LAMPIRAN

Lampiran 1. Dokumentasi Pengumpulan Data

Contoh citra ukiran Bali kelas Taliilut yang berhasil dikumpulkan.

Pemotongan citra ukiran Taliilut

Lampiran 2. Praktisi

Pakar/Praktisi 1	Nama : I Komang Subrata, S.Pd, M.Pd Tempat tgl lahir : Karangasem, 17 Juli 1965 Alamat : Jln Delod puri, Singapadu, Gianyar Tempat Kerja : Guru Seni Kriya Kayu SMKN 2 Sukawati Pengalaman : Juri PKB Bidang bidang kerajinan tahun 2006 Duta seni sebagai pengajar ukiran di Denmark dan Swedia tahun 2002
Pakar/Praktisi 2	Nama : I Nyoman Karyana, S.Pd Tempat tgl lahir : Sibanggede, 14 Pebruari 1962 Alamat : Br Badung Sibang Gede, Abiansemal, Badung Tempat Kerja : Guru Kriya SMKN 1 Sukawati Pengalaman : Guru Ukir SMKN 1 Sukasada 1975-2005 Guru Ukir SMKN 1 Sukawati 2005-sekarang Usaha ukir 1975-1996
Pakar/Praktisi 3	Nama : I Made Suyatna, S.Sn Tempat tgl lahir : Mengwitani, 9 Agustus 1982 Alamat : Br. Culag Calig, Mengwitani, Mengwi, Badung Tempat Kerja : Guru Seni Budaya SMK PGRI 2 Badung Pengalaman : Sebagai Seniman Ornamen Lukis 2010-sekarang Guru Seni Budaya SMK PGRI 2 Badung 2014 - sekarang

Lampiran 3. Hasil Validasi Kategori Ukiran Bali

No	Nama Ukiran	Pakar 1	Pakar 2	Pakar 3
1.	Batu-Batuan	Batu-Batuan	Batu-Batuan	Batu-Batuan
2.	Batun Timun	Batun Timun	Batun Timun	Batun Timun
3.	Kakul-Kakulan	Kakul-Kakulan	Kakul-Kakulan	Kakul-Kakulan
4.	Mas-Masan	Mas-Masan	Mas-Masan	Mas-Masan
5.	Mote-Motean	Mote-Motean	Mote-Motean	Mote-Motean
6.	Pid-Pid	Pid-Pid	Pid-Pid	Pid-Pid
7.	Tali Ilut	Tali Ilut	Tali Ilut	Tali Ilut
8.	Patra Kuta Mesir	Patra Kuta Mesir	Patra Kuta Mesir	Patra Kuta Mesir
9.	Patra Bun-Bunan	Keketusan	Patra Bun-Bunan	Patra Bun-Bunan
10.	Patra Wangga	Patra Wangga	Patra Wangga	Patra Wangga
11.	Patra Sari	Patra Sari	Patra Sari	Patra Sari
12.	Patra Punggel	Patra Punggel	Patra Punggel	Patra Punggel
13.	Patra Samblung	Patra Samblung	Patra Samblung	Patra Samblung
14.	Patra Ulanda	Patra Ulanda	Patra Ulanda	Patra Ulanda
15.	Patra Cina	Patra Cina	Patra Cina	Patra Cina
16.	Patra Bali	Patra Sari	Patra Sari	Patra Bali
17.	Patra Banci	Patra Banci	Patra Banci	Patra Banci
18.	Patra Pae	Ganggong	Ganggong	Ganggong
19.	Patra Sulur	Bun	Sulur	Tidak Tahu
20.	Patra Ganggong	Paku-Pakuan	Paku	Tidak Tahu
21.	Patra Api-Apian	Patra Api-Apian	Patra Api-Apian	Patra Api-Apian
22.	Patra Kera	Relief	Patra Kera	Relief
23.	Patra Naga	Relief	Patra Naga	Karang Naga
24.	Patra Garuda	Relief	Patung Garuda	Karang Garuda
25.	Patra Singa	Relief	Patung Singa	Karang Singa

26.	Karang Simbar	Karang Simbar	Karang Simbar	Karang Simbar
27.	Karang Bunga	Karang Bunga	Karang Bunga	Karang Bunga
28.	Karang Boma	Karang Boma	Karang Boma	Karang Boma
29.	Karang Sae	Karang Sae	Karang Sae	Karang Sae
30.	Karang Gajah	Karang Gajah	Karang Gajah	Karang Gajah
31.	Karang Goak	Karang Goak	Karang Goak	Karang Goak
32.	Karang Tapel	Karang Tapel	Karang Tapel	Karang Tapel
33.	Karang Bentulu	Karang Bentulu	Karang Bentulu	Karang Bentulu
34.	Karang Batu	Karang Batu	Karang Batu	Karang Batu

Lampiran 4. Dokumentasi Validasi dan Pemberian Label Data

Lampiran 5. *Source Code* Implementasi Pengenalan Citra Ukiran

PENGENALAN CITRA UKIRAN ORNAMEN TRADISIONAL MENGGUNAKAN METODE DWT DANCNN

Berikut dijelaskan beberapa tahap dalam pengenalan citra ukiran ornamen tradisional.

Inisialisasi Modul

Pada tahap pertama dilakukan pemanggilan modul yang akan digunakan dalam pengolahan data.

```
import os
import time
import re
from glob import glob
import shutil
import numpy as np
import math
import matplotlib.pyplot as plt
import random
import pandas as pd
import itertools

from sklearn import metrics
from sklearn.metrics import confusion_matrix

import tensorflow as tf
from PIL import Image
from tensorflow.keras import datasets, layers, models
from google.colab import drive
import pywt


print("Tensorflow: v{}".format(tf._version_))
#%matplotlib inline
```

```
Tensorflow: v2.8.2
```

Definisi Prosedur

Pada Tahap kedua dilakukan proses pendefinisian prosedur dalam pemanggilan data dan melakukan proses DWT

```
def loadImage(f, label):
    # load the file into tensor
    image = tf.io.read_file(f)
    # Decode it to JPEG format
    image = tf.image.decode_jpeg(image)
    # Convert it to tf.float32
    image = tf.cast(image, tf.float32)
```

```
    return image / 255.0, label

def loadImage2(f, label):
    # load the file into tensor
    #print(f);
    image = tf.io.read_file(f)
    # Decode it to JPEG format
    image = tf.image.decode_jpeg(image)
    # Convert it to tf.float32
    image = tf.cast(image, tf.float32)


    return image, label

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])

def plot_confusion_matrix(cm, classes,
   normalize=False,
   title='Confusion matrix',
   cmap=plt.cm.Blues):

   if normalize:
     cm = (cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]) * 100

     print("Normalized confusion matrix")
   else:
     print("Confusion matrix, without normalization")

   plt.figure(figsize=(20,15))
   plt.imshow(cm, interpolation='nearest', cmap=cmap)
   plt.title(title)
   plt.colorbar()
   tick_marks = np.arange(len(classes))
   #print(tick_marks)
   plt.xticks(tick_marks, classes, rotation=45)
   plt.yticks(tick_marks, classes)

   fmt = '.2f' if normalize else 'd'
   thresh = cm.max() / 2.
   for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
      a = format(cm[i, j], fmt)
      if a == "nan":
        a = 0;

      #a = np.round(int(a),0);
      plt.text(j, i, a, horizontalalignment="center", color="white" if cm[i, j] > thresh e

   plt.ylabel('True label')
   plt.xlabel('Predicted label')

print("Pendefinisian Modul Berhasil")
```

```
Pendefinisian Modul Berhasil
```

Mounting Drive

Melakukan mounting terhadap google drive yang digunakan. pada implementasi ini menggunakan drive pada "/content/drive/MyDrive/Project/Tesis/Kurnia/Data Set Ukiran/Versi 1 (Rusdy)/

```
drive.mount('/content/drive',);

dir = "/content/drive/MyDrive/dataset utk revisi/Data Training/";
os.chdir(dir);
list_kategori_training = sorted(os.listdir());
print(list_kategori_training)

dir2 = "/content/drive/MyDrive//dataset utk revisi/Data Testing/";
os.chdir(dir2);
list_kategori_testing = sorted(os.listdir());
print(list_kategori_testing)
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
['01_Keketusan', '02_Pepatraan', '03_Kekarangan']
['01_Keketusan', '02_Pepatraan', '03_Kekarangan']
```

