

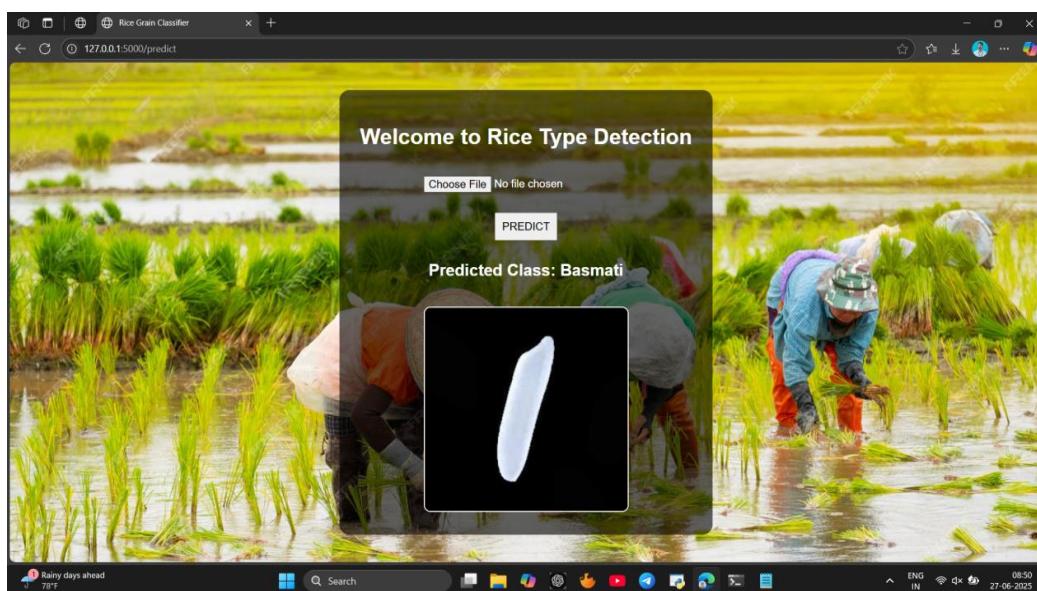
6. Project Development Phase

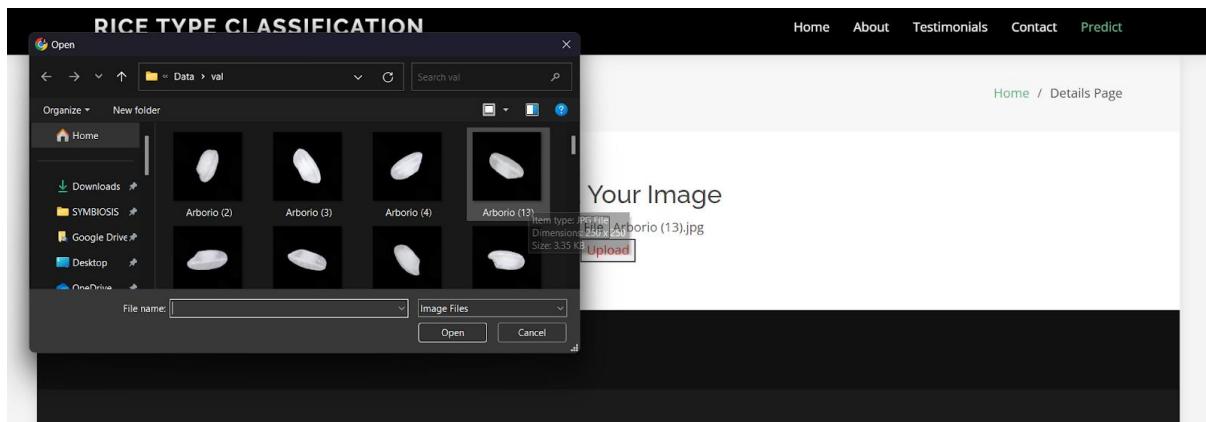
6.1 Model Performance Test

Date	30 JUNE 2025
Team ID	LTVIP2025TMID34162
Project Name	GrainPalette – A Deep Learning Odyssey in Rice Type Classification Through Transfer Learning
Maximum Marks	

