

## Project Development Phase Model Performance Test

Date	27 June 2025
Team ID	LTVIP2025TMID41438
Project Name	GrainPalette - A Deep Learning Odyssey In Rice Type Classification Through Transfer Learning
Maximum Marks	10 Marks

### Model Performance Testing:

Model performance testing :

S.No.	Parameter	Values	Screenshot
1.	Metrics	<p><b>Classification Model:</b></p> <p><b>Confusion Matrix:</b> [[111 0 0 3 2],[0 122 2 0 0],[0 0 109 0 0],[0 0 0 98 0],[1 0 0 0 114]]</p> <p><b>Accuracy Score:</b> 98.58%</p> <p><b>Classification Report:</b></p> <p>1.Arborio → Precision: 0.99,Recall: 0.96, F1: 0.97</p> <p>2.Basmati → Precision: 1.00,Recall: 0.98, F1: 0.99</p> <p>3.Ipsala → Precision: 1.00,Recall: 1.00, F1: 1.00</p> <p>4.Jasmine → Precision: 1.00,Recall:1.00, F1: 1.00</p> <p>5.Karacadagi → Precision:0.98,Recall: 0.99, F1: 0.99</p> <p><b>Overall Accuracy: 98.58%</b></p>	<pre> y_pred_probs = model.predict(x_test) y_pred = np.argmax(y_pred_probs, axis=1) # Calculate and print the Confusion Matrix print("\n--- Confusion Matrix ---") cm = confusion_matrix(y_test, y_pred) print(cm)  # Define target names for better readability in the classification report target_names = list(df_images.keys()) # ['arborio', 'basmati', 'ipsala', 'jasmine', 'karacadag'] # Calculate and print the Classification Report print("\n--- Classification Report ---") cr = classification_report(y_test, y_pred, target_names=target_names) print(cr) </pre> <p>18/18 [=====] - 9s 461ms/step</p> <pre> --- Confusion Matrix --- [[111  0  0  3  2]  [  0 122  0  2  0]  [  0  0 109  0  0]  [  0  0  0 98  0]  [  1  0  0  0 114]]  --- Classification Report ---               precision    recall  f1-score   support   arborio         0.99         0.96         0.97         116   basmati         1.00         0.98         0.99         124    ipsala         1.00         1.00         1.00         109    jasmine        0.95         1.00         0.98          98  karacadag        0.98         0.99         0.99         115   accuracy              0.99              0.99         0.99         562  macro avg              0.99              0.99         0.99         562  weighted avg           0.99              0.99         0.99         562 </pre>
2.	Tune the Model	<p><b>Training Log:</b>Used MobileNetV2 feature extractor + 2 Dense layers</p> <p><b>Validation Accuracy:</b> Consistently high(98.4%)</p> <p><b>Best Epoch Accuracy:</b> 99.33%</p> <p><b>Validation Method:</b> 70/15/15 split with 10 epochs</p>	<pre> history=model.fit(x_train,y_train,epochs=10,validation_data=(x_val,y_val))  Epoch 1/10 71/71 [=====] - 24s 287ms/step - loss: 0.5110 - accuracy: 0.8671 - val_loss: 0.1798 - val_accuracy: 0.9787 Epoch 2/10 71/71 [=====] - 38s 535ms/step - loss: 0.1426 - accuracy: 0.9698 - val_loss: 0.1135 - val_accuracy: 0.9840 Epoch 3/10 71/71 [=====] - 31s 431ms/step - loss: 0.1000 - accuracy: 0.9800 - val_loss: 0.0860 - val_accuracy: 0.9840 Epoch 4/10 71/71 [=====] - 30s 430ms/step - loss: 0.0798 - accuracy: 0.9836 - val_loss: 0.0703 - val_accuracy: 0.9840 Epoch 5/10 71/71 [=====] - 32s 440ms/step - loss: 0.0656 - accuracy: 0.9862 - val_loss: 0.0637 - val_accuracy: 0.9840 Epoch 6/10 71/71 [=====] - 31s 432ms/step - loss: 0.0564 - accuracy: 0.9898 - val_loss: 0.0571 - val_accuracy: 0.9840 Epoch 7/10 71/71 [=====] - 39s 540ms/step - loss: 0.0526 - accuracy: 0.9893 - val_loss: 0.0548 - val_accuracy: 0.9894 Epoch 8/10 71/71 [=====] - 37s 526ms/step - loss: 0.0458 - accuracy: 0.9920 - val_loss: 0.0482 - val_accuracy: 0.9840 Epoch 9/10 71/71 [=====] - 38s 529ms/step - loss: 0.0408 - accuracy: 0.9924 - val_loss: 0.0503 - val_accuracy: 0.9840 Epoch 10/10 71/71 [=====] - 38s 537ms/step - loss: 0.0363 - accuracy: 0.9933 - val_loss: 0.0471 - val_accuracy: 0.9840 </pre>