Mendefinisikan Variabel dan Menyimpan Data Sampel

Memanggil setiap gambar yang ada pada setiap folder, yang menggambarkan class itu sendiri. Sampel yang dipanggil dimasukkan ke dalam variabel **data_train** dan **label_train** untuk sampel data Pelatihan dan variabel **data_tes** dan **label_test** untuk sampel data Pengujian

```
data_train_head = {};
data_test_head = {};
label_train_head = {};
label_test_head = {};
index_kategori_head = 0;
index_train_head = 0;
index_test_head = 0;
nama_class_head = {};
jml_data_train_head = {};
jml_data_test_head = {};

data_train_detail = {};
data_test_detail = {};
label_train_detail = {};
label_test_detail = {};
index_kategori_detail = 0;
index_train_detail = 0;
index_test_detail = 0;
nama_class_detail = {};
jml_data_train_detail = {};
```

```
jml_data_test_detail = {};
jumlah_class_head = len(list_kategori_training);
jumlah_class_detail = {};
list_gambar_detail = [];
index_kategori_detail = 0;
index_kategori_head = 0;

kelompok_data_train = {};
i = 0;
for kategori_head in list_kategori_training:
  pilih_folder_head = dir + kategori_head + "/"
  os.chdir(pilih_folder_head);
  list_head = sorted(os.listdir());
  # print(pilih_folder);
  # print(len(list_gambar));
  # print(jml_data);
  # print(jml_train);
  # jml_data_train[index_kategori] = len(list_gambar);
  # print(list_head)
  jumlah_class_detail[index_kategori_head] = len(list_head)

  for kategori_detail in list_head:
    pilih_folder_detail = pilih_folder_head + kategori_detail + "/"
    os.chdir(pilih_folder_detail);
    list_detail = sorted(os.listdir());

    jml_data_train_detail[index_kategori_detail] = len(list_detail);
    # print(list_detail);
    for gambar in list_detail:
      # print(gambar);
      labelx_train_detail, labelx_detail = loadImage2(gambar, index_kategori_detail);

      # index_train_detail = index_train_detail + 1;

      if len(label_train_detail) == 0:
        data_train_detail = np.array([labelx_train_detail])
        data_train_head = np.array([labelx_train_detail])
        label_train_detail = np.array([labelx_detail])
        label_train_head = np.array([index_kategori_head])

      else:
        labelx_train_detail = np.array([labelx_train_detail])
        data_train_detail = np.concatenate([data_train_detail, labelx_train_detail])
        data_train_head = np.concatenate([data_train_head, labelx_train_detail])
        labelx_detail = np.array([labelx_detail])
        labelx_head = np.array([index_kategori_head])
        #print(label_train)
        label_train_detail = np.concatenate([label_train_detail, labelx_detail])
        label_train_head = np.concatenate([label_train_head, labelx_head])

      kelompok_data_train[i,0] = index_kategori_head;
      kelompok_data_train[i,1] = index_kategori_detail;
      i = i + 1;

    if len(nama_class_detail) == 0:
```

```
        nama_class_detail = np.array([kategori_detail]);
      else:
        nama_class_detail = np.concatenate([nama_class_detail,np.array([kategori_detail])]);

    #print(list_gambar[0]);

      index_kategori_detail = index_kategori_detail + 1;

    if len(nama_class_head) == 0:
        nama_class_head = np.array([kategori_head]);
    else:
      nama_class_head = np.concatenate([nama_class_head,np.array([kategori_head])]);
    index_kategori_head = index_kategori_head + 1;

index_kategori_detail = 0;
index_kategori_head = 0;
kelompok_data_test = {};
i = 0;
for kategori_head in list_kategori_testing:
  pilih_folder_head = dir2 + kategori_head + "/"
  os.chdir(pilih_folder_head);
  list_head = sorted(os.listdir());
  # print(pilih_folder);
  # print(len(list_gambar));
  # print(jml_data);
  # print(jml_train);
  # jml_data_train[index_kategori] = len(list_gambar);
  # print(list_head)
  for kategori_detail in list_head:
    pilih_folder_detail = pilih_folder_head + kategori_detail + "/"
    os.chdir(pilih_folder_detail);
    list_detail = sorted(os.listdir());

    jml_data_test_detail[index_kategori_detail] = len(list_detail);
    # print(list_detail);
    for gambar in list_detail:
      # print(gambar);
      labelx_test_detail, labelx_detail = loadImage2(gambar, index_kategori_detail);

      # index_train_detail = index_train_detail + 1;

      if len(label_test_detail) == 0:
        data_test_detail = np.array([labelx_test_detail])
        data_test_head = np.array([labelx_test_detail])
        label_test_detail = np.array([labelx_detail])
        label_test_head = np.array([index_kategori_head])

      else:
        labelx_test_detail = np.array([labelx_test_detail])
        data_test_detail = np.concatenate([data_test_detail, labelx_test_detail])
        data_test_head = np.concatenate([data_test_head, labelx_test_detail])
        labelx_detail = np.array([labelx_detail])
        labelx_head = np.array([index_kategori_head])
        #print(label_train)
        label_test_detail = np.concatenate([label_test_detail, labelx_detail])
```

```
          label_test_head = np.concatenate([label_test_head, labelx_head])



        kelompok_data_test[i,0] = index_kategori_head;
        kelompok_data_test[i,1] = index_kategori_detail;
        i = i + 1;

      # if len(nama_class_detail) == 0:
      #   nama_class_detail = np.array([kategori_detail]);
      # else:
      #   nama_class_detail = np.concatenate([nama_class_detail,np.array([kategori_detail])]

    #print(list_gambar[0]);

      index_kategori_detail = index_kategori_detail + 1;
    index_kategori_head = index_kategori_head + 1;

    # if len(nama_class_head) == 0:
    #     nama_class_head = np.array([kategori_head]);
    # else:
    #   nama_class_head = np.concatenate([nama_class_head,np.array([kategori_head])]);
    # index_kategori_head = index_kategori_head + 1;

print("selesai");

     selesai


print(nama_class_detail)

     ['Batu-batuan' 'Batun Timun' 'Kakul-kakulan' 'Mas-masan' 'Mote-motean'
      'Pidpid' 'Tali ilut' 'Patra Api' 'Patra Banci' 'Patra Cina' 'Patra Mesir'
      'Patra Punggel' 'Patra Samblung' 'Patra Sari' 'Patra Ulanda'
      'Patra Wangga' 'Karang Batu' 'Karang Bentulu' 'Karang Boma'
      'Karang Bunga' 'Karang Gajah' 'Karang Goak' 'Karang Sae' 'Karang Simbar'
      'Karang Tapel']


data_train_Keketusan = {};
data_test_Keketusan = {};
data_train_Pepatraan = {};
data_test_Pepatraan = {};
data_train_Kekarangan = {};
data_test_Kekarangan = {};
label_train_Keketusan = {};
label_test_Keketusan = {};
label_train_Pepatraan = {};
label_test_Pepatraan = {};
label_train_Kekarangan = {};
label_test_Kekarangan = {};
nama_class_keketusan = {};
nama_class_pepatraan = {};
nama_class_kekarangan = {};
```

```
index_kategori_head = 0;
class_head = {};
jml_data_train_head = {};
jml_data_test_head = {};
total_data_head = 0;
index_head = 0;
p_train = int(len(kelompok_data_train)/2);
p_test = int(len(kelompok_data_test)/2);

for i in range(p_train):
  # print(kelompok_data);
  if(kelompok_data_train[i,0] == 0):
    # print(label_train_Keketusan)
    if(len(label_train_Keketusan) == 0):
      data_train_Keketusan = np.array([data_train_detail[i]]);
      label_train_Keketusan = np.array([label_train_detail[i]]);

    else:
      data = np.array([data_train_detail[i]]);
      label = np.array([label_train_detail[i]]);
      data_train_Keketusan = np.concatenate([data_train_Keketusan, data]);
      label_train_Keketusan = np.concatenate([label_train_Keketusan, label]);

  elif kelompok_data_train[i,0] == 1:
    if(len(label_train_Pepatraan) == 0):
      data_train_Pepatraan = np.array([data_train_detail[i]]);
      label_train_Pepatraan = np.array([label_train_detail[i]]);

    else:
      data = np.array([data_train_detail[i]]);
      label = np.array([label_train_detail[i]]);
      data_train_Pepatraan = np.concatenate([data_train_Pepatraan, data]);
      label_train_Pepatraan = np.concatenate([label_train_Pepatraan, label]);


  elif kelompok_data_train[i,0] == 2:
    if(len(label_train_Kekarangan) == 0):
      data_train_Kekarangan = np.array([data_train_detail[i]]);
      label_train_Kekarangan = np.array([label_train_detail[i]]);

    else:
      data = np.array([data_train_detail[i]]);
      label = np.array([label_train_detail[i]]);
      data_train_Kekarangan = np.concatenate([data_train_Kekarangan, data]);
      label_train_Kekarangan = np.concatenate([label_train_Kekarangan, label]);


for i in range(p_test):
  # print(kelompok_data);
  if(kelompok_data_test[i,0] == 0):
    # print(label_test_Keketusan)
    if(len(label_test_Keketusan) == 0):
      data_test_Keketusan = np.array([data_test_detail[i]]);
      label_test_Keketusan = np.array([label_test_detail[i]]);
```

```
      else:
        data = np.array([data_test_detail[i]]);
        label = np.array([label_test_detail[i]]);
        data_test_Keketusan = np.concatenate([data_test_Keketusan, data]);
        label_test_Keketusan = np.concatenate([label_test_Keketusan, label]);

    elif kelompok_data_test[i,0] == 1:
      if(len(label_test_Pepatraan) == 0):
        data_test_Pepatraan = np.array([data_test_detail[i]]);
        label_test_Pepatraan = np.array([label_test_detail[i]]);

      else:
        data = np.array([data_test_detail[i]]);
        label = np.array([label_test_detail[i]]);
        data_test_Pepatraan = np.concatenate([data_test_Pepatraan, data]);
        label_test_Pepatraan = np.concatenate([label_test_Pepatraan, label]);

    elif kelompok_data_test[i,0] == 2:
      if(len(label_test_Kekarangan) == 0):
        data_test_Kekarangan = np.array([data_test_detail[i]]);
        label_test_Kekarangan = np.array([label_test_detail[i]]);

      else:
        data = np.array([data_test_detail[i]]);
        label = np.array([label_test_detail[i]]);
        data_test_Kekarangan = np.concatenate([data_test_Kekarangan, data]);
        label_test_Kekarangan = np.concatenate([label_test_Kekarangan, label]);

  for kategori_head in list_kategori_training:
    print("=============================================================================")
    print("|\t\t\t|\tJumlah Data Sampel " + kategori_head + "\t\t|")
    print("|\tKategori\t|-------------------------------------------------------------|")
    print("|\t\t\t|       Pelatihan \t|       Pengujian \t|")
    print("=============================================================================")
    jml_data_train = {};
    class_index = {};
    total_data_train = 0;
    head = -1;
    index = -1;


    for i in range(p_train):
      # print(kelompok_data);
      if(kelompok_data_train[i,0] == index_kategori_head):
        total_data_train = total_data_train + 1;
        if(kelompok_data_train[i,1] == head):
          jml_data_train[index] = jml_data_train[index] + 1;
        else:
          index = index + 1;
          head = kelompok_data_train[i,1];
          jml_data_train[index] = 1;
          class_index[index] = head;
```

```
jml_data_test = {};
class_index = {};
total_data_test = 0;
head = -1;
index = -1;
for i in range(p_test):
  # print(kelompok_data);
  if(kelompok_data_test[i,0] == index_kategori_head):
    total_data_test = total_data_test + 1;
    if(kelompok_data_test[i,1] == head):
      jml_data_test[index] = jml_data_test[index] + 1;
    else:
      index = index + 1;
      head = kelompok_data_test[i,1];
      jml_data_test[index] = 1;
      class_index[index] = head;

class_head[index_head] = kategori_head;
jml_data_train_head[index_head] = total_data_train;
jml_data_test_head[index_head] = total_data_test;
index_head = index_head + 1;

for i in range(index+1):
  if(index_kategori_head == 0):
    if len(nama_class_keketusan) == 0:
      nama_class_keketusan = np.array([nama_class_detail[class_index[i]]]);
    else:
      nama_class_keketusan = np.concatenate([nama_class_keketusan ,np.array([nama_class_

  elif(index_kategori_head == 1):
    if len(nama_class_pepatraan) == 0:
      nama_class_pepatraan = np.array([nama_class_detail[class_index[i]]]);
    else:
      nama_class_pepatraan = np.concatenate([nama_class_pepatraan ,np.array([nama_class_

  elif(index_kategori_head == 2):
    if len(nama_class_kekarangan) == 0:
      nama_class_kekarangan = np.array([nama_class_detail[class_index[i]]]);
    else:
      nama_class_kekarangan = np.concatenate([nama_class_kekarangan ,np.array([nama_clas

  if len(nama_class_detail[class_index[i]]) < 12:
    print("|   " + str(nama_class_detail[class_index[i]]) + "\t\t|\t    " + str(jml_data
  else:
    print("|   " + str(nama_class_detail[class_index[i]]) + "\t|\t    " + str(jml_data_t

index_kategori_head = index_kategori_head + 1;

print("================================================================================")
print("|   Total Data\t\t|\t   " + str(total_data_train) + "\t\t|\t   " + str(total_data
print("================================================================================")
print("\n\n\n");
```

```
print("=============================================================================")
print("|\t\t\t|\t\tJumlah Data Sampel\t\t|")
print("|\tKategori\t|-------------------------------------------------------------|")
print("|\t\t\t|       Pelatihan \t|       Pengujian \t|")
print("=============================================================================")
total_data_train = 0;
total_data_test = 0;
for i in range(index_head):
  if len(class_head[i]) < 12:
      print("|   " + str(class_head[i]) + "\t\t|\t    " + str(jml_data_train_head[i]) + "\
  else:
    print("|   " + str(class_head[i]) + "\t|\t    " + str(jml_data_train_head[i]) + "\t\t|
  total_data_train = total_data_train + jml_data_train_head[i];
  total_data_test = total_data_test + jml_data_test_head[i];


print("=============================================================================")
print("|   Total Data\t\t|\t   " + str(total_data_train) + "\t\t|\t   " + str(total_data_t
print("=============================================================================")
print("\n\n\n");

label_train_Pepatraan = label_train_Pepatraan - 7;
label_test_Pepatraan = label_test_Pepatraan - 7;
label_train_Kekarangan = label_train_Kekarangan - 16;
label_test_Kekarangan = label_test_Kekarangan - 16;
```

