**🌾 1. Project Overview**

**GrainPalette** is an AI-powered model built to classify rice grain types using image input. Leveraging transfer learning, it identifies **five rice varieties** by analyzing visual features of grains. Users simply upload an image to receive type predictions.[ijeais.org+12github.com+12github.com+12](https://github.com/MeghanaBehara12/GrainPalette---A-Deep-Learning-Odyssey-In-Rice-Type-Classification-Through-Transfer-Learning?utm_source=chatgpt.com)

**2. Dataset & Scope**

* Built on the **Rice Image Dataset** (from Kaggle: ~75k images; ~15k per class)[reddit.com+4github.com+4github.com+4](https://github.com/ujwalaamulya1/Grain-Palette-a-deep-learning-odyssey-in-rice?utm_source=chatgpt.com)
* Common rice types include Arborio, Basmati, Ipsala, Jasmine, and Karacadag
* Dataset split: approx. **70% training / 30% testing** — ~10,500 train vs. 4,500 test images per class[github.com+3researchgate.net+3ijeais.org+3](https://www.researchgate.net/publication/391001738_Rice_Variety_Identification_Based_on_Transfer_Learning_Architecture_Using_DENS-INCEP?utm_source=chatgpt.com)

**3. Methodology**

* **Transfer Learning** using pre-trained CNN architectures from TensorFlow Hub (e.g., ResNet, MobileNet, DenseNet, VGG)
* Core pipeline:
  1. Image preprocessing (resize to 224×224, normalization)
  2. Freeze convolutional backbone, attach a custom fully‑connected head
  3. Use data augmentation, early stopping, LR schedulers[researchgate.net+1ijeais.org+1](https://www.researchgate.net/publication/391001738_Rice_Variety_Identification_Based_on_Transfer_Learning_Architecture_Using_DENS-INCEP?utm_source=chatgpt.com)[github.com](https://github.com/veronicamorelli/Rice-Grain-Image-Classification?utm_source=chatgpt.com)
* Implemented in **Python**, with **TensorFlow / Keras**, and tools like OpenCV, PIL, scikit‑learn, matplotlib[github.com](https://github.com/Ayush95697/Rice-Classification?utm_source=chatgpt.com)

**4. Performance & Benchmarks**

* **GrainPalette** achieves “high accuracy” (though exact numbers not stated in the repo)[github.com+4github.com+4github.com+4](https://github.com/Ayush95697/Rice-Classification?utm_source=chatgpt.com)
* Fair comparisons:
  + A **MobileNetV2** model (similar five-class task) reported ~**97.3%** accuracy[arxiv.org+14github.com+14github.com+14](https://github.com/Kkoustav/Rice-grain-classification?utm_source=chatgpt.com)
  + A comparative study using EfficientNet, ResNet, VGG, and MobileNet on 75k images found:
    - **EfficientNet**: 99.76% accuracy
    - **MobileNet**: fastest classification (~2556 s total)[ijeais.org+2sciencedirect.com+2researchgate.net+2](https://www.sciencedirect.com/science/article/pii/S2666154323003976?utm_source=chatgpt.com)
  + Another academic result achieved ~~99.96% accuracy across five varieties[github.com+3ijeais.org+3github.com+3](https://ijeais.org/wp-content/uploads/2024/4/abs/IJAISR240404.html?utm_source=chatgpt.com)

➡️ Likely, GrainPalette's performance aligns in the **~97–99% range**, depending on backbone and training protocol.

**5. Technical Pipeline**

text

CopyEdit

Input Image (resized 224×224)

↓

Pre-trained Backbone (e.g. ResNet50, MobileNetV2) – weights frozen

↓

Augmentation + Batch Norm + Fully Connected Layers + Softmax

↓

Prediction: One of 5 rice types with confidence score

Training includes validation monitoring, early stopping to avoid overfitting, and test evaluation via confusion matrix and classification report.[github.com+2github.com+2github.com+2](https://github.com/Ayush95697/Rice-Classification?utm_source=chatgpt.com)

**6. Strengths & Limitations**

**✅ Strengths**

* Reusable backbone (transfer learning cuts down training cost)
* Strong visual classification across common rice types
* Modular pipelines (can easily swap backbones)
* Open-source and extensible for deployment

**⚠️ Limitations**

* Performance specifics (e.g. accuracy per class) not clearly documented
* Requires dataset bias checks: lighting, grain orientation, background uniformity
* No post-deployment interface besides notebook (future deployment needed)

**7. Future Improvements**

* **Backbone experimentation**: EfficientNet for better accuracy, or MobileNet for faster inference
* **Hyperparameter tuning**: Learning rates, dropout, augmentations
* **Dataset expansion**: Rare or regional varieties to increase model coverage
* **Explainable AI**: Grad-CAM or saliency maps to visualize decision-making
* **Deployment**: Web/mobile app with real-time grain identification

These align with recommendations in the repo and wider research trends.[github.com+12github.com+12ijeais.org+12](https://github.com/Ayush95697/Rice-Classification?utm_source=chatgpt.com)

**8. Context & Industry Benchmarking**

| **Study/Model** | **Architecture** | **Dataset Size** | **Accuracy** |
| --- | --- | --- | --- |
| GrainPalette | Transfer Learning (various) | 75 k images (5 classes) | “High” (est. 97–99%) |
| Kkoustav et al. | MobileNetV2 | Five varieties | 97.33% [github.com](https://github.com/Kkoustav/Rice-grain-classification?utm_source=chatgpt.com) |
| ScienceDirect study | EfficientNet | 75 k images | 99.76% |
| IJAISR 2024 | CNN (scratch/transfer) | 75 k images | 99.96% |

dataset/

├── train/

│ ├── Basmati/

│ ├── Arborio/

│ ├── Jasmine/

│ ├── Ipsala/

│ └── Karacadag/

└── test/

└── (same structure as above)

