Image Recognition System Using Teachable Machine and TensorFlow

Artificial Intelligence

[Task 1]

Mohamad Abdulmoula,R. Rama

ramamohameed9@gmail.com

Department of Software Engineer

Under the Supervision Of:

Eng Asmaa Duramae

July 22, 2025

Table of Contents

Glossary of Terms

Term	**Definition**
AI (Artificial Intelligence)	The simulation of human intelligence in machines programmed to think and learn.
Teachable Machine	A Google-powered web-based tool for training simple machine learning models using images, audio, or poses.
TensorFlow	An open-source machine learning framework developed by Google, widely used for deep learning models.
Keras Model	A high-level API built on TensorFlow for building and training deep learning models.
Image Classification	The task of assigning a label (“Bird” or “Drone”) to an input image based on its features.
OpenCV	An open-source computer vision library used for image processing and annotation.
Jira	A software tool used for issue tracking and project management.
Xray	A test management plugin for Jira used to create, manage, and report on test cases and executions.
Manual Testing	A software testing process where test cases are executed manually without automation tools.
Inference	The process of using a trained model to make predictions on new, unseen data.

Project Overview

This project presents the development of a lightweight image classification system that can distinguish between two visual classes: Bird and Drone. The model was trained using Google Teachable Machine and deployed via TensorFlow. The final output is both visual and textual displaying the classification result on the image and printing the predicted class in the terminal.

I. Planning

A. **Objective:** To design a deployable image classifier using a pre-trained deep learning model.
B. **Purpose:** Automate the identification of drones and birds in static images for surveillance, research, or educational use.
C. **Tools Used:** Teachable Machine, TensorFlow, Python, Jira, Xray.

a) Classify a given image into either "Bird" or "Drone".
b) Display the classification result on the image.
c) Print the result to the console.

A. Non-Functional Requirements:

a) Ensure compatibility with local machines using minimal dependencies.
b) Output must be clear and visually identifiable.
c) Fast prediction time and low memory consumption.

II. Design

A. UML Activity Diagram – Inference Pipeline

To illustrate the operational flow of the image classification system during prediction, the following UML Activity Diagram outlines the inference pipeline executed by the Python script:

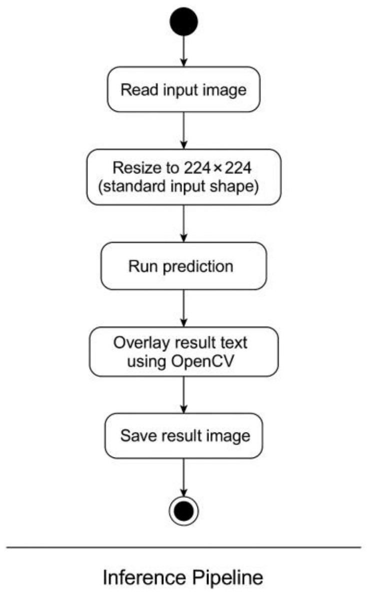

Read input image
Resize to 224×224 (standard input shape)
Run prediction
Overlay result text using OpenCV
Save result image
Inference Pipeline

Figure 1 UML Activity Diagram representing the sequence of actions in the inference process for image classification.

B. Description of Workflow:

a) **Read input image**: The system loads an external image (test.jpg) for classification.

b) **Resize image**: The image is resized to 224×224 pixels, which matches the expected input size of the model.

c) **Normalize pixel values**: The image array is normalized (scaled between 0 and 1) for compatibility with the model.

d) **Run prediction**: The TensorFlow Keras model is used to predict the image class based on its feature representation.

e) **Overlay result**: OpenCV is used to draw the predicted class name on the image.

f) **Save output**: The resulting annotated image is saved as output.jpg.