Kategori	Jumlah Data Sampel 01_Keketusan	
	Pelatihan	Pengujian
Batu-batuan	10	7
Batun Timun	13	8
Kakul-kakulan	9	6
Mas-masan	13	7
Mote-motean	7	6
Pidpid	19	7
Tali ilut	3	2
Total Data	74	43

Kategori	Jumlah Data Sampel 02_Pepatraan	
	Pelatihan	Pengujian
Patra Api	3	2
Patra Banci	11	7
Patra Cina	8	6
Patra Mesir	10	7
Patra Punggel	11	7
Patra Samblung	9	7
Patra Sari	10	7

Patra Ulanda	14	7
Patra Wangga	3	2
Total Data	79	52

Kategori	Jumlah Data Sampel 03_Kekarangan	
	Pelatihan	Pengujian
Karang Batu	3	2
Karang Bentulu	3	2
Karang Boma	3	2
Karang Bunga	3	2
Karang Gajah	11	8
Karang Goak	11	7
Karang Sae	3	2
Karang Simbar	3	2
Karang Tapel	19	7
Total Data	59	34

Proses Sub Class

Melakukan proses pada sub class keketusan - pepatraan - kekearangan

Menampilkan data

Menampilkan data pada terdapat pada data sampel

```
print("Keketusan")
plt.figure(figsize=(10,10))
index_kategori = -1;
j = 0;
for i in range(len(data_train_Keketusan)):
    if index_kategori != label_train_Keketusan[i]:
      index_kategori = label_train_Keketusan[i];
      plt.subplot(5,5,j+1)
      plt.xticks([])
      plt.yticks([])
      plt.grid(False)
      plt.imshow(data_train_Keketusan[i] / 255.0)
      plt.xlabel(nama_class_keketusan[label_train_Keketusan[i]])
      j = j+1;

plt.show()

print("\n\nPepatraan")
plt.figure(figsize=(10,10))
index_kategori = -1;
```

```
j = 0;
for i in range(len(data_train_Pepatraan)):
    if index_kategori != label_train_Pepatraan[i]:
      index_kategori = label_train_Pepatraan[i];
      plt.subplot(5,5,j+1)
      plt.xticks([])
      plt.yticks([])
      plt.grid(False)
      plt.imshow(data_train_Pepatraan[i] / 255.0)
      plt.xlabel(nama_class_pepatraan[label_train_Pepatraan[i]])
      j = j+1;

plt.show()

print("\n\nKekarangan")
plt.figure(figsize=(10,10))
index_kategori = -1;
j = 0;
for i in range(len(data_train_Kekarangan)):
    if index_kategori != label_train_Kekarangan[i]:
      index_kategori = label_train_Kekarangan[i];
      plt.subplot(5,5,j+1)
      plt.xticks([])
      plt.yticks([])
      plt.grid(False)
      plt.imshow(data_train_Kekarangan[i] / 255.0)
      plt.xlabel(nama_class_kekarangan[label_train_Kekarangan[i]])
      j = j+1;

plt.show()

print("\n\nAll")
plt.figure(figsize=(10,10))
index_kategori = -1;
j = 0;
for i in range(len(data_train_head)):
    if index_kategori != label_train_head[i]:
      index_kategori = label_train_head[i];
      plt.subplot(5,5,j+1)
      plt.xticks([])
      plt.yticks([])
      plt.grid(False)
      plt.imshow(data_train_head[i] / 255.0)
      plt.xlabel(nama_class_head[label_train_head[i]])
      j = j+1;

plt.show()
```

Keketusan

Pepatraan

Kekarangan

Keketusan - Proses DWT

Melakukan praproses dengan perintah Discrete Wavelet Transform

```
data_train_dwt_keketusan = {};
for i in range(len(data_train_Keketusan)):
  x1 = rgb2gray(data_train_Keketusan[i]);
  gambar = pywt.dwt2(x1, 'bior1.3')
  LL, (LH, HL, HH) = gambar;
  LH = np.array([LH]) /255.0
  LL = np.array([LL]) /255.0
  HL = np.array([HL]) /255.0
```

```
    HH = np.array([HH]) /255.0
    result = np.concatenate([LH, LL]);
    result = np.concatenate([result, HL]);
    result = np.concatenate([result, HH]);

    result = np.array([result])
    # print(result)
    # print(np.array(data_train_dwt).shape)
    result = np.transpose(result, (0, 3,2,1));
    if len(data_train_dwt_keketusan) == 0:
      data_train_dwt_keketusan = result;
      #print(data_train_dwt.shape);
    else:
      data_train_dwt_keketusan = np.concatenate([data_train_dwt_keketusan, result]);
      #print(data_train_dwt.shape);

  data_test_dwt_keketusan = {};
  for j in range(len(data_test_Keketusan)):
    x1 = rgb2gray(data_test_Keketusan[j]);
    gambar = pywt.dwt2(x1, 'bior1.3')
    LL, (LH, HL, HH) = gambar;
    LH = np.array([LH]) /255.0
    LL = np.array([LL]) /255.0
    HL = np.array([HL]) /255.0
    HH = np.array([HH]) /255.0
    result = np.concatenate([LH, LL]);
    result = np.concatenate([result, HL]);
    result = np.concatenate([result, HH]);
    result = np.array([result])
    result = np.transpose(result, (0, 3,2,1));
    if len(data_test_dwt_keketusan) == 0:
      data_test_dwt_keketusan = result;
      #print(data_train_dwt.shape);
    else:
      data_test_dwt_keketusan = np.concatenate([data_test_dwt_keketusan, result]);


  print("Proses DWT Keketusan Berhasil ")


  data_train_dwt_pepatraan = {};
  for i in range(len(data_train_Pepatraan)):
    x1 = rgb2gray(data_train_Pepatraan[i]);
    gambar = pywt.dwt2(x1, 'bior1.3')
    LL, (LH, HL, HH) = gambar;
    LH = np.array([LH]) /255.0
    LL = np.array([LL]) /255.0
    HL = np.array([HL]) /255.0
    HH = np.array([HH]) /255.0
    result = np.concatenate([LH, LL]);
    result = np.concatenate([result, HL]);
    result = np.concatenate([result, HH]);

    result = np.array([result])
    # print(result)
```

```
    # print(np.array(data_train_dwt).shape)
    result = np.transpose(result, (0, 3,2,1));
    if len(data_train_dwt_pepatraan) == 0:
      data_train_dwt_pepatraan = result;
      #print(data_train_dwt.shape);
    else:
      data_train_dwt_pepatraan = np.concatenate([data_train_dwt_pepatraan, result]);
      #print(data_train_dwt.shape);

  data_test_dwt_pepatraan = {};
  for j in range(len(data_test_Pepatraan)):
    x1 = rgb2gray(data_test_Pepatraan[j]);
    gambar = pywt.dwt2(x1, 'bior1.3')
    LL, (LH, HL, HH) = gambar;
    LH = np.array([LH]) /255.0
    LL = np.array([LL]) /255.0
    HL = np.array([HL]) /255.0
    HH = np.array([HH]) /255.0
    result = np.concatenate([LH, LL]);
    result = np.concatenate([result, HL]);
    result = np.concatenate([result, HH]);
    result = np.array([result])
    result = np.transpose(result, (0, 3,2,1));
    if len(data_test_dwt_pepatraan) == 0:
      data_test_dwt_pepatraan = result;
      #print(data_train_dwt.shape);
    else:
      data_test_dwt_pepatraan = np.concatenate([data_test_dwt_pepatraan, result]);


  print("Proses DWT Pepatraan Berhasil ")

  data_train_dwt_kekarangan = {};
  for i in range(len(data_train_Kekarangan)):
    x1 = rgb2gray(data_train_Kekarangan[i]);
    gambar = pywt.dwt2(x1, 'bior1.3')
    LL, (LH, HL, HH) = gambar;
    LH = np.array([LH]) /255.0
    LL = np.array([LL]) /255.0
    HL = np.array([HL]) /255.0
    HH = np.array([HH]) /255.0
    result = np.concatenate([LH, LL]);
    result = np.concatenate([result, HL]);
    result = np.concatenate([result, HH]);

    result = np.array([result])
    # print(result)
    # print(np.array(data_train_dwt).shape)
    result = np.transpose(result, (0, 3,2,1));
    if len(data_train_dwt_kekarangan) == 0:
      data_train_dwt_kekarangan = result;
      #print(data_train_dwt.shape);
    else:
      data_train_dwt_kekarangan = np.concatenate([data_train_dwt_kekarangan, result]);
      #print(data_train_dwt.shape);
```

```
data_test_dwt_kekarangan = {};
for j in range(len(data_test_Kekarangan)):
  x1 = rgb2gray(data_test_Kekarangan[j]);
  gambar = pywt.dwt2(x1, 'bior1.3')
  LL, (LH, HL, HH) = gambar;
  LH = np.array([LH]) /255.0
  LL = np.array([LL]) /255.0
  HL = np.array([HL]) /255.0
  HH = np.array([HH]) /255.0
  result = np.concatenate([LH, LL]);
  result = np.concatenate([result, HL]);
  result = np.concatenate([result, HH]);
  result = np.array([result])
  result = np.transpose(result, (0, 3,2,1));
  if len(data_test_dwt_kekarangan) == 0:
    data_test_dwt_kekarangan = result;
    #print(data_train_dwt.shape);
  else:
    data_test_dwt_kekarangan = np.concatenate([data_test_dwt_kekarangan, result]);


print("Proses DWT Kekarangan Berhasil ")


data_train_dwt_all = {};
for i in range(len(data_train_head)):
  x1 = rgb2gray(data_train_head[i]);
  gambar = pywt.dwt2(x1, 'bior1.3')
  LL, (LH, HL, HH) = gambar;
  LH = np.array([LH]) /255.0
  LL = np.array([LL]) /255.0
  HL = np.array([HL]) /255.0
  HH = np.array([HH]) /255.0
  result = np.concatenate([LH, LL]);
  result = np.concatenate([result, HL]);
  result = np.concatenate([result, HH]);

  result = np.array([result])
  # print(result)
  # print(np.array(data_train_dwt).shape)
  result = np.transpose(result, (0, 3,2,1));
  if len(data_train_dwt_all) == 0:
    data_train_dwt_all = result;
    #print(data_train_dwt.shape);
  else:
    data_train_dwt_all = np.concatenate([data_train_dwt_all, result]);
    #print(data_train_dwt.shape);

data_test_dwt_all = {};
for j in range(len(data_test_head)):
  x1 = rgb2gray(data_test_head[j]);
  gambar = pywt.dwt2(x1, 'bior1.3')
  LL, (LH, HL, HH) = gambar;
  LH = np.array([LH]) /255.0
```

```
  LL = np.array([LL]) /255.0
  HL = np.array([HL]) /255.0
  HH = np.array([HH]) /255.0
  result = np.concatenate([LH, LL]);
  result = np.concatenate([result, HL]);
  result = np.concatenate([result, HH]);
  result = np.array([result])
  result = np.transpose(result, (0, 3,2,1));
  if len(data_test_dwt_all) == 0:
    data_test_dwt_all  =  result;
    #print(data_train_dwt.shape);
  else:
    data_test_dwt_all = np.concatenate([data_test_dwt_all, result]);


print("Proses DWT All Berhasil ")
```

```
Proses DWT Keketusan Berhasil
Proses DWT Pepatraan Berhasil
Proses DWT Kekarangan Berhasil
Proses DWT All Berhasil
```

