A ResNet34 model with pre-trained weights was fine-tuned for the EuroSAT dataset. ResNet's skip connections allow efficient training by addressing vanishing gradient problems, making it a popular choice for image classification tasks.

VGG16: In this work, pre-trained VGG16 has been utilized as the base feature-extracting model; then some fully connected layers have been appended to the base model to tailor the architecture to the specifics of the classification task.

Model Summary:

|  |  |  |
| --- | --- | --- |
| Layer Type | Output Shape | Parameters |
| Input | (256,256,3) | - |
| VGG16 Convolutional layers | (8,8,512) | 14,714,688 |
| Flatten | (32768) | 0 |
| Dense(512 units) | (512) | 16,777,288 |
| Dropout (0.3) | (512) | 0 |
| Dense(64 units) | (64) | 32,896 |
| Dropout(0.3) | (64) | 0 |
| Dense(10 units) | (10) | 650 |

This fine-tuning and data augmentation methods applied give the model the ability for classification and enhance it for further modifications on the image architecture to suit a specific task. Fine-tuning uses any pre-trained model to transfer learn some features learned to the new task. Data augmentation performs a transformation on the input images to increase the diversity of the dataset, thus reducing overfitting and enabling better generalization of the model.

FINE TUNING:

As base feature extractor, the VGG16 model pre-trained on ImageNet was taken. For fine-tuning the model towards the particular task, the first 15 layers of VGG16 were frozen to keep the learned features intact and treat the other layer as trainable. In this way, the model keeps the general features such as edges and textures, while, at the same time, fine-tunes the higher layers for the specific classification of categories in the domain. Fine-tuning was done by adding some fully connected layers to this base model, including one dense layer of 512 neurons, another of 64 neurons dense layer, and dropout layers (30 percent) to overcome overfitting. Subsequently, the model ended with a softmax output layer for multi-class classifications. Proposed models(Logistic Regression, Resnet, VGG16) are built and trained using the FastAI and Kerasframeworks. Models are evaluated using metrics such as accuracy, precision, recall, and F1-score.

**2.3**  **Software Details**

* **Programming Language:** Python, due to its extensive libraries for machine learning and deep learning.
* **Framework:** FastAI for implementing machine learning models and managing workflows efficiently.
* **Tools and Libraries:**

Jupyter Notebook: For interactive coding and visualization.

PyTorch: For building and training neural networks.

NumPy and Pandas: For data manipulation and preprocessing

Matplotlib and Seaborn: For visualizing results and data insights.

**CHAPTER 3**

**RESULTS AND DISCUSSIONS**

Each model demonstrated varying degrees of success, with CNN-based models (ResNet34 and VGG16) outperforming logistic regression and sequential neural networks. The key findings included:

* Logistic Regression achieved an accuracy of 65.4%, providing a baseline and illustrating the limitations of linear models in complex image classification tasks.
* Sequential Neural Network improved upon logistic regression with an accuracy of 78.2%, showing that deep learning models, even without convolutions, can capture some degree of complex spatial patterns.
* VGG16 achieved a comparable accuracy of 96.57%, demonstrating the effectiveness of deeper architectures in learning detailed features in land cover images.

Machine Learning Model:

Logistic regression:

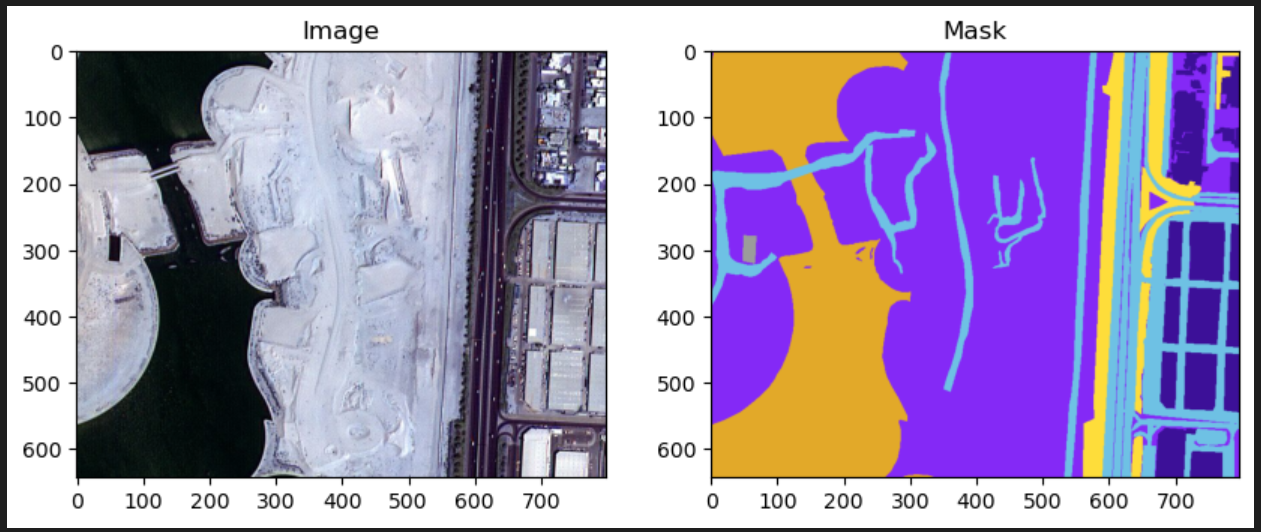


Figure 1. Input Image and Mask

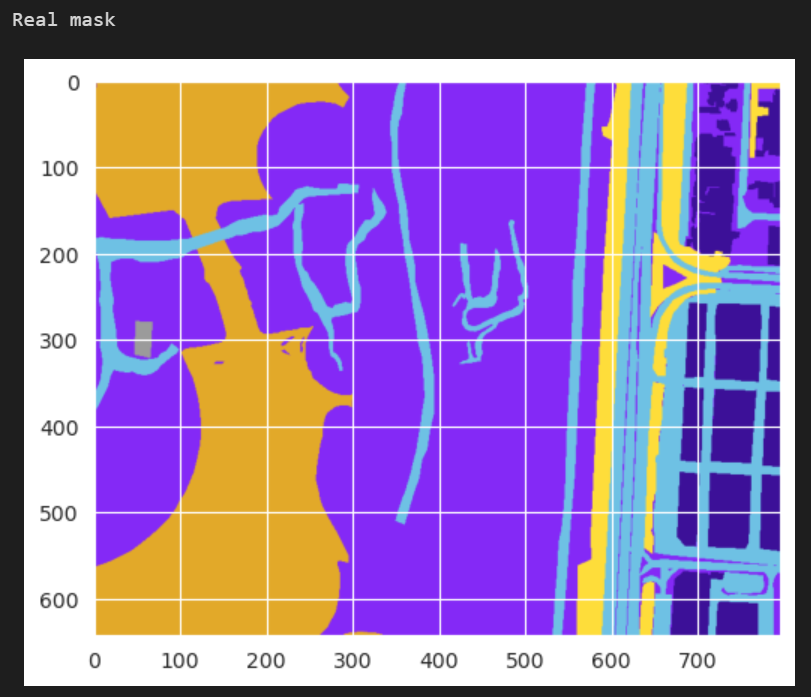
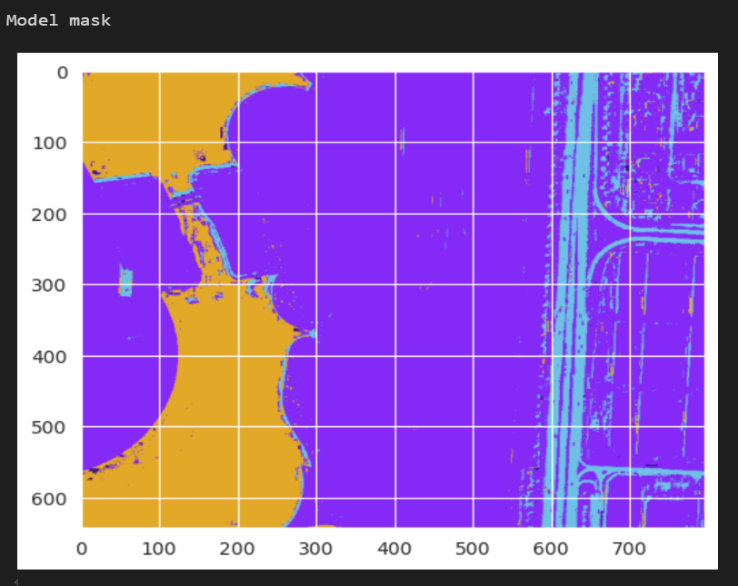


Figure 2. Comparing model and real mask

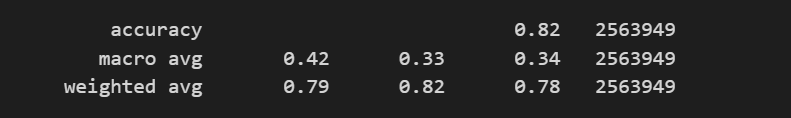


Figure 3. Accuracy of logistic regression

Deep Learning Models:

Sequential Model:



Figure 4.Data Image

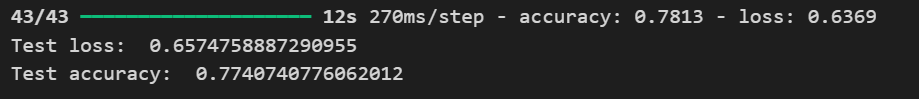


Figure 5.Accuracy of sequential model

Resnet(CNN):



Figure 6.Input Image

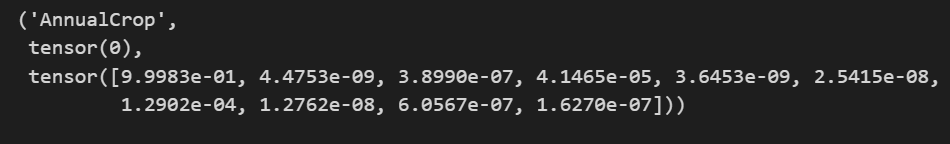


Figure 7.Prediction of the class

VGG16(CNN):