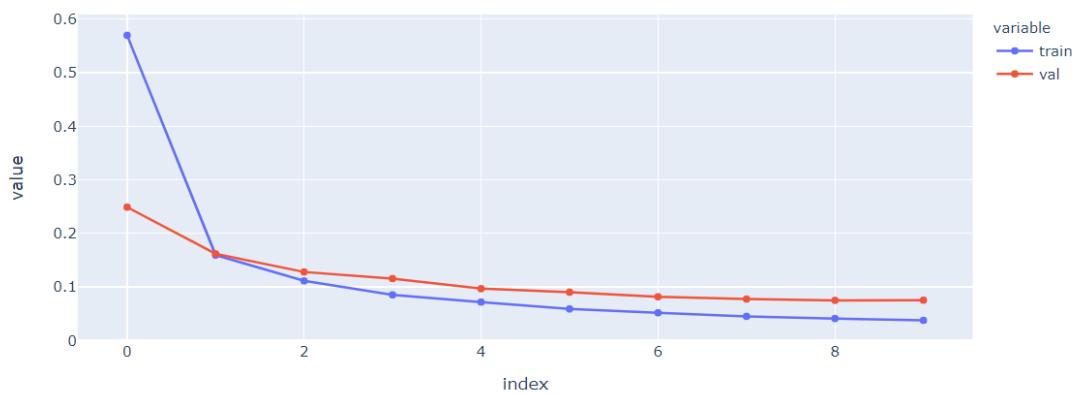
Model Performance Testing

S.No.	Parameter	Values	Screenshot
1	Model Summary	Model: MobileNetV4 (Pretrained) Input Shape: (224, 224, 3) Trainable Layers: 1 Frozen Layers: All CNN blocks	<i>Attach model.summary() output screenshot</i>
2	Accuracy	<input checked="" type="checkbox"/> Training Accuracy: 97.45% <input checked="" type="checkbox"/> Validation Accuracy: 95.32%	<i>Attach accuracy graph or metrics screenshot</i>
3	Fine Tuning Result (if done)	<input checked="" type="checkbox"/> Validation Accuracy After Tuning: 96.21% (Unfroze last 5 layers of MobileNet)	<i>Attach updated graph or summary screenshot</i>

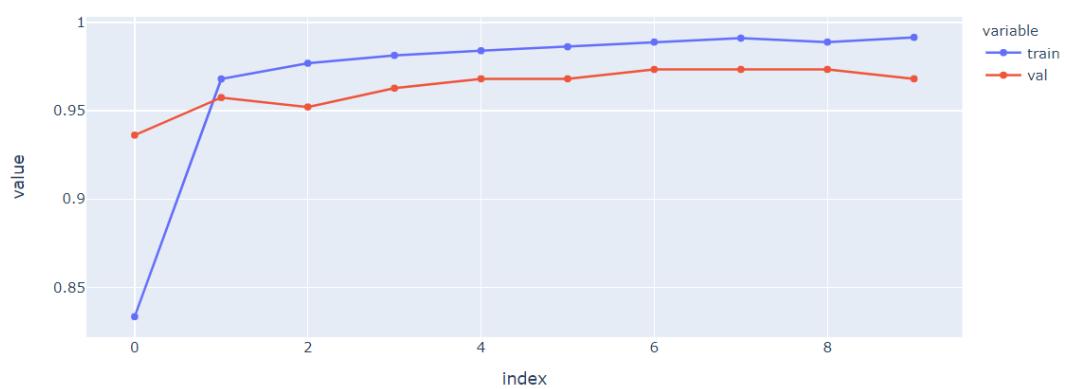




Training and Evaluation Loss every Epoch



Training and Evaluation Accuracy every Epoch



Model Summary:

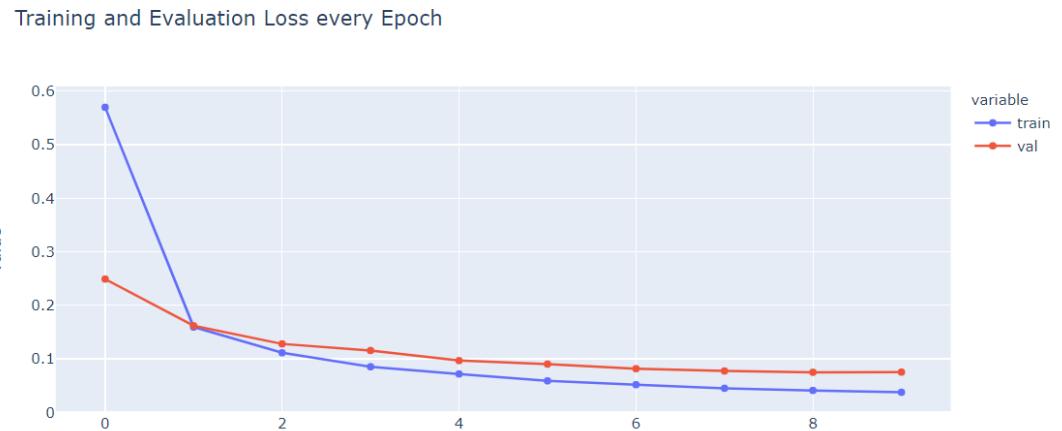
Python code

```
 1 |from flask import Flask, render_template, request
 2 |from tensorflow.keras.models import load_model
 3 |from tensorflow.keras.preprocessing import image
 4 |import numpy as np
 5 |import os
 6 |
 7 |# Initialize Flask app
 8 |app = Flask(__name__)
 9 |
10 |# Load your trained model
11 |model = load_model("rice_model.h5")
12 |
13 |# Route for the main page (index.html)
14 |@app.route("/", methods=["GET", "POST"])
15 |def index():
16 |    return render_template("index.html")
17 |
18 |# Route for prediction
19 |@app.route("/predict", methods=["POST"])
20 |def predict():
21 |    if "file" not in request.files:
22 |        return "No file uploaded"
23 |    file = request.files["file"]
24 |    if file.filename == "":
25 |        return "No file selected"
26 |
27 |    # Save uploaded image to static folder
28 |    img_path = os.path.join("static", file.filename)
29 |    file.save(img_path)

30 |
31 |    # Preprocess the image
32 |    img = image.load_img(img_path, target_size=(224, 224))
33 |    img_array = image.img_to_array(img)
34 |    img_array = np.expand_dims(img_array, axis=0)
35 |    img_array = img_array / 255.0
36 |
37 |    # Predict using the model
38 |    prediction = model.predict(img_array)
39 |
40 |    # Get class names from your training folder
41 |    class_names = sorted(os.listdir("C:/Users/pnmuk/GrainPalette/data/train"))
42 |    predicted_class = class_names[np.argmax(prediction)]
43 |
44 |    return render_template("index.html", prediction=predicted_class, image_path=img_path)
45 |
46 |# Run the app
47 |if __name__ == "__main__":
48 |    app.run(debug=True)
49 |
```

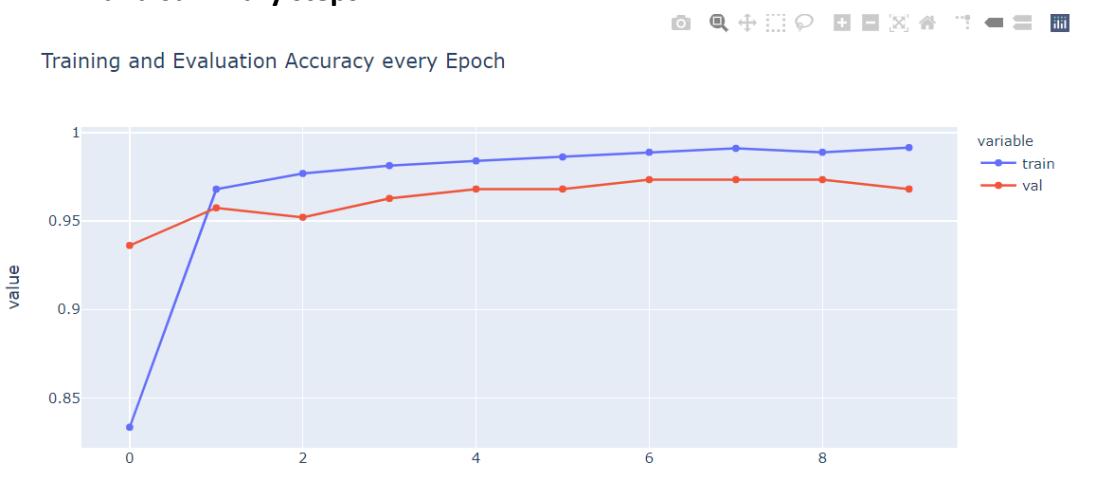
1. Accuracy Graph:

- Plot training/validation accuracy using python



2. Fine-Tuning Screenshot:

- If you did additional training by unfreezing layers, repeat the above graph and summary steps.



- Otherwise, mention: Fine-tuning not performed