Contoh Hasil DWT

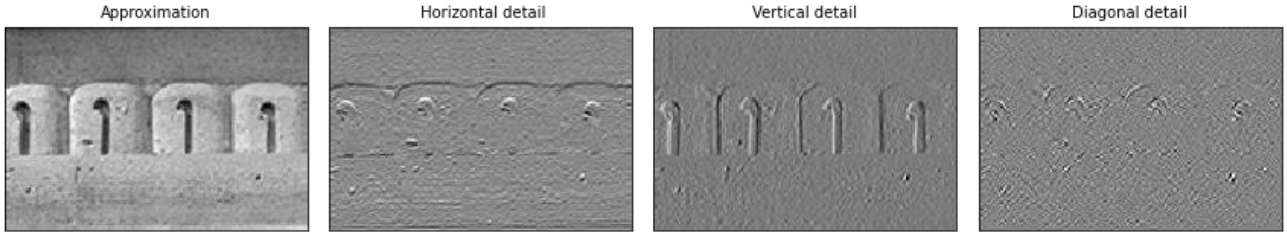
Berikut salah satu dari hasil proses DWT pada data training

```
print("Keketusan")
index_kategori = -1;
j = 0;
for i in range(len(data_train_Keketusan)):
    if index_kategori != label_train_Keketusan[i]:
      index_kategori = label_train_Keketusan[i];

      x1 = rgb2gray(data_train_Keketusan[i]);
      gambar = pywt.dwt2(x1, 'bior1.3')
      LL, (LH, HL, HH) = gambar
      print(nama_class_keketusan[label_train_Keketusan[i]])
      # Wavelet transform of image, and plot approximation and details
      titles = ['Approximation', ' Horizontal detail',
                'Vertical detail', 'Diagonal detail']
      fig = plt.figure(figsize=(12, 3))
      for i, a in enumerate([LL, LH, HL, HH]):
          ax = fig.add_subplot(1, 4, i + 1)
          ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
          ax.set_title(titles[i], fontsize=10)
          ax.set_xticks([])
          ax.set_yticks([])

      fig.tight_layout()
      plt.show()
```

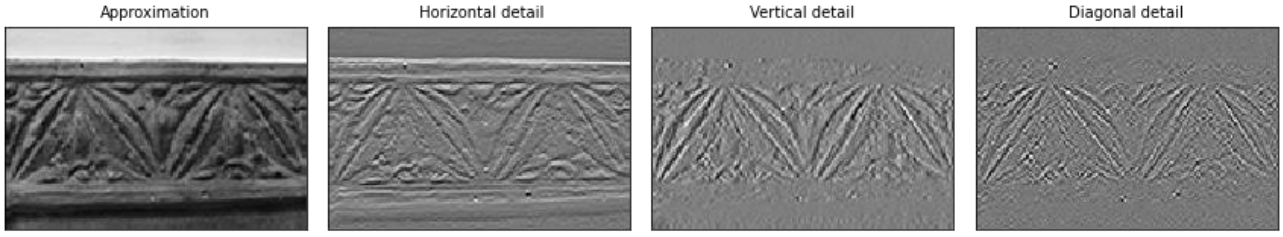
```
print("\n\nPepatraan")
index_kategori = -1;
j = 0;
for i in range(len(data_train_Pepatraan)):
    if index_kategori != label_train_Pepatraan[i]:
      index_kategori = label_train_Pepatraan[i];

      x1 = rgb2gray(data_train_Pepatraan[i]);
      gambar = pywt.dwt2(x1, 'bior1.3')
      LL, (LH, HL, HH) = gambar
      print(nama_class_pepatraan[label_train_Pepatraan[i]])
      # Wavelet transform of image, and plot approximation and details
      titles = ['Approximation', ' Horizontal detail',
                'Vertical detail', 'Diagonal detail']
      fig = plt.figure(figsize=(12, 3))
      for i, a in enumerate([LL, LH, HL, HH]):
          ax = fig.add_subplot(1, 4, i + 1)
          ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
          ax.set_title(titles[i], fontsize=10)
          ax.set_xticks([])
          ax.set_yticks([])

      fig.tight_layout()
      plt.show()

print("\n\nKekarangan")
index_kategori = -1;
j = 0;
for i in range(len(data_train_Kekarangan)):
    if index_kategori != label_train_Kekarangan[i]:
      index_kategori = label_train_Kekarangan[i];

      x1 = rgb2gray(data_train_Kekarangan[i]);
      gambar = pywt.dwt2(x1, 'bior1.3')
      LL, (LH, HL, HH) = gambar
      print(nama_class_kekarangan[label_train_Kekarangan[i]])
      # Wavelet transform of image, and plot approximation and details
      titles = ['Approximation', ' Horizontal detail',
                'Vertical detail', 'Diagonal detail']
      fig = plt.figure(figsize=(12, 3))
      for i, a in enumerate([LL, LH, HL, HH]):
          ax = fig.add_subplot(1, 4, i + 1)
          ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
          ax.set_title(titles[i], fontsize=10)
          ax.set_xticks([])
          ax.set_yticks([])

      fig.tight_layout()
      plt.show()

print("\n\nAll")
index_kategori = -1;
j = 0;
for i in range(len(data_train_head)):
    if index_kategori != label_train_head[i]:
```

```
index_kategori = label_train_head[i];

x1 = rgb2gray(data_train_head[i]);
gambar = pywt.dwt2(x1, 'bior1.3')
LL, (LH, HL, HH) = gambar
print(nama_class_head[label_train_head[i]])
# Wavelet transform of image, and plot approximation and details
titles = ['Approximation', ' Horizontal detail',
          'Vertical detail', 'Diagonal detail']
fig = plt.figure(figsize=(12, 3))
for i, a in enumerate([LL, LH, HL, HH]):
    ax = fig.add_subplot(1, 4, i + 1)
    ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
    ax.set_title(titles[i], fontsize=10)
    ax.set_xticks([])
    ax.set_yticks([])

fig.tight_layout()
plt.show()
```