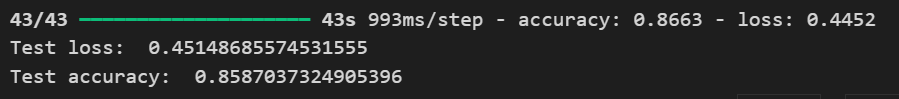


Figure 8. VGG 16 Accuracy

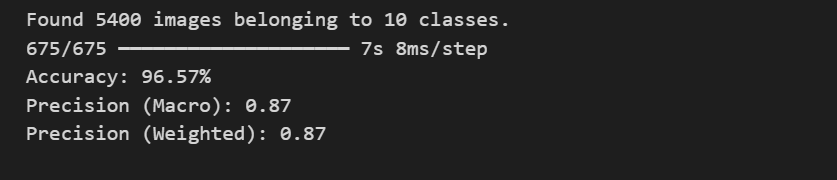


Figure 8. VGG 16 Accuracy with Fine tuning and Data Augmentation

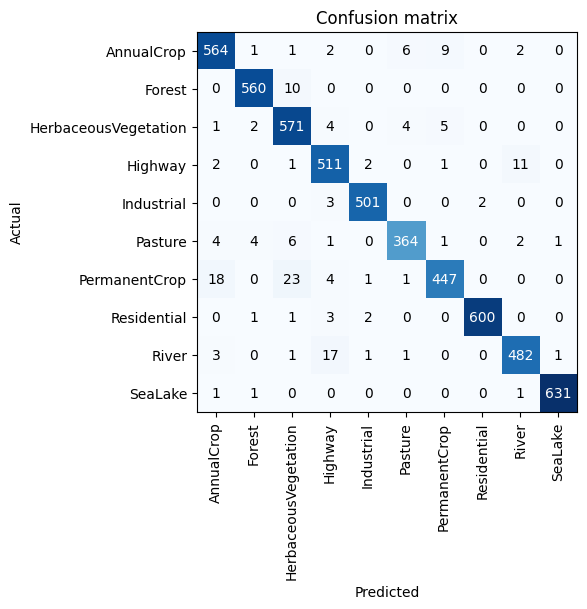


Figure 9.Confusion Matrix

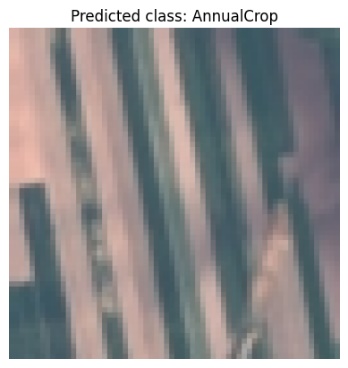
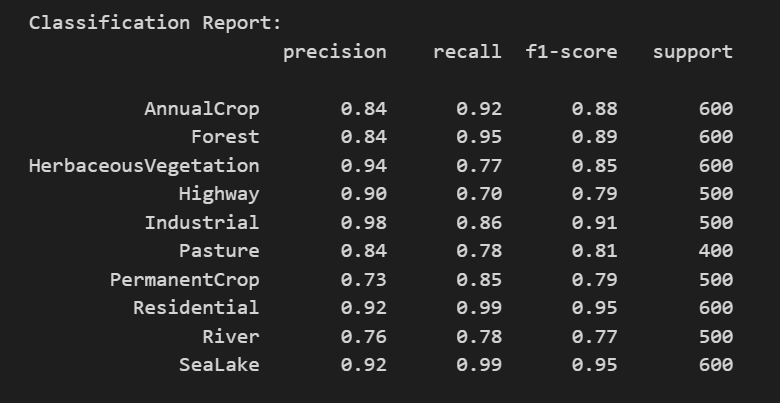


