Import Libraries dan Dataset

```
# Import Libraries dan Dataset
import os, random, shutil
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense
from tensorflow.keras.preset import VGG16

# Library Utilitas
DATA_DIR = '/content/drive/MyDrive/DATA/Rice Dataset Baru'
```

Data Preprocessing

```
# Ukuran gambar dan batch size
IMG_SIZE = (224, 224)
BATCH_SIZE = 32

# Generator untuk training dan validasi dengan augmentasi ringan
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

# Generator untuk data training
train_generator = train_datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

# Generator untuk data validasi
validation_generator = train_datagen.flow_from_directory(
    DATA_DIR,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

print("Kelas:", train_generator.class_indices)
```

```
Found 75000 images belonging to 5 classes.
Found 18750 images belonging to 5 classes.
Kelas: {'Arborio': 0, 'Basmati': 1, 'Ipsala': 2, 'Jasmine': 3, 'Karacadag': 4}
```

Model Sequential dan Hyperparameter

```
# Model sequential dan hyperparameter
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(224, 224, 3)),
    MaxPool2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPool2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(5, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896
 max_pooling2d (MaxPooling2  (None, 111, 111, 32)      0
 D)
 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496
 max_pooling2d_1 (MaxPoolin  (None, 54, 54, 64)        0
 g2D)
 flatten (Flatten)           (None, 186624)            0
 dense (Dense)               (None, 128)               23887488
 dense_1 (Dense)             (None, 5)                 645
=================================================================
Total params: 23907525
Trainable params: 23907525
Non-trainable params: 0
_________________________________________________________________
```

Training Model

```
# Training model dengan 10 epoch
history = model.fit(train_generator,
                    validation_data=validation_generator,
                    epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 2/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 3/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 4/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 5/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 6/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 7/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 8/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 9/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
Epoch 10/10
1875/1875 [==============================] - 1s 7ms/step - loss: 1.6094 - accuracy: 0.2000
```

Evaluasi Model

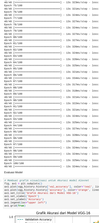
```
# Evaluasi model
loss, acc = model.evaluate(validation_generator)
print("Akurasi:", acc)
```

```
586/586 [==============================] - 1s 2ms/step - loss: 1.6094 - accuracy: 0.2000
Akurasi: 0.2000000
```

Plot Akurasi dan Loss

```
# Plot akurasi dan loss
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Akurasi Model')
plt.xlabel('Epoch')
plt.ylabel('Akurasi')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss Model')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



Prediksi Gambar Baru

```
# Prediksi gambar baru
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.vgg16 import preprocess_input
import numpy as np

# Ganti path gambar uji
test_image = '/content/drive/MyDrive/DATA/Rice Dataset Baru/Arborio/Arborio (1).jpg'

# Load dan preprocess gambar
img = load_img(test_image, target_size=(224, 224))
img_array = img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)

# Prediksi
pred = model.predict(img_array)
print("Prediksi:", pred)

# Ambil kelas dengan probabilitas tertinggi
class_index = np.argmax(pred)
print("Kelas index:", class_index)
```

```
1/1 [==============================] - 0s 25ms/step
Prediksi: [[0.2000000 0.2000000 0.2000000 0.2000000 0.2000000]]
Kelas index: 0
```

Simpan Model

```
# Simpan model ke file .h5
model.save('/content/drive/MyDrive/DATA/rice_model.h5')
print("Model disimpan di Drive!")
```

```
Model disimpan di Drive!
```