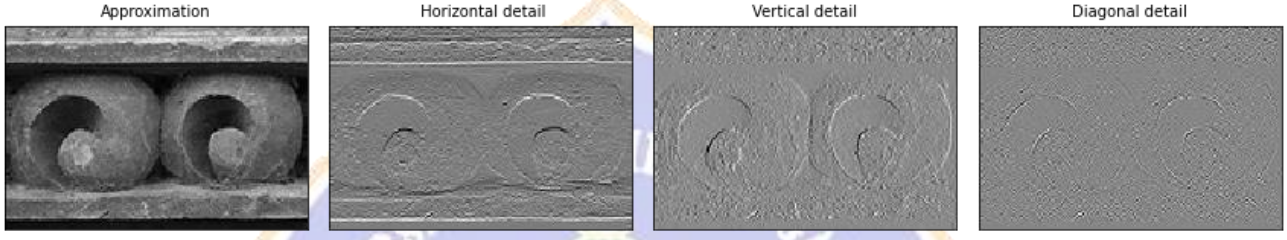
Keketusan
Batu-batuan



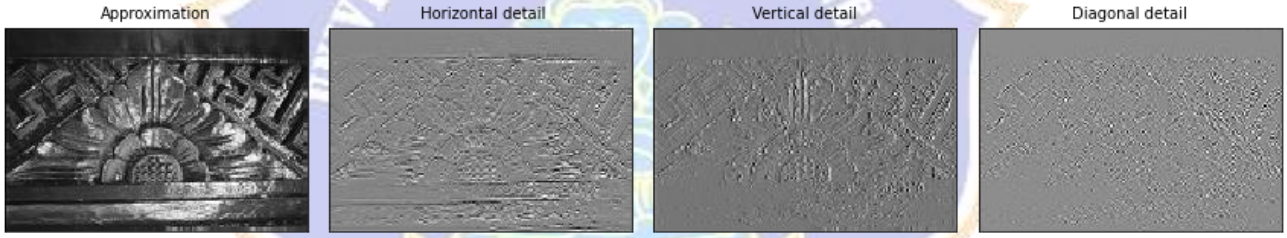
Batun Timun



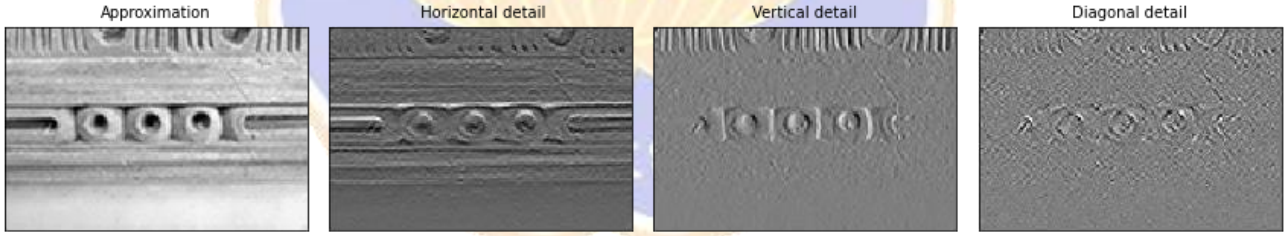
Kakul-kakulan



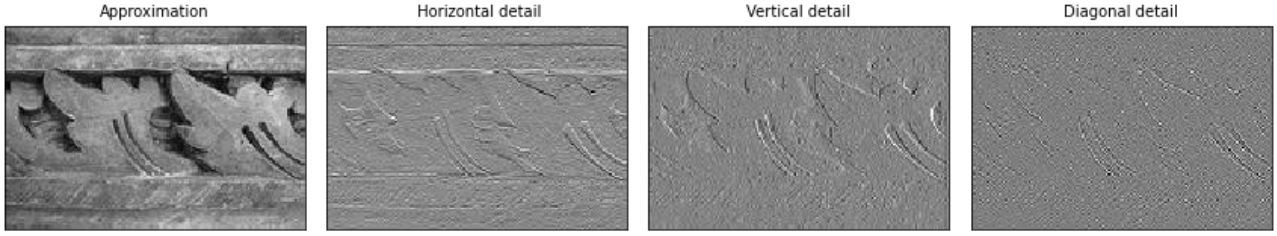
Mas-masan



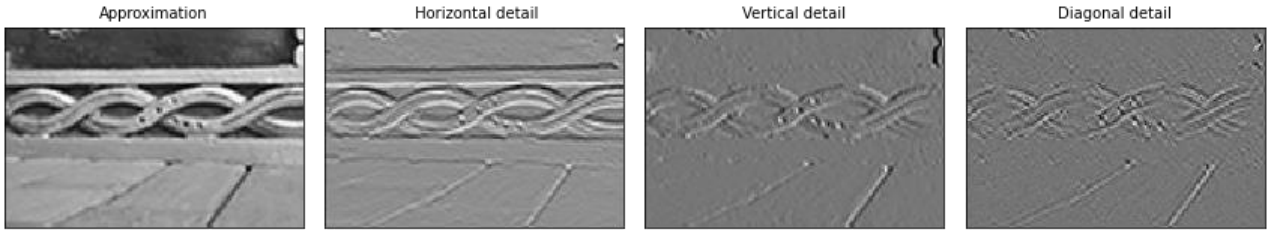
Mote-motean



Pidpid

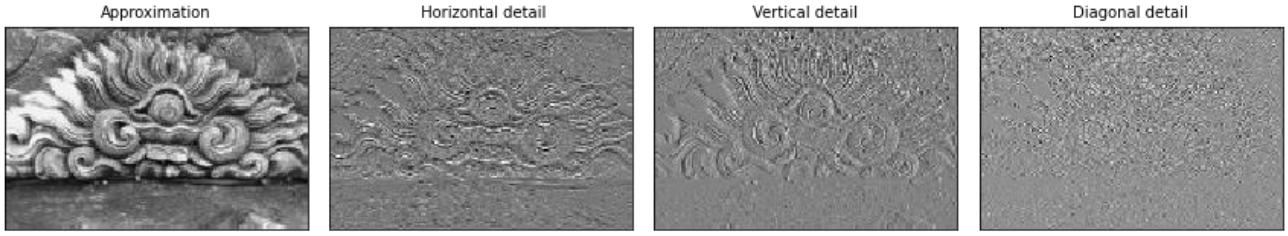


Tali ilut

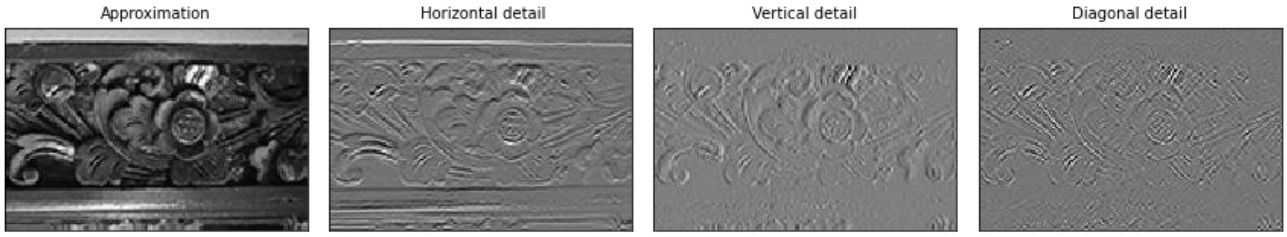


Pepatraan

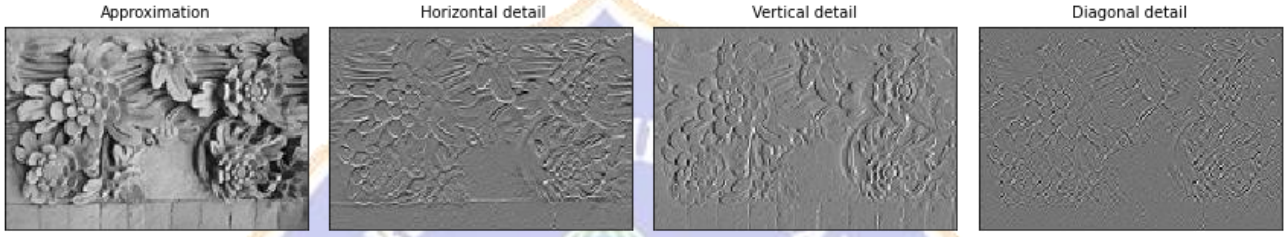
Patra Api



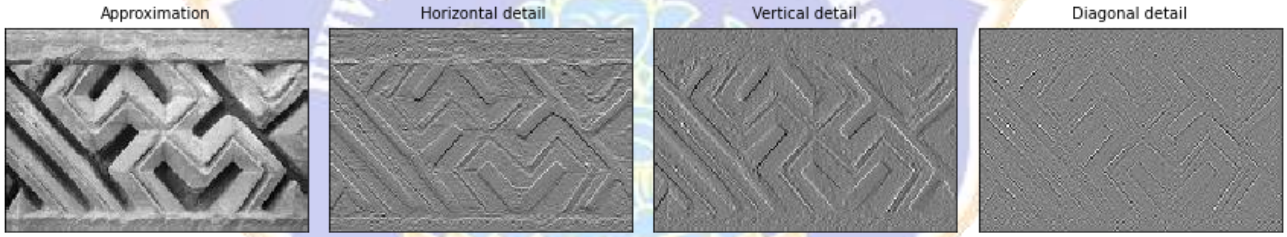
Patra Banci



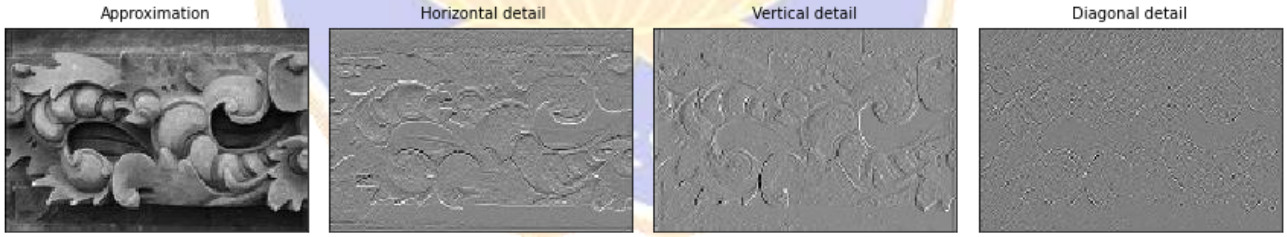
Patra Cina



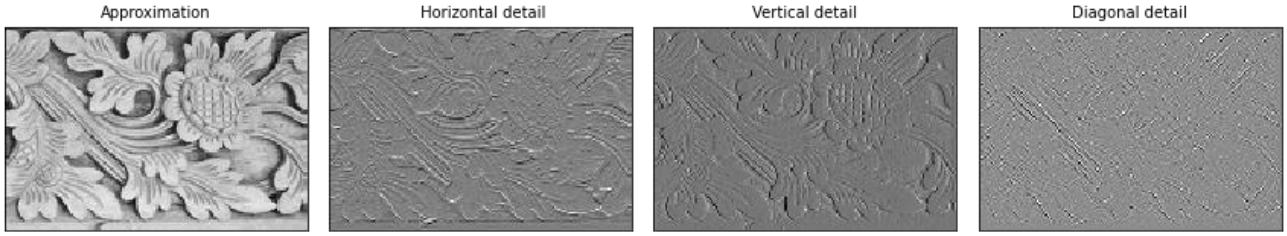
Patra Mesir



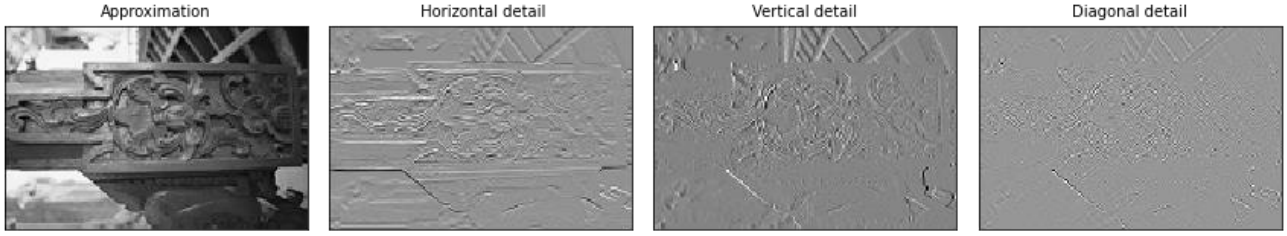
Patra Punggel



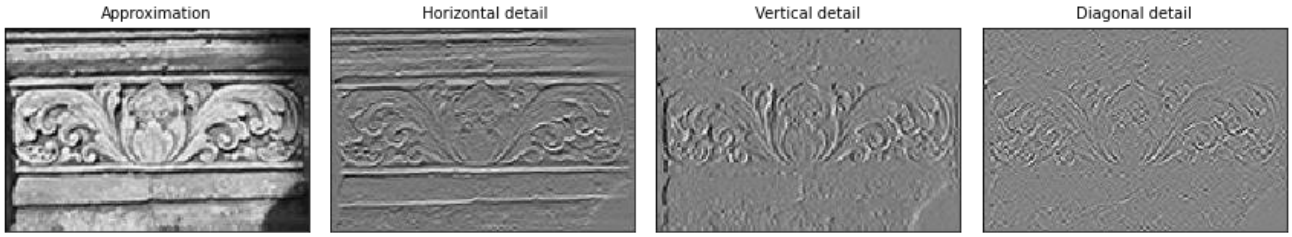
Patra Samblung



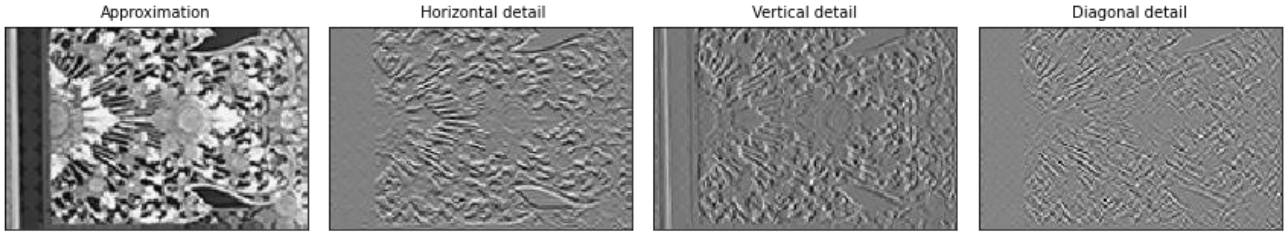
Patra Sari



Patra Ulanda



Patra Wangga

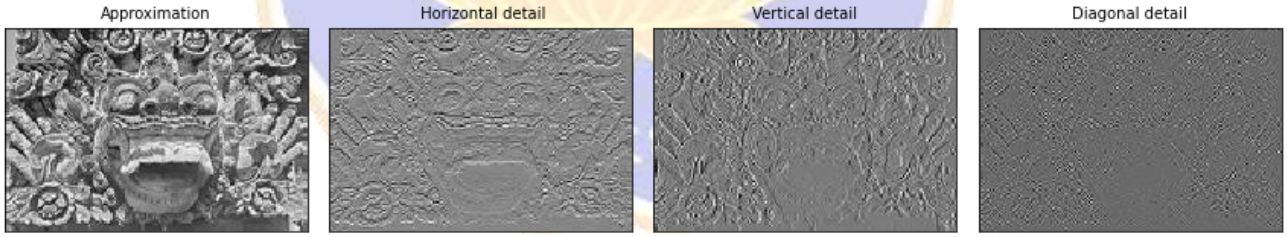


Kekarangan

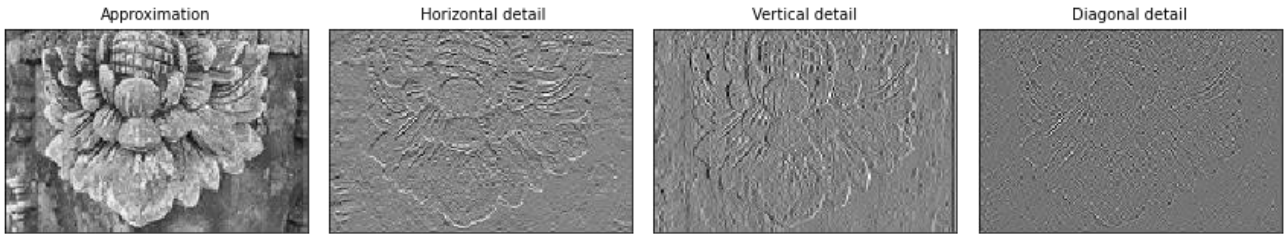
Karang Batu

Karang Bentulu

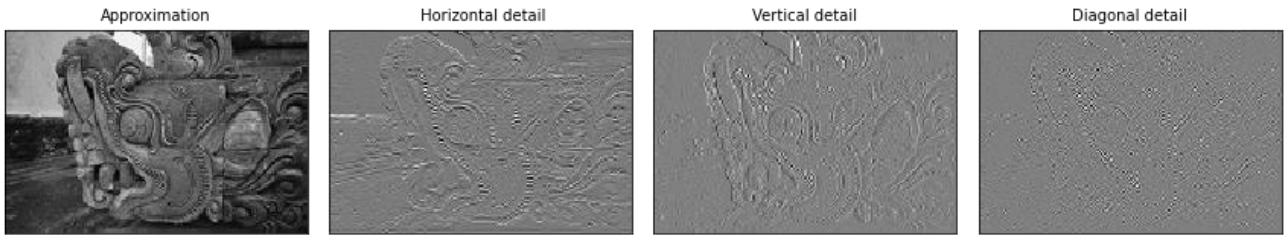
Karang Boma



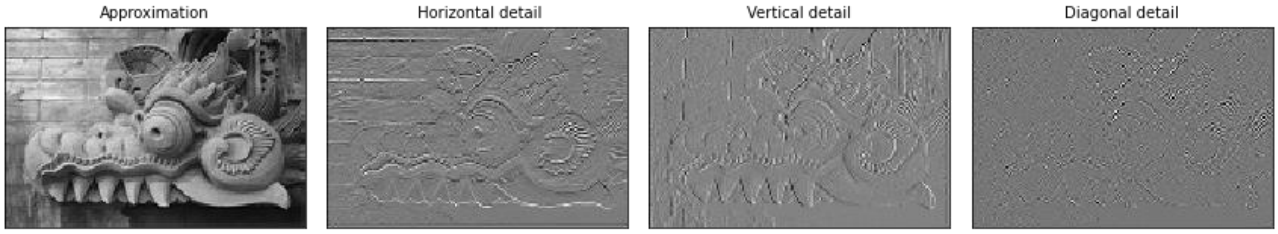
Karang Bunga



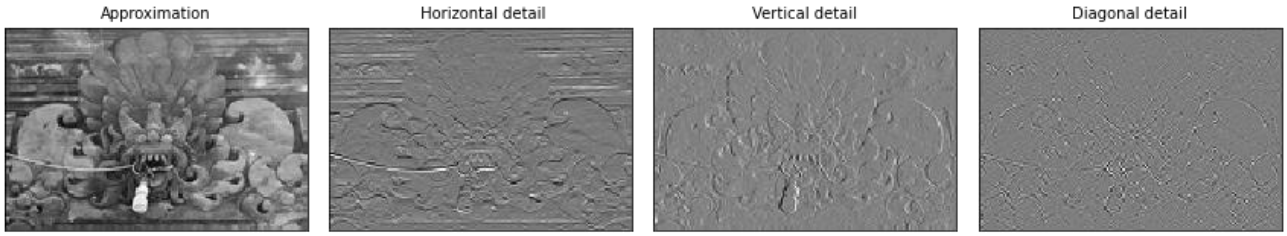
Karang Gajah



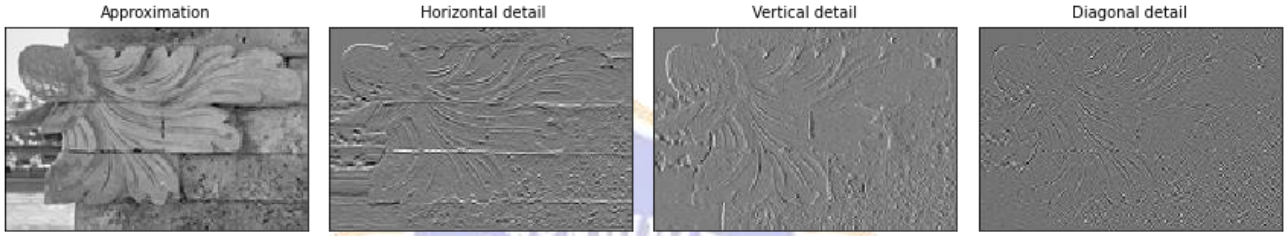
Karang Goak



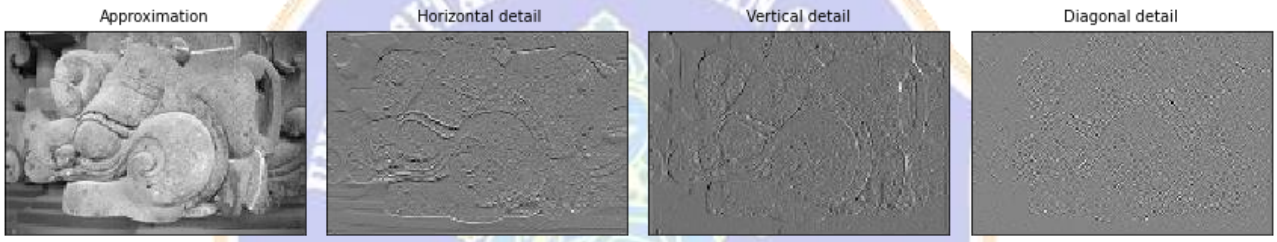
Karang Sae



Karang Simbar

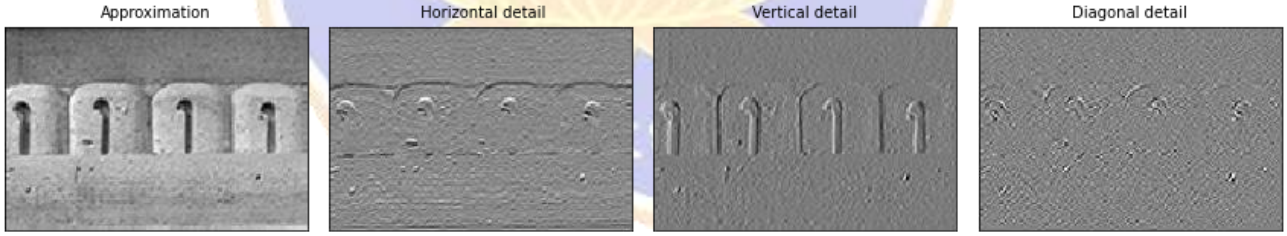


Karang Tapel

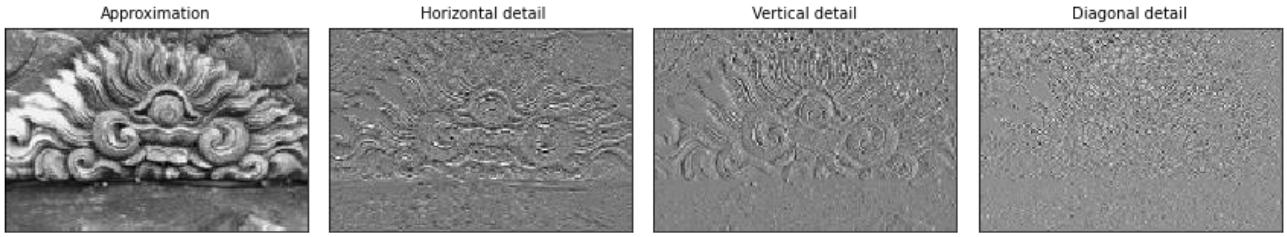


All

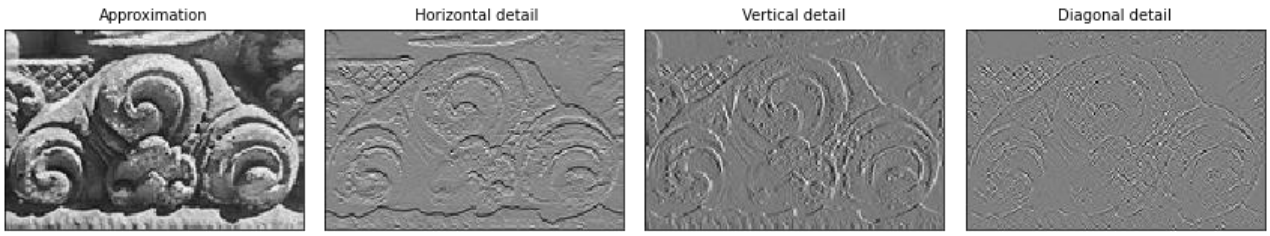
01_Keketusan



02_Pepatraan



03_Kekarangan



Mendefiniskan CNN Layer

Mendefinikan layer **dari** Convolutional Neural Network, dimana memasukkan layer yang diperlukan dalam proses pembelajaran

```
N, H, W, C = data_train_dwt_keketusan.shape;
jml_class = jumlah_class_detail[0];
model_keketusan = models.Sequential()
model_keketusan.add(layers.Conv2D(32, kernel_size=(3, 3),activation='elu',padding='same',i
model_keketusan.add(layers.LeakyReLU(alpha=0.1))
model_keketusan.add(layers.MaxPooling2D((2, 2),padding='same'))
model_keketusan.add(layers.Dropout(0.5))
model_keketusan.add(layers.Conv2D(64, (3, 3), activation='elu',padding='same'))
model_keketusan.add(layers.LeakyReLU(alpha=0.1))
model_keketusan.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_keketusan.add(layers.Dropout(0.5))
model_keketusan.add(layers.Conv2D(128, (3, 3), activation='elu',padding='same'))
model_keketusan.add(layers.LeakyReLU(alpha=0.1))
model_keketusan.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_keketusan.add(layers.Dropout(0.7))
model_keketusan.add(layers.Flatten())
# model.add(layers.Dense(128, activation='linear'))
# model.add(layers.LeakyReLU(alpha=0.1))
# model.add(layers.Dropout(0.9))
model_keketusan.add(layers.Dense(jml_class, activation='softmax'))
```

```
N, H, W, C = data_train_dwt_pepatraan.shape;
jml_class = jumlah_class_detail[1];
model_pepatraan = models.Sequential()
model_pepatraan.add(layers.Conv2D(32, kernel_size=(3, 3),activation='elu',padding='same',i
model_pepatraan.add(layers.LeakyReLU(alpha=0.1))
model_pepatraan.add(layers.MaxPooling2D((2, 2),padding='same'))
model_pepatraan.add(layers.Dropout(0.5))
model_pepatraan.add(layers.Conv2D(64, (3, 3), activation='elu',padding='same'))
model_pepatraan.add(layers.LeakyReLU(alpha=0.1))
model_pepatraan.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_pepatraan.add(layers.Dropout(0.5))
model_pepatraan.add(layers.Conv2D(128, (3, 3), activation='elu',padding='same'))
model_pepatraan.add(layers.LeakyReLU(alpha=0.1))
model_pepatraan.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_pepatraan.add(layers.Dropout(0.7))
model_pepatraan.add(layers.Flatten())