Figure 10. Predicted Classes



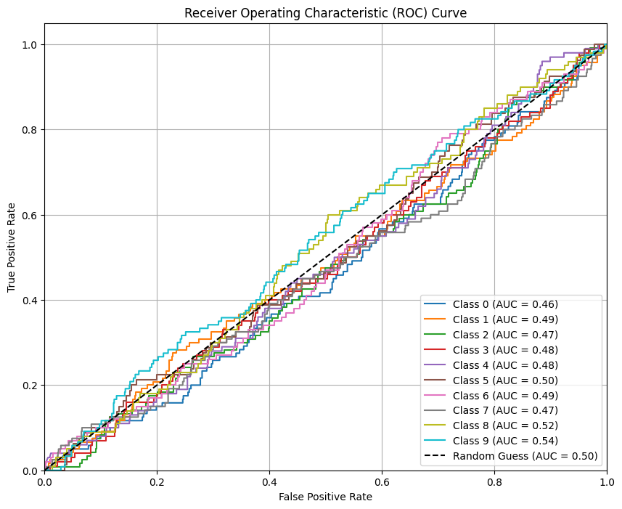
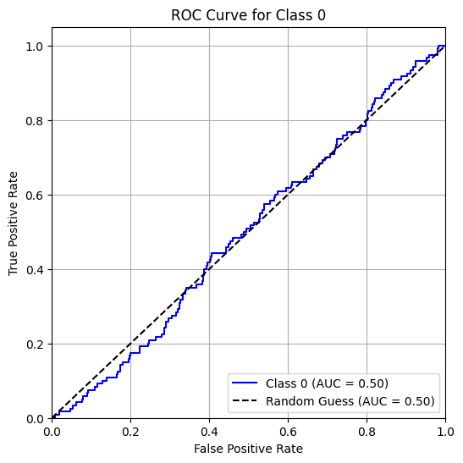
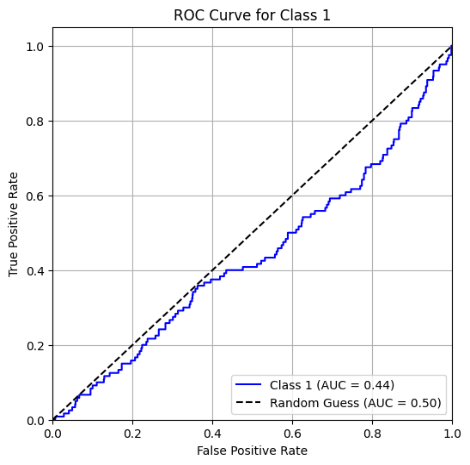
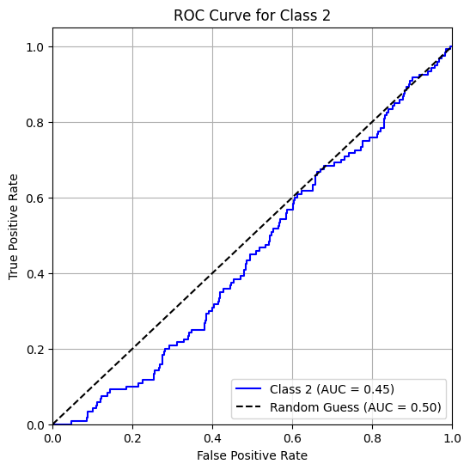
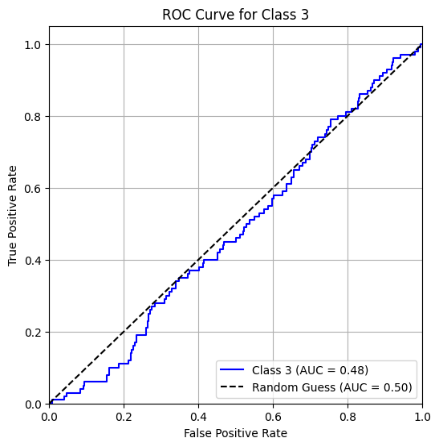
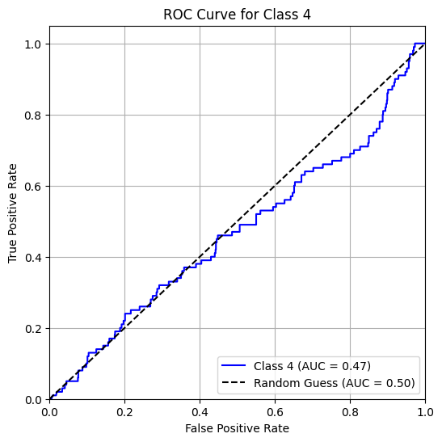
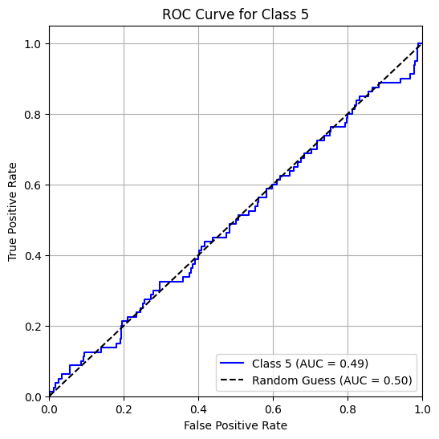
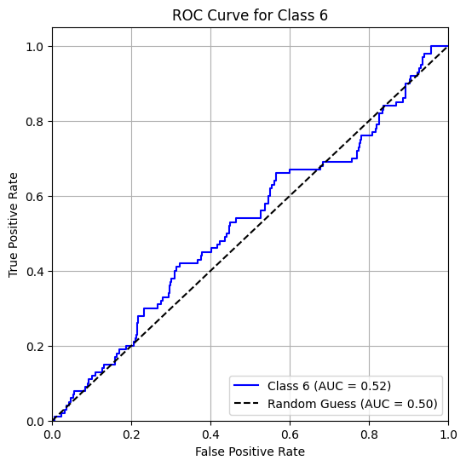
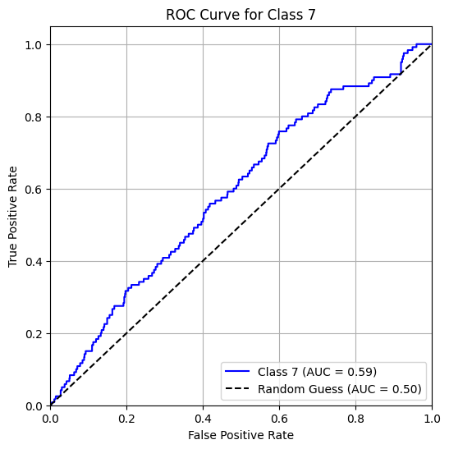
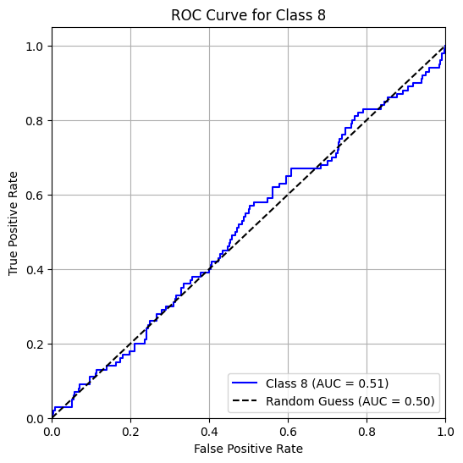