✅ Requirements

First, install the required libraries:

bash

Copy

Edit

pip install tensorflow matplotlib scikit-learn

📦 Transfer Learning Code (MobileNetV2)

python

Copy

Edit

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications import MobileNetV2

from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout

from tensorflow.keras.models import Model

from sklearn.metrics import classification\_report, confusion\_matrix

import matplotlib.pyplot as plt

import numpy as np

import os

# Set paths

train\_dir = 'dataset/train'

test\_dir = 'dataset/test'

image\_size = (224, 224)

batch\_size = 32

num\_classes = 5 # Change this if your dataset has a different number of classes

# Data Augmentation

train\_datagen = ImageDataGenerator(

rescale=1./255,

zoom\_range=0.2,

horizontal\_flip=True,

rotation\_range=10

)

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

# Data Loaders

train\_gen = train\_datagen.flow\_from\_directory(

train\_dir, target\_size=image\_size, batch\_size=batch\_size, class\_mode='categorical'

)

test\_gen = test\_datagen.flow\_from\_directory(

test\_dir, target\_size=image\_size, batch\_size=batch\_size, class\_mode='categorical', shuffle=False

)

# Build Model using Transfer Learning

base\_model = MobileNetV2(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))

base\_model.trainable = False # Freeze base layers

# Custom Classification Head

x = base\_model.output

x = GlobalAveragePooling2D()(x)

x = Dropout(0.3)(x)

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

predictions = Dense(num\_classes, activation='softmax')(x)

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

# Compile Model

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

# Train

history = model.fit(train\_gen, validation\_data=test\_gen, epochs=10)

# Evaluate

test\_gen.reset()

predictions = model.predict(test\_gen, verbose=1)

y\_pred = np.argmax(predictions, axis=1)

y\_true = test\_gen.classes

# Classification Report

print("Classification Report:")

print(classification\_report(y\_true, y\_pred, target\_names=list(test\_gen.class\_indices.keys())))

# Confusion Matrix

print("Confusion Matrix:")

print(confusion\_matrix(y\_true, y\_pred))

# Plot Accuracy

plt.plot(history.history['accuracy'], label='train')

plt.plot(history.history['val\_accuracy'], label='val')

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend()

plt.show()

🔄 To Fine-tune Further

If you want to unfreeze some of MobileNetV2's top layers:

python

Copy

Edit

base\_model.trainable = True

for layer in base\_model.layers[:-30]: # Freeze all but the top 30 layers

layer.trainable = False

model.compile(optimizer=tf.keras.optimizers.Adam(1e-5), loss='categorical\_crossentropy', metrics=['accuracy'])

🧪 Prediction on Single Image

python

Copy

Edit

from tensorflow.keras.preprocessing import image

import numpy as np

def predict\_rice\_type(img\_path):

img = image.load\_img(img\_path, target\_size=image\_size)

img\_array = image.img\_to\_array(img) / 255.0

img\_array = np.expand\_dims(img\_array, axis=0)

prediction = model.predict(img\_array)

class\_names = list(train\_gen.class\_indices.keys())

predicted\_class = class\_names[np.argmax(prediction)]

print(f"Predicted rice type: {predicted\_class}")

# Example

predict\_rice\_type('sample\_rice.jpg')

**📂 1. Adaptable PowerPoint Templates**

**▶️ "AI Convolutional Neural Network for Rice Grain Classification" (SlideShare)**

* Contains an overview, dataset info, model architecture, and results layout
* A solid starting point—download and customize with GrainPalette-specific details [slideserve.com+5github.com+5github.com+5](https://github.com/MeghanaBehara12/GrainPalette---A-Deep-Learning-Odyssey-In-Rice-Type-Classification-Through-Transfer-Learning?utm_source=chatgpt.com)[github.com+1github.com+1](https://github.com/Shardy2907/CNN-using-Transfer-Learning?utm_source=chatgpt.com)

**▶️ "Technical seminar on classification of rice granules using image processing and neural network" (SlideServe)**

* Covers image acquisition, preprocessing, feature extraction methods
* Useful for visualizing how your dataset is processed [slideserve.com](https://www.slideserve.com/bud/technical-seminar-on-classification-of-rice-granules-using-image-processing-and-neural-network?utm_source=chatgpt.com)

**🛠 2. Slide Deck Outline**

You can craft your PPT with roughly **10–12 slides**:

1. **Title** – GrainPalette: Deep Learning in Rice Type Classification
2. **Introduction** – Problem, motivation, importance
3. **Dataset** – 75k images, 5 rice types, train/test split
4. **Transfer Learning Overview** – Benefits, why chosen
5. **Model Architecture** – Pre‑trained CNN + custom head
6. **Training Pipeline** – Preprocessing, augmentation, freezing & fine-tuning
7. **Performance Metrics** – Accuracy, classification report, confusion matrix
8. **Comparison with Literature** – e.g. ~97–99% accuracy using EfficientNet/VGG16 [github.com+3github.com+3reddit.com+3](https://github.com/Shardy2907/CNN-using-Transfer-Learning?utm_source=chatgpt.com)[slideserve.com+1slideshare.net+1](https://www.slideserve.com/bud/technical-seminar-on-classification-of-rice-granules-using-image-processing-and-neural-network?utm_source=chatgpt.com)[github.com](https://github.com/Ayush95697/Rice-Classification?utm_source=chatgpt.com)
9. **Visual Demos** – Sample predictions with confidence
10. **Strengths & Limitations**
11. **Future Work** – Explainability, deployment, more varieties
12. **Q&A / References** – Cite GitHub repo, Kaggle dataset, papers