# model.add(layers.Dense(128, activation='linear'))
# model.add(layers.LeakyReLU(alpha=0.1))
# model.add(layers.Dropout(0.9))
model_pepatraan.add(layers.Dense(jml_class, activation='softmax'))


N, H, W, C = data_train_dwt_kekarangan.shape;
jml_class = jumlah_class_detail[2];
model_kekarangan = models.Sequential()
model_kekarangan.add(layers.Conv2D(32, kernel_size=(3, 3),activation='elu',padding='same',
model_kekarangan.add(layers.LeakyReLU(alpha=0.1))
model_kekarangan.add(layers.MaxPooling2D((2, 2),padding='same'))
model_kekarangan.add(layers.Dropout(0.5))
model_kekarangan.add(layers.Conv2D(64, (3, 3), activation='elu',padding='same'))
model_kekarangan.add(layers.LeakyReLU(alpha=0.1))
model_kekarangan.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_kekarangan.add(layers.Dropout(0.5))
model_kekarangan.add(layers.Conv2D(128, (3, 3), activation='elu',padding='same'))
model_kekarangan.add(layers.LeakyReLU(alpha=0.1))
model_kekarangan.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_kekarangan.add(layers.Dropout(0.7))
model_kekarangan.add(layers.Flatten())
# model.add(layers.Dense(128, activation='linear'))
# model.add(layers.LeakyReLU(alpha=0.1))
# model.add(layers.Dropout(0.9))
model_kekarangan.add(layers.Dense(jml_class, activation='softmax'))


N, H, W, C = data_train_dwt_all.shape;
jml_class = 3;
model_all = models.Sequential()
model_all.add(layers.Conv2D(32, kernel_size=(3, 3),activation='elu',padding='same',input_s
model_all.add(layers.LeakyReLU(alpha=0.1))
model_all.add(layers.MaxPooling2D((2, 2),padding='same'))
model_all.add(layers.Dropout(0.5))
model_all.add(layers.Conv2D(64, (3, 3), activation='elu',padding='same'))
model_all.add(layers.LeakyReLU(alpha=0.1))
model_all.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_all.add(layers.Dropout(0.5))
```

```
model_all.add(layers.Conv2D(128, (3, 3), activation='elu',padding='same'))
model_all.add(layers.LeakyReLU(alpha=0.1))
model_all.add(layers.MaxPooling2D(pool_size=(2, 2),padding='same'))
model_all.add(layers.Dropout(0.7))
model_all.add(layers.Flatten())
# model.add(layers.Dense(128, activation='linear'))
# model.add(layers.LeakyReLU(alpha=0.1))
# model.add(layers.Dropout(0.9))
model_all.add(layers.Dense(jml_class, activation='softmax'))
```

Proses Pelatihan

Melakukan proses pelatihan terhadap sampel data Pelatihan dan sampel data Pengujian dimana variabel yang digunakan alaha

optimasi : adam optimizer

epoch : 10

```
batchsize = 10;

model_keketusan.compile(optimizer='rmsprop',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
#label_train2 = np.array(label_train)
#label_test2 = np.array(label_test)

#history = model.fit(data_train, label_train2, epochs=10)

history_keketusan = model_keketusan.fit(data_train_dwt_keketusan, label_train_Keketusan,
                    epochs=25,
                    batch_size = batchsize,
                    shuffle=True,
                    validation_data=(data_test_dwt_keketusan, label_test_Keketusan))


model_pepatraan.compile(optimizer='rmsprop',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
#label_train2 = np.array(label_train)
#label_test2 = np.array(label_test)

#history = model.fit(data_train, label_train2, epochs=10)

history_pepatraan = model_pepatraan.fit(data_train_dwt_pepatraan, label_train_Pepatraan,
                    epochs=25,
                    batch_size = batchsize,
                    shuffle=True,
                    validation_data=(data_test_dwt_pepatraan, label_test_Pepatraan))

model_kekarangan.compile(optimizer='rmsprop',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
```

```
                metrics=['accuracy'])
#label_train2 = np.array(label_train)