Figure 12.ROC Curv

ROC curves for each class:

Comparing Table:

|  |  |  |
| --- | --- | --- |
| **Model** | **VGG16 with 2 dense and 2 dropout layers** | **VGG16 with Data Augmentation and Fine Tuning** |
| **Test Accuracy** | 0.859259 | 0.965734 |
| **Test Loss** | 0.469799 | 0.298694 |

Final Comparison of all models :

|  |  |
| --- | --- |
| Model | Accuracy |
| Logistic regression | 82.0 |
| Sequential Model | 78.2 |
| VGG 16 | 85.92 |
| VGG 16 with Fine tuning and Data Augmentation | 96.57 |

**CHAPTER 4**

**CONCLUSION AND FUTURE WORK**

This particular project, namely Agriculture Land Use Classification, intends to improve accuracy and efficiency of land-use classification through the application of deep learning techniques and machine learning techniques. The applications of various models including logistic regression, sequential neural networks (SNNs), and advanced convolutional neural networks (CNNs) like ResNet34 and VGG16 explore the usage of the EuroSAT dataset in terms of multi-spectral satellite images. The FastAI framework was used to make these models capable of handling huge datasets and easily train deep learning models.

Results from this study have shown that CNNs, in particular ResNet34, outperform conventional models like logistic regression and SNNs in terms of classification accuracy. This is because CNNs are good in capturing spatial patterns present in satellite imagery, which play a huge part in knowing land-use categories such as agricultural land, forests, and the residential area. Data augmentation techniques such as rotation and flipping in place have proven to make the models more robust, thereby reducing overfitting and ensuring that the models generalized well across different types of images.

The key contributions of this project include:

1. **Accurate Land Use Classification**: By implementing deep learning, this project offers a data-driven reliable technique to classify agricultural land use from satellite imagery, aiding precision agriculture and optimal resource use.

2**. Model Comparison**: The comparison of logistic regression, SNNs, and CNNs underscores the merits of CNN-based models in highly complex image recognition tasks.

3. **Integration of Advanced Techniques:** The combination of data augmentation with deep learning models has proved vital in the management.

**Future Work**

While this project creates a fabulous base for the agricultural land use classification system, future work can improve the system in many directions and channels of application outside:

1 **Incorporating Additional Datasets:** in the present study the EuroSAT dataset is used, which although represents precious resource, relates to a very specific geographical area and land types. For example, enriching the model's ability to generalize into different land-use patterns and environments can be done through the future integration of other datasets from different regions and satellites, like Landsat or MODIS.

2. **Transfer Learning:** Because of the good performance of CNNs in image classification, transfer learning can also be a technique to improve the system's performance, especially for smaller or region-restricted datasets. Pre-trained models like those from ImageNet can be fine-tuned on satellite data to improve classification accuracy with less computational burden.

3**. Real-Time Monitoring of Lands:** Future work can be improved by developing such a system where real-time classification of land use can be performed using satellite and ground truth data.

4**. Multi-Task Learning:** Similar to land-use classification but predicting other variables, such as soil health, vegetation density, or crop yield, would enable new models to learn multiple tasks. That gives a more integrated agricultural management and environmental monitoring.

5. **incorporation with Geographic Information Systems (GIS):** Land use could be classified and seen in GIS so that large-scale spatial data may now also being analyzed with bountiful trails.

**CHAPTER 5**

**APPENDIX**

CODE:

1.Libraries imported:

#Import needed libraries

import os, shutil

from PIL import Image, ImageOps

#Standard Libraries

import numpy as np

import pandas as pd

# Visualizations

from matplotlib import pyplot as plt

import seaborn as sns

from sklearn.metrics import confusion\_matrix, multilabel\_confusion\_matrix, classification\_report, ConfusionMatrixDisplay

#TensorFlow

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator, array\_to\_img

from keras import models, layers, optimizers, regularizers

from tensorflow.keras import Model

from tensorflow.data.experimental import cardinality

from tensorflow.keras.utils import to\_categorical

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense # creates densely connected layer object

from tensorflow.keras.layers import Flatten # takes 2D input and turns into 1D array

from tensorflow.keras.layers import Conv2D # convolution layer

from tensorflow.keras.layers import MaxPooling2D # max pooling layer

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

#Transfer Learning

from keras.applications import ResNet50, VGG19, VGG16

%reload\_ext autoreload

%autoreload 2

%matplotlib inline

from fastai.vision import \* # import the vision module

from fastai.metrics import error\_rate # import our evaluation metric

import zipfile # import module to unzip the data

import urllib.request

import os # import module to access file paths

2.Path of dataset and extracting file with images:

zip\_path = '/content/EuroSAT.zip'