III. Coding

A. Model Training:

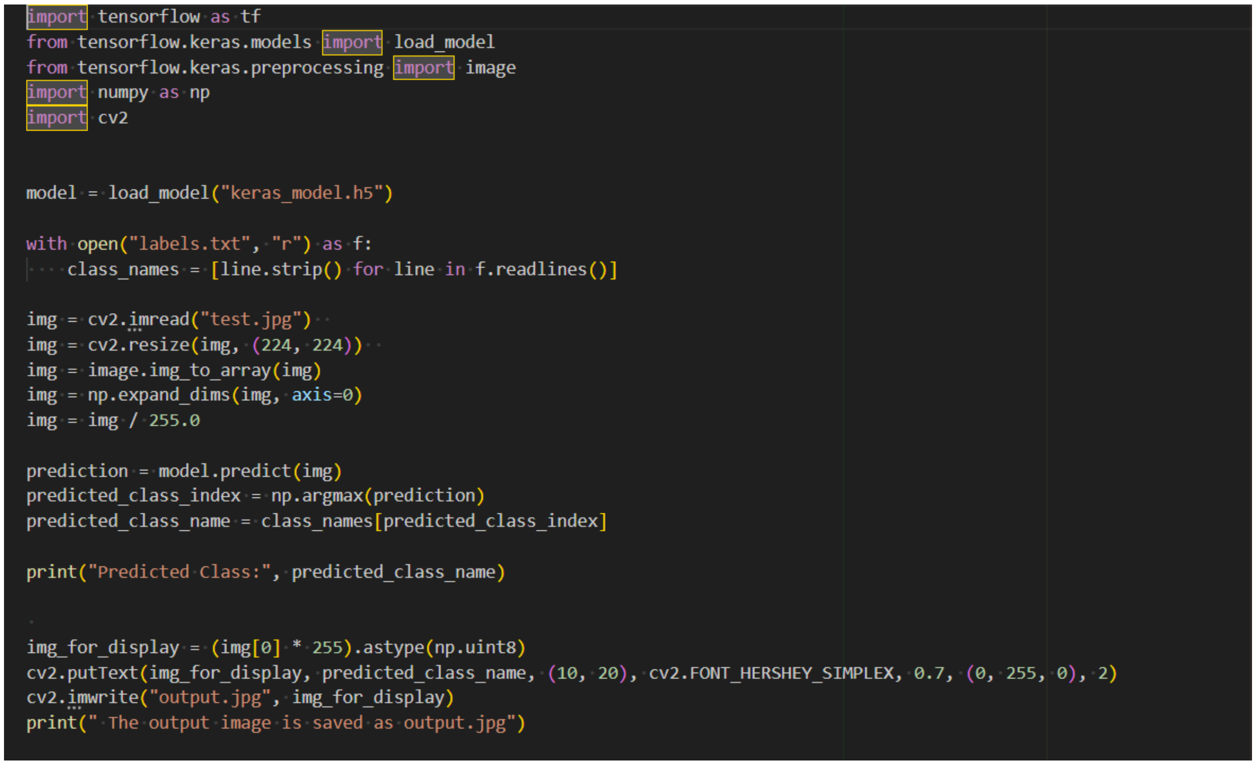
a) Two image classes were trained: Bird and Drone.

b) The model was exported from Teachable Machine as keras_model.h5 along with labels.txt.

B. Python Deployment:

a) Key libraries: TensorFlow, NumPy.

b) The model is loaded with load_model(), image resized and normalized, and prediction performed using model.predict().



```python
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
import cv2


model = load_model("keras_model.h5")

with open("labels.txt", "r") as f:
    class_names = [line.strip() for line in f.readlines()]

img = cv2.imread("test.jpg")
img = cv2.resize(img, (224, 224))
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img = img / 255.0

prediction = model.predict(img)
predicted_class_index = np.argmax(prediction)
predicted_class_name = class_names[predicted_class_index]

print("Predicted Class:", predicted_class_name)


img_for_display = (img[0] * 255).astype(np.uint8)
cv2.putText(img_for_display, predicted_class_name, (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
cv2.imwrite("output.jpg", img_for_display)
print(" The output image is saved as output.jpg")
```

Figure 2 Python Script for Image Classification Using TensorFlow

This script loads a pre-trained Keras model (keras_model.h5) and associated class labels (labels.txt) to classify an input image (test.jpg). The image is resized, normalized, and passed through the model. The predicted class is printed to the terminal and annotated on the image which is then saved as output.jpg.