#label_test2 = np.array(label_test)

#history = model.fit(data_train, label_train2, epochs=10)

history_kekarangan = model_kekarangan.fit(data_train_dwt_kekarangan, label_train_Kekaranga
                    epochs=25,
                    batch_size = batchsize,
                    shuffle=True,
                    validation_data=(data_test_dwt_kekarangan, label_test_Kekarangan))

model_all.compile(optimizer='rmsprop',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
#label_train2 = np.array(label_train)
#label_test2 = np.array(label_test)

#history = model.fit(data_train, label_train2, epochs=10)

history_all = model_all.fit(data_train_dwt_all, label_train_head,
                    epochs=25,
                    batch_size = batchsize,
                    shuffle=True,
                    validation_data=(data_test_dwt_all, label_test_head))
```

```
Epoch 1/25
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082: Us
  return dispatch_target(*args, **kwargs)
8/8 [==============================] - 5s 462ms/step - loss: 4.4053 - accuracy: 0.
Epoch 2/25
8/8 [==============================] - 3s 435ms/step - loss: 1.9428 - accuracy: 0.
Epoch 3/25
8/8 [==============================] - 3s 436ms/step - loss: 1.8902 - accuracy: 0.
Epoch 4/25
8/8 [==============================] - 5s 710ms/step - loss: 1.9680 - accuracy: 0.
Epoch 5/25
8/8 [==============================] - 3s 437ms/step - loss: 1.8698 - accuracy: 0.
Epoch 6/25
8/8 [==============================] - 3s 438ms/step - loss: 1.9138 - accuracy: 0.
Epoch 7/25
8/8 [==============================] - 3s 438ms/step - loss: 1.8769 - accuracy: 0.
Epoch 8/25
8/8 [==============================] - 3s 436ms/step - loss: 1.8715 - accuracy: 0.
Epoch 9/25
8/8 [==============================] - 3s 441ms/step - loss: 1.8669 - accuracy: 0.
Epoch 10/25
8/8 [==============================] - 3s 438ms/step - loss: 1.7748 - accuracy: 0.
Epoch 11/25
8/8 [==============================] - 3s 438ms/step - loss: 1.8468 - accuracy: 0.
Epoch 12/25
8/8 [==============================] - 3s 441ms/step - loss: 1.7586 - accuracy: 0.
Epoch 13/25
8/8 [==============================] - 3s 437ms/step - loss: 1.7908 - accuracy: 0.
Epoch 14/25
8/8 [==============================] - 3s 439ms/step - loss: 1.6568 - accuracy: 0.
Epoch 15/25
```

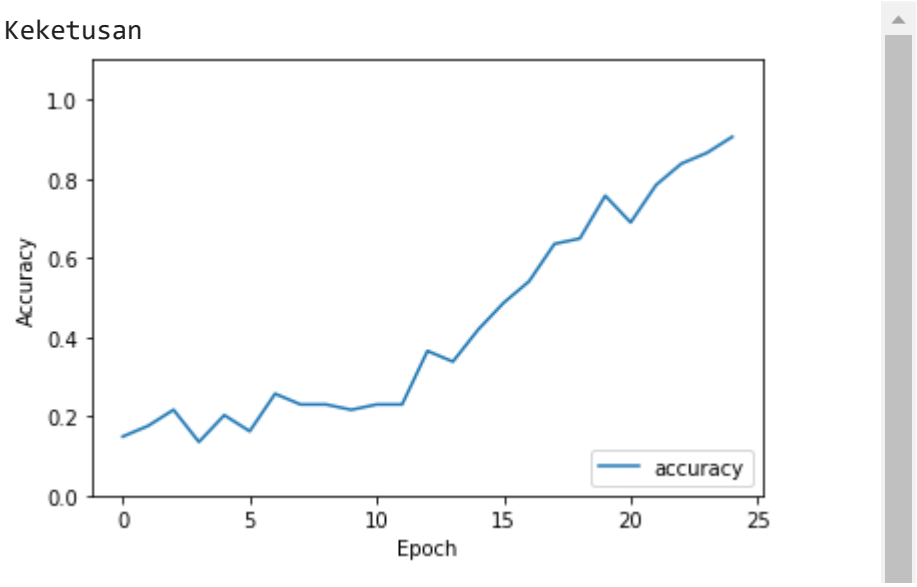
```
8/8 [==============================] - 3s 438ms/step - loss: 1.5641 - accuracy: 0.
Epoch 16/25
8/8 [==============================] - 3s 438ms/step - loss: 1.4540 - accuracy: 0.
Epoch 17/25
8/8 [==============================] - 3s 436ms/step - loss: 1.2940 - accuracy: 0.
Epoch 18/25
8/8 [==============================] - 3s 441ms/step - loss: 1.1245 - accuracy: 0.
Epoch 19/25
8/8 [==============================] - 3s 439ms/step - loss: 1.0415 - accuracy: 0.
Epoch 20/25
8/8 [==============================] - 3s 438ms/step - loss: 0.8085 - accuracy: 0.
Epoch 21/25
8/8 [==============================] - 3s 440ms/step - loss: 0.7618 - accuracy: 0.
Epoch 22/25
8/8 [==============================] - 3s 438ms/step - loss: 0.5817 - accuracy: 0.
Epoch 23/25
8/8 [==============================] - 3s 438ms/step - loss: 0.5485 - accuracy: 0.
Epoch 24/25
8/8 [==============================] - 3s 437ms/step - loss: 0.4229 - accuracy: 0.
Epoch 25/25
8/8 [==============================] - 3s 437ms/step - loss: 0.2860 - accuracy: 0.
Epoch 1/25
8/8 [==============================] - 5s 510ms/step - loss: 3.5225 - accuracy: 0.
Epoch 2/25
8/8 [==============================] - 4s 483ms/step - loss: 2.2002 - accuracy: 0.
```