# Extract the zip file

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:

zip\_ref.extractall('/content/EuroSAT')

# Define paths for each category within the extracted '2750' folder

data\_AnnualCrop = '/content/EuroSAT/2750/AnnualCrop'

data\_Forest = '/content/EuroSAT/2750/Forest'

data\_HerbaceousVegetation = '/content/EuroSAT/2750/HerbaceousVegetation'

data\_Highway = '/content/EuroSAT/2750/Highway'

data\_Industrial = '/content/EuroSAT/2750/Industrial'

data\_Pasture = '/content/EuroSAT/2750/Pasture'

data\_PermanentCrop = '/content/EuroSAT/2750/PermanentCrop'

data\_Residential = '/content/EuroSAT/2750/Residential'

data\_River = '/content/EuroSAT/2750/River'

data\_SeaLake = '/content/EuroSAT/2750/SeaLake'

zip\_path = '/content/split\_dataset.zip'

# Extract only the 'split' folder from the zip file

with zipfile.ZipFile(zip\_path, 'r') as zip\_ref:

# List all files and directories in the zip

for file in zip\_ref.namelist():

if file.startswith('data/split/'): # Only extract files in the 'split' folder

zip\_ref.extract(file, '/content/')

# Define path to the extracted 'split' folder

new\_dir = '/content/data/split/'

3.Training folders:

#Set up the Train folder and subfolders

train\_folder = os.path.join(new\_dir, 'train')

train\_AnnualCrop = os.path.join(train\_folder, 'AnnualCrop')

train\_Forest = os.path.join(train\_folder, 'Forest')

train\_HerbaceousVegetation = os.path.join(train\_folder, 'HerbaceousVegetation')

train\_Highway = os.path.join(train\_folder, 'Highway')

train\_Industrial = os.path.join(train\_folder, 'Industrial')

train\_Pasture = os.path.join(train\_folder, 'Pasture')

train\_PermanentCrop = os.path.join(train\_folder, 'PermanentCrop')

train\_Residential = os.path.join(train\_folder, 'Residential')

train\_River = os.path.join(train\_folder, 'River')

train\_SeaLake = os.path.join(train\_folder, 'SeaLake')

4.)Testing folders:

#Set up the Test folder and subfolders

test\_folder = os.path.join(new\_dir, 'test')

test\_AnnualCrop = os.path.join(test\_folder, 'AnnualCrop')

test\_Forest = os.path.join(test\_folder, 'Forest')

test\_HerbaceousVegetation = os.path.join(test\_folder, 'HerbaceousVegetation')

test\_Highway = os.path.join(test\_folder, 'Highway')

test\_Industrial = os.path.join(test\_folder, 'Industrial')

test\_Pasture = os.path.join(test\_folder, 'Pasture')

test\_PermanentCrop = os.path.join(test\_folder, 'PermanentCrop')

test\_Residential = os.path.join(test\_folder, 'Residential')

test\_River = os.path.join(test\_folder, 'River')

test\_SeaLake = os.path.join(test\_folder, 'SeaLake')

5.)Normalization:

train\_folder = 'data/split/train'

test\_folder = 'data/split/test'

# Normalize images

train\_gen = ImageDataGenerator(rescale=1./255, validation\_split = 0.125)

test\_gen = ImageDataGenerator(rescale=1./255)

#Import data as 70% Train (10% Validation of orginal data set) and 20% Test

train\_generator = train\_gen.flow\_from\_directory(train\_folder,

class\_mode = 'categorical',

subset ='training',

batch\_size=128,

shuffle=True,

seed=42)

val\_generator= train\_gen.flow\_from\_directory(train\_folder,

class\_mode= 'categorical',

subset = "validation",

batch\_size=128,

shuffle=True,

seed=42)

test\_generator= test\_gen.flow\_from\_directory(test\_folder,

class\_mode= 'categorical',

batch\_size=128,

shuffle=False,

seed=42)

6.)Creating datasets:

# create the data sets

train\_images, train\_labels = next(train\_generator)

test\_images, test\_labels = next(test\_generator)

val\_images, val\_labels = next(val\_generator)

7.)Checking class Imbalance:

#Confirm class balance for train and test

train\_classes = train\_generator.classes

val\_classes = val\_generator.classes

test\_classes = test\_generator.classes

#Look at image distribution by class across train, test, and validation sets.

train\_class, train\_count = np.unique(train\_classes, return\_counts=True)

val\_class, val\_count = np.unique(val\_classes, return\_counts=True)

test\_class, test\_count = np.unique(test\_classes, return\_counts=True)

print('Train ~ {}'.format(list(zip(train\_class, train\_count))))

print('Validation ~ {}'.format(list(zip(val\_class, val\_count))))

print('Test ~ {}'.format(list(zip(test\_class, test\_count))))