IV. Testing

A. Testing Objective

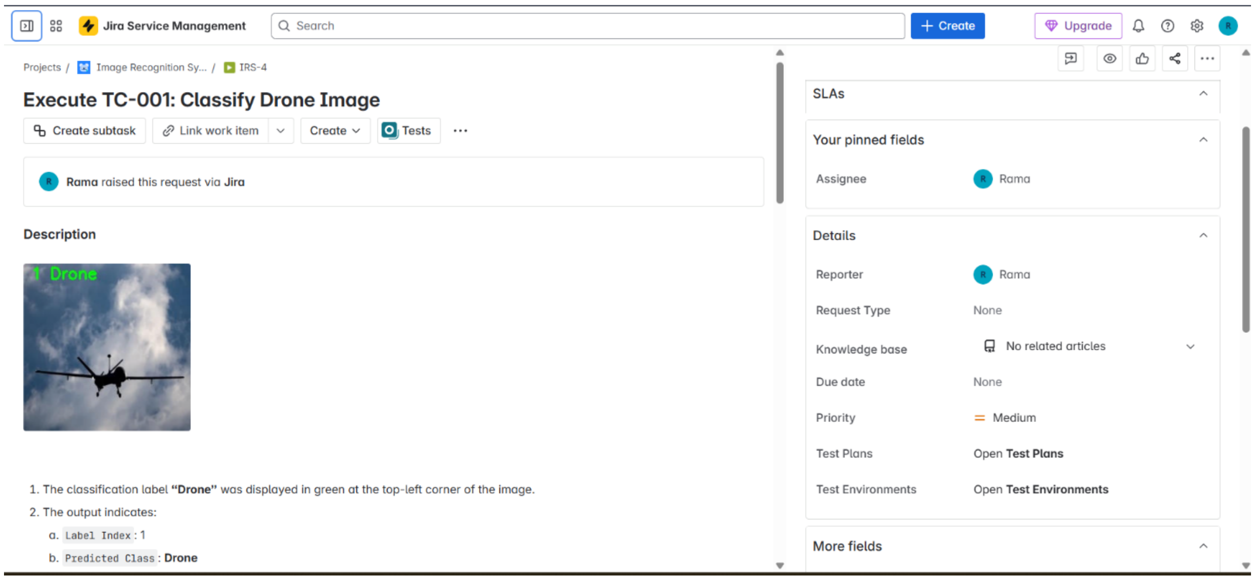
The goal of testing was to verify that the deployed image classification system accurately distinguishes between bird and drone images, and that it properly displays and logs the results as expected.

B. Testing Approach

We adopted a **manual functional testing** strategy using **Xray Test Management integrated with Jira** to manage, execute, and trace test cases and their outcomes. Each test case focused on verifying a specific functionality or output behavior of the model.

C. Test Case Summary

Test Case ID	Summary	Type	Expected Result	Status
TC-001	Classify Drone Image	Manual	Image labeled "Drone"	Passed
TC-002	Verify Classification Accuracy	Manual	Output matches ground truth	Passed



Projects / Image Recognition Sy... / IRS-4

Execute TC-001: Classify Drone Image

Rama raised this request via Jira

Description

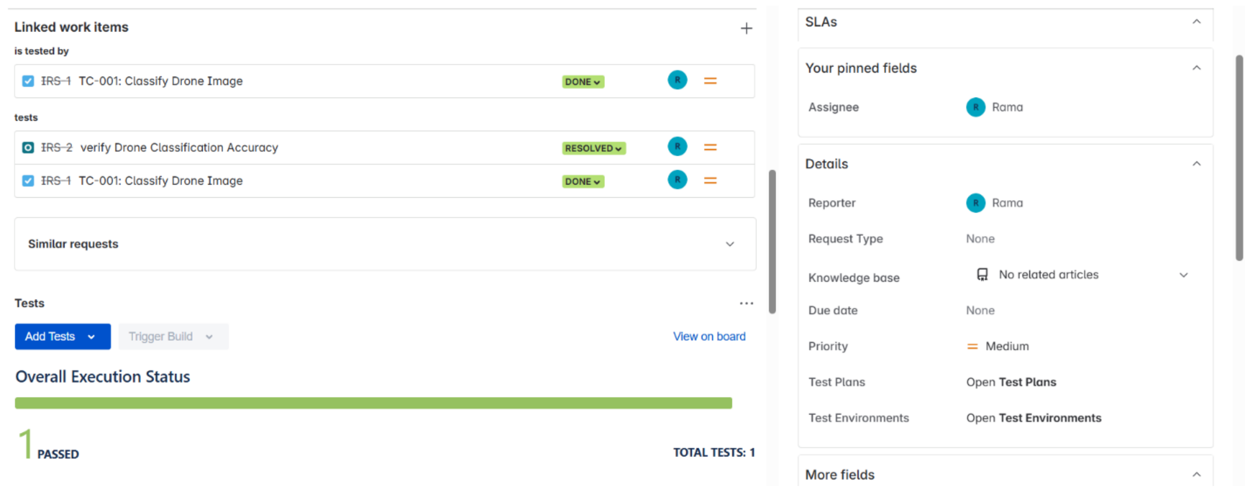
1. The classification label **"Drone"** was displayed in green at the top-left corner of the image.
2. The output indicates:
   a. `Label Index` : 1
   b. `Predicted Class` : **Drone**

Assignee	Rama

Details

Reporter	Rama
Request Type	None
Knowledge base	No related articles
Due date	None
Priority	Medium
Test Plans	Open **Test Plans**
Test Environments	Open **Test Environments**

Figure 3 Test Execution for TC-001 – Classify Drone Image in Jira.



