**Machine Learning Project Documentation**

**Deployment**

**1. Overview**

The deployment phase ensures that the trained machine learning model is accessible and usable in a real-world environment. The model is integrated into a user-friendly interface using **Tkinter**, allowing users to input images and receive predictions on waste classification. The deployment process involves model serialization, serving, API integration, security considerations, and monitoring.

**2. Model Serialization**

To deploy the trained CNN model, it is serialized using **TensorFlow’s HDF5 (.h5) format** for efficient storage and portability. The process includes:

* **Saving the model:**
* import tensorflow as tf
* model.save("waste\_classifier\_model.h5")
* **Loading the model for deployment:**
* loaded\_model = tf.keras.models.load\_model("waste\_classifier\_model.h5")

Serialization ensures that the model remains lightweight and can be loaded efficiently for predictions.

**3. Model Serving**

The trained model is integrated into a **Tkinter-based graphical user interface (GUI)**, allowing users to load an image and classify it using the model.

* **Steps for serving:**
  1. Load the serialized model.
  2. Preprocess the user-input image.
  3. Pass the image to the model for prediction.
  4. Display the predicted class on the interface.
* **Tkinter GUI Code Example:**

import tkinter as tk

from tkinter import filedialog

from PIL import Image, ImageTk

import numpy as np

import tensorflow as tf

# Load model

model = tf.keras.models.load\_model("waste\_classifier\_model.h5")

def classify\_image():

file\_path = filedialog.askopenfilename()

image = Image.open(file\_path).resize((128, 128))

image\_array = np.array(image) / 255.0

image\_array = image\_array.reshape((1, 128, 128, 3))

prediction = model.predict(image\_array)

result\_label.config(text=f"Predicted Class: {np.argmax(prediction)}")

root = tk.Tk()

root.title("Waste Classification")

btn = tk.Button(root, text="Select Image", command=classify\_image)

btn.pack()

result\_label = tk.Label(root, text="Prediction will appear here")

result\_label.pack()

root.mainloop()

**4. Security Considerations**

Security measures implemented during deployment include:

* **Input Validation:** Ensuring that only valid image files are processed.
* **Restricted Model Access:** The model is loaded locally, reducing exposure to unauthorized users.
* **Error Handling:** Using try-except blocks in the Tkinter app to prevent crashes.

**6. Monitoring and Logging**

To ensure the deployed model is performing well:

* **Logging Predictions:** Store input images and corresponding predictions for review.
* **Tracking Performance:** Log model confidence scores to identify misclassifications.
* **Updating the Model:** Periodically retrain the model with new data to maintain accuracy.

**Conclusion**

The deployment of the waste classification model using Tkinter ensures an interactive and user-friendly experience. While the current implementation is focused on local execution, future work may involve deploying the model as a cloud-based API for remote accessibility.