8.)Previewing shape of images:

# Preview the shape of both the images and labels for both the train, validation, and test sets (8 objects total)

print("Train")

print(np.shape(train\_images))

print(np.shape(train\_labels))

print("Validation")

print(np.shape(val\_images))

print(np.shape(val\_labels))

print("Test")

print(np.shape(test\_images))

print(np.shape(test\_labels))

9.) Reshaping taken input:

#Reshape our input

vgg16 = VGG16(weights='imagenet',

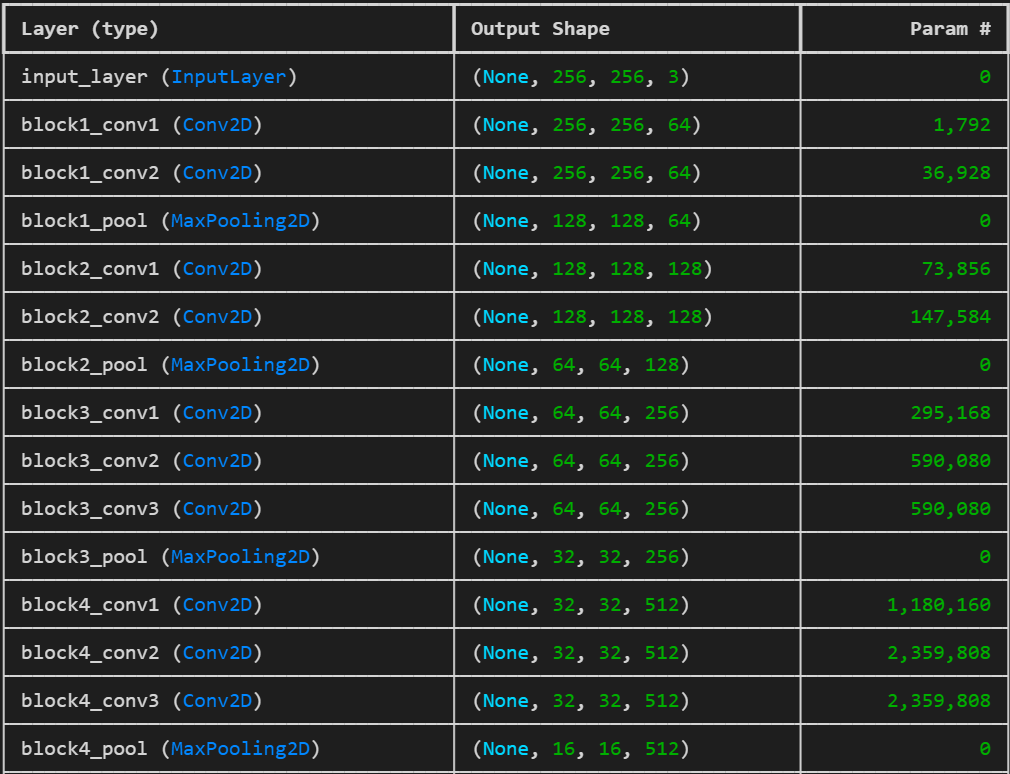
include\_top=False,

input\_shape=(256, 256, 3))

10.)Loading VGG16 model :

#Check to ensure there is no prediction layer

vgg16.summary()



11.)Fitting the model:

# Define Stopping Criteria

valcallback = [EarlyStopping(monitor='val\_accuracy', mode='max', verbose = 1, patience = 5),]

# Fit the model

vgg16\_model\_history = vgg16\_model.fit(vgg16\_train\_generator,

epochs= 50,

validation\_data = vgg16\_val\_generator,

callbacks= valcallback,

batch\_size=128,

verbose = 1)

12.)Calculating loss and accuracy:

# Check loss and accuracy on test data

test\_loss, test\_acc = vgg16\_model.evaluate(vgg16\_test\_generator, verbose = 1)

print('Test loss: ', test\_loss)

print('Test accuracy: ', test\_acc)

13.)Fine tuning to increase accuracy:

# Fine-tune from layer 15 onwards

fine\_tune\_at = 15

# Freeze all the layers before the `fine\_tune\_at` layer

for layer in vgg16.layers[:fine\_tune\_at]:

layer.trainable = False

#Sanity check that VGG19 Layer is frozen

for layer in aug\_vgg16\_model.layers:

print(layer.name, layer.trainable)

14.)Run the model after fine tuning and data augumentation:

# Clear previous models from memory

clear\_session()

# Enable mixed precision training to optimize memory usage

policy = Policy('mixed\_float16')

set\_global\_policy(policy)

# Initialize ImageDataGenerator with smaller image sizes and reduced batch size

datagen = ImageDataGenerator(rescale=1./255, validation\_split=0.2)

train\_generator = datagen.flow\_from\_directory(

'data/split/train/',

target\_size=(150, 150), # Optimized image size

batch\_size=8, # Reduced batch size

class\_mode='categorical',

subset='training'

)

val\_generator = datagen.flow\_from\_directory(

'data/split/test/',

target\_size=(150, 150),

batch\_size=8,

class\_mode='categorical',

subset='validation'

)