Linked work items

is tested by

~~IRS-1~~ TC-001: Classify Drone Image	DONE	R	=

tests

~~IRS-2~~ verify Drone Classification Accuracy	RESOLVED	R	=
~~IRS-1~~ TC-001: Classify Drone Image	DONE	R	=

Similar requests

Tests

Add Tests | Trigger Build | View on board

Overall Execution Status

1 PASSED TOTAL TESTS: 1

SLAs

Your pinned fields

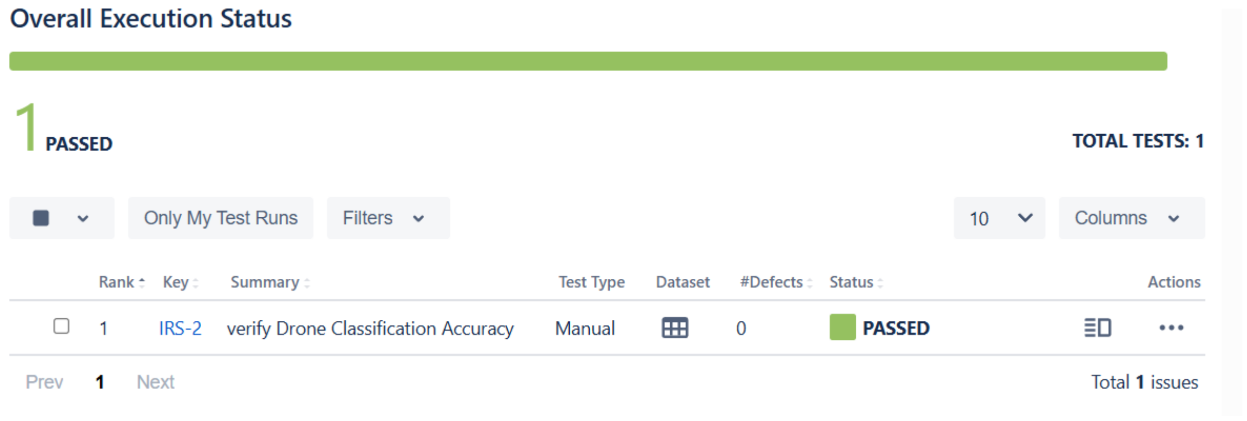
Assignee	Rama

Details

Reporter	Rama
Request Type	None
Knowledge base	No related articles
Due date	None
Priority	Medium
Test Plans	Open Test Plans
Test Environments	Open Test Environments

More fields

Figure 4 Overall Test Execution Summary in Xray for Drone Classification.



Overall Execution Status

1 PASSED TOTAL TESTS: 1

Only My Test Runs | Filters | 10 | Columns

	Rank	Key	Summary	Test Type	Dataset	#Defects	Status	Actions
☐	1	IRS-2	verify Drone Classification Accuracy	Manual		0	🟩 PASSED	

Prev 1 Next Total 1 issues

Figure 5 Manual Test Execution Result for Drone Classification Accuracy.

The following figures summarize the test execution results of the image classification system. Figure 3 shows the execution details of test case TC-001, which verifies the classification of a drone image. Figure 4 provides an overview of the execution status, indicating that all linked tests have successfully passed. Figure 5 presents a detailed result of test IRS-2, confirming the accurate classification of the drone image with a "Passed" status and no reported defects.

V. Conclusion

This project successfully demonstrates the implementation of a lightweight image classification system capable of distinguishing between drones and birds. Through the integration of Teachable Machine for model training, TensorFlow for deployment, and Xray for structured test management within Jira, the system achieved its objectives in terms of accuracy, speed, and simplicity. The visual output clearly displayed classification labels on the input images, while textual logs ensured transparency and traceability. Testing outcomes confirmed that the classifier performed as expected, passing all defined test cases with zero defects. This system offers a solid foundation for further enhancement or integration into larger surveillance, research, or educational platforms.

References

A. TensorFlow – Open Source Machine Learning Framework by Google.
Available at: https://www.tensorflow.org

B. Teachable Machine – A No-Code Tool to Train Machine Learning Models.
Available at: https://teachablemachine.withgoogle.com

C. OpenCV (Open Source Computer Vision Library) – Library for Real-Time Computer Vision.
Available at: https://opencv.org

D. Keras Documentation – Deep Learning for Humans.
Available at: https://keras.io

E. Jira Software by Atlassian – Project and Issue Tracking.
Available at: https://www.atlassian.com/software/jira

F. Xray Test Management for Jira – Test Management and Quality Assurance Tool.
Available at: https://www.getxray.app

G. Brownlee, J. (2019). Deep Learning for Computer Vision. Machine Learning Mastery.
https://machinelearningmastery.com/deep-learning-for-computer-vision