# Load pre-trained VGG16 model without the top layers

base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(150, 150, 3))

# Freeze the base model layers to save memory

for layer in base\_model.layers:

layer.trainable = False

# Add custom layers for classification

x = Flatten()(base\_model.output)

x = Dense(128, activation='relu')(x)

x = Dense(train\_generator.num\_classes, activation='softmax', dtype='float32')(x) # Output is explicitly set to float32

# Create the full model

model = Model(inputs=base\_model.input, outputs=x)

# Compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Train the model

history = model.fit(

train\_generator,

epochs=10,

validation\_data=val\_generator,

steps\_per\_epoch=train\_generator.samples // train\_generator.batch\_size,

validation\_steps=val\_generator.samples // val\_generator.batch\_size

)

# Save the model

model.save('/content/vgg16\_optimized\_model.h5')

15.)Evaluation metrics:

# Get true labels

true\_labels = test\_generator.classes

class\_labels = list(test\_generator.class\_indices.keys()) # Get class labels

# Predict on test data

predictions = model.predict(test\_generator)

predicted\_classes = np.argmax(predictions, axis=1)

# Accuracy

accuracy = accuracy\_score(true\_labels, predicted\_classes)

print(f"Accuracy: {accuracy \* 100:.2f}%")

# Precision (macro and weighted)

precision\_macro = precision\_score(true\_labels, predicted\_classes, average='macro')

precision\_weighted = precision\_score(true\_labels, predicted\_classes, average='weighted')

print(f"Precision (Macro): {precision\_macro:.2f}")

print(f"Precision (Weighted): {precision\_weighted:.2f}")

# Detailed classification report

print("\nClassification Report:")

print(classification\_report(true\_labels, predicted\_classes, target\_names=class\_labels))

16.)Plots:

# One-hot encode the true labels if not already one-hot encoded

y\_true = to\_categorical(y\_true, num\_classes=num\_classes)

# Calculate ROC for each class

fpr = {}

tpr = {}

roc\_auc = {}

for i in range(num\_classes):

fpr[i], tpr[i], \_ = roc\_curve(y\_true[:, i], y\_pred[:, i])

roc\_auc[i] = auc(fpr[i], tpr[i])

# Plot the ROC curves

plt.figure(figsize=(10, 8))

for i in range(num\_classes):

plt.plot(fpr[i], tpr[i], label=f"Class {i} (AUC = {roc\_auc[i]:.2f})")

plt.plot([0, 1], [0, 1], 'k--', label='Random Guess (AUC = 0.50)')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("Receiver Operating Characteristic (ROC) Curve")

plt.legend(loc="lower right")

plt.grid()

plt.show()

# Calculate micro-average AUC (useful for multi-class classification)

y\_true\_flat = y\_true.ravel()

y\_pred\_flat = y\_pred.ravel()

micro\_auc = auc(\*roc\_curve(y\_true\_flat, y\_pred\_flat)[:2])

print(f"Micro-Average AUC: {micro\_auc:.2f}")

17.)Testing the model and predicting classes for images .

# Load the image you want to classify

img\_path = '/content/AnnualCrop\_176.jpg' # Replace with your image path

img = image.load\_img(img\_path, target\_size=(256, 256)) # Resize to the input

# Convert the image to a numpy array

img\_array = image.img\_to\_array(img)

img\_array = np.expand\_dims(img\_array, axis=0) # Add batch dimension

# Preprocess the image

img\_array = tf.keras.applications.vgg16.preprocess\_input(img\_array) # VGG16# Predict the class probabilities

predictions = vgg16\_model.predict(img\_array)

# Get the predicted class index

predicted\_class\_index = np.argmax(predictions, axis=1)[0]

# Load class labels from the generator (assuming the train generator is available)

class\_labels = vgg16\_train\_generator.class\_indices

class\_labels = {v: k for k, v in class\_labels.items()} # Reverse the dictionary to get class names

# Get the predicted class label

predicted\_class\_label = class\_labels[predicted\_class\_index]

plt.imshow(img) # Display the image

plt.title(f"Predicted class: {predicted\_class\_label}") # Title with predicted class

plt.axis('off') # Hide axis

plt.show()

# Output the prediction

print(f"Predicted class: {predicted\_class\_label}")

**REFERENCES**

[1] M. Helber, B. Bischke, A. Dengel, and D. Borth, "EuroSAT: A novel dataset and deep learning benchmark for land use and land cover classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 7, pp. 2217–2226, Jul. 2019. *(references)*

[2] J. Chen, X. Y. Chen, and J. W. Gong, "Land-use/land-cover change detection: Techniques and applications," *International Journal of Remote Sensing*, vol. 23, no. 4, pp. 1113–1126, Feb. 2002.

[3] P. Defourny, P. Vancutsem, E. C. Bontemps, G. L. Schouten, and L. R. Giri, "Remote sensing of land use and land cover at global scale," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 7, pp. 233–238, 2014.

[4] T. J. Maxwell, J. R. Silva, and S. J. Walsh, "A comparison of object-based image analysis and machine learning algorithms for land use/land cover classification," *Remote Sensing*, vol. 7, no. 3, pp. 2566–2588, Mar. 2015