Menampilkan Hasil Pelatihan dalam Grafik

Melakukan evaluasi terhadap data pengujian sehingga mengetahui tingkat akurasi dari proses pengujian yang digambarkan ke dalam sebuah grafik

```
print("Keketusan")

plt.plot(history_keketusan.history['accuracy'], label='accuracy')
#plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.0, 1.1])
plt.legend(loc='lower right')
plt.show();

print("\n\nPepatraan")
plt.plot(history_pepatraan.history['accuracy'], label='accuracy')
#plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.0, 1.1])
plt.legend(loc='lower right')
plt.show();

print("\n\nKekarangan")
plt.plot(history_kekarangan.history['accuracy'], label='accuracy')
#plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
```

```
plt.ylabel('Accuracy')
plt.ylim([0.0, 1.1])
plt.legend(loc='lower right')
plt.show();

print("\n\nAll")
plt.plot(history_all.history['accuracy'], label='accuracy')
#plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.0, 1.1])
plt.legend(loc='lower right')
plt.show();
```



Menampilkan Tingkat Akurasi Pelatihan dan Pengujian

Menampilkan hasil pelatihan dan pengujian berupa informasi tingkat akurasi dan loss.

```
print("Keketusan")
train_loss_keketusan, train_acc_keketusan = model_keketusan.evaluate(data_train_dwt_keketu
test_loss_keketusan, test_acc_keketusan = model_keketusan.evaluate(data_test_dwt_keketusan

print("----------------------------------------------------------------")
print("Akurasi Training = " + str(round(train_acc_keketusan * 100,2)))
print("Training Loss= " + str(train_loss_keketusan))
print("----------------------------------------------------------------")
print("Akurasi Testing = " + str(round(test_acc_keketusan * 100,2)))
print("Testing Loss= " + str(test_loss_keketusan))

print("\n\nPepatraan")
train_loss_pepatraan, train_acc_pepatraan = model_pepatraan.evaluate(data_train_dwt_pepatr
test_loss_pepatraan, test_acc_pepatraan = model_pepatraan.evaluate(data_test_dwt_pepatraan

print("----------------------------------------------------------------")
print("Akurasi Training = " + str(round(train_acc_pepatraan * 100,2)))
print("Training Loss= " + str(train_loss_pepatraan))
print("----------------------------------------------------------------")
print("Akurasi Testing = " + str(round(test_acc_pepatraan * 100,2)))
print("Testing Loss= " + str(test_loss_pepatraan))

print("\n\nKekarangan")
train_loss_kekarangan, train_acc_kekarangan = model_kekarangan.evaluate(data_train_dwt_kek
test_loss_kekarangan, test_acc_kekarangan = model_kekarangan.evaluate(data_test_dwt_kekara

print("----------------------------------------------------------------")
print("Akurasi Training = " + str(round(train_acc_kekarangan * 100,2)))
print("Training Loss= " + str(train_loss_kekarangan))
print("----------------------------------------------------------------")
print("Akurasi Testing = " + str(round(test_acc_kekarangan * 100,2)))
print("Testing Loss= " + str(test_loss_kekarangan))
```

```
print("\n\nAll")
train_loss_all, train_acc_all = model_all.evaluate(data_train_dwt_all, label_train_head,
test_loss_all, test_acc_all = model_all.evaluate(data_test_dwt_all,  label_test_head, batc

print("----------------------------------------------------------------")
print("Akurasi Training = " + str(round(train_acc_all * 100,2)))
print("Training Loss= " + str(train_loss_all))
print("----------------------------------------------------------------")
print("Akurasi Testing = " + str(round(test_acc_all * 100,2)))
print("Testing Loss= " + str(test_loss_all))
```

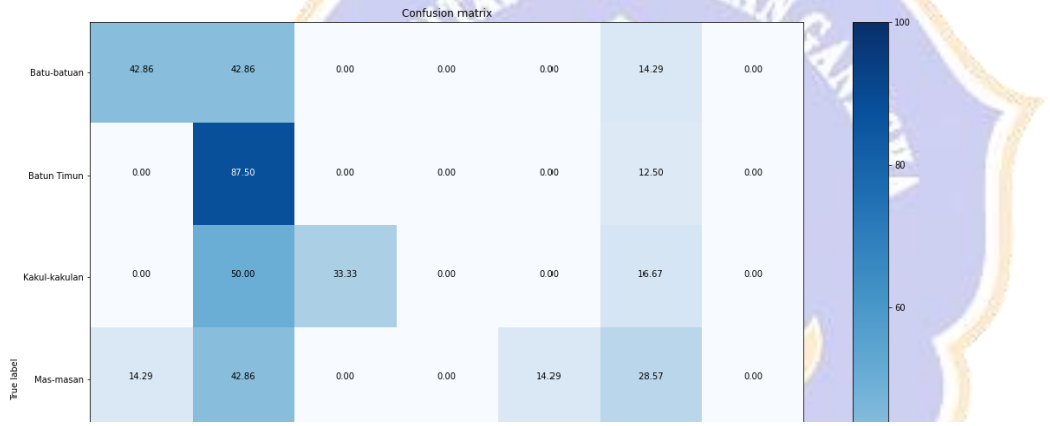
```
Keketusan
8/8 [==============================] - 1s 115ms/step - loss: 0.2771 - accuracy: 0.94
5/5 [==============================] - 1s 105ms/step - loss: 1.8348 - accuracy: 0.44
----------------------------------------------------------
Akurasi Training = 94.59
Training Loss= 0.2771463692188263
----------------------------------------------------------
Akurasi Testing = 44.19
Testing Loss= 1.8348489999771118


Pepatraan
8/8 [==============================] - 1s 123ms/step - loss: 0.0468 - accuracy: 1.00
6/6 [==============================] - 1s 104ms/step - loss: 2.1476 - accuracy: 0.48
----------------------------------------------------------
Akurasi Training = 100.0
Training Loss= 0.04676533862948418
----------------------------------------------------------
Akurasi Testing = 48.08
Testing Loss= 2.1475958824157715


Kekarangan
6/6 [==============================] - 1s 122ms/step - loss: 0.2987 - accuracy: 0.93
4/4 [==============================] - 0s 107ms/step - loss: 2.5770 - accuracy: 0.29
----------------------------------------------------------
Akurasi Training = 93.22
Training Loss= 0.29867634177207947
----------------------------------------------------------
Akurasi Testing = 29.41
Testing Loss= 2.57696533203125


All
22/22 [==============================] - 3s 119ms/step - loss: 0.0670 - accuracy: 0.
13/13 [==============================] - 2s 125ms/step - loss: 0.9144 - accuracy: 0.
------------------------------------------
Akurasi Training = 99.06
Training Loss= 0.06704495847225189
------------------------------------------
Akurasi Testing = 72.09
Testing Loss= 0.9144167900085449
```

Confusion Matrix

Hasil dari confusion matrix

```
import numpy as np
from sklearn import metrics
from sklearn.metrics import confusion_matrix
print("Keketusan")
prediksi_keketusan = model_keketusan.predict_generator(data_test_dwt_keketusan,100);
y_pred_keketusan = np.argmax(prediksi_keketusan, axis=1);

confusion_matrix_keketusan = confusion_matrix(label_test_Keketusan, y_pred_keketusan)
plot_confusion_matrix(confusion_matrix_keketusan, nama_class_keketusan, normalize=True);

print("\n\nPepatraan")
prediksi_pepatraan = model_pepatraan.predict_generator(data_test_dwt_pepatraan,100);
y_pred_pepatraan = np.argmax(prediksi_pepatraan, axis=1);

confusion_matrix_pepatraan = confusion_matrix(label_test_Pepatraan, y_pred_pepatraan)
plot_confusion_matrix(confusion_matrix_pepatraan, nama_class_pepatraan, normalize=True);

print("\n\nKekarangan")
prediksi_kekarangan = model_kekarangan.predict_generator(data_test_dwt_kekarangan,100);
y_pred_kekarangan = np.argmax(prediksi_kekarangan, axis=1);

confusion_matrix_kekarangan = confusion_matrix(label_test_Kekarangan, y_pred_kekarangan)
plot_confusion_matrix(confusion_matrix_kekarangan, nama_class_kekarangan, normalize=True);

print("\n\nAll")
prediksi_all = model_all.predict_generator(data_test_dwt_all,100);
y_pred_all= np.argmax(prediksi_all, axis=1);

confusion_matrix_all = confusion_matrix(label_test_head, y_pred_all)
plot_confusion_matrix(confusion_matrix_all, nama_class_head, normalize=True);
```

```
Keketusan
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.p
  """
WARNING:tensorflow:Your input ran out of data; interrupting
Normalized confusion matrix


Pepatraan
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.p
  if sys.path[0] == '':
WARNING:tensorflow:Your input ran out of data; interrupting
Normalized confusion matrix


Kekarangan
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.p
WARNING:tensorflow:Your input ran out of data; interrupting
Normalized confusion matrix


All
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.p
WARNING:tensorflow:Your input ran out of data; interrupting
Normalized confusion matrix
```



Keketusan - ACcuracy, Recall dan Precision

```
print("Keketusan")
from sklearn.metrics import classification_report
print(classification_report(label_test_Keketusan, y_pred_keketusan))

print("\n\nPepatraan")
from sklearn.metrics import classification_report
print(classification_report(label_test_Pepatraan, y_pred_pepatraan))

print("\n\nKekarangan")
from sklearn.metrics import classification_report
print(classification_report(label_test_Kekarangan, y_pred_kekarangan))

print("\n\nAll")
from sklearn.metrics import classification_report
print(classification_report(label_test_head, y_pred_all))
```

```
Keketusan
              precision    recall  f1-score   support
```

```
              0       0.75      0.43      0.55         7
              1       0.32      0.88      0.47         8
              2       1.00      0.33      0.50         6
              3       0.00      0.00      0.00         7
              4       0.50      0.33      0.40         6
              5       0.33      0.43      0.38         7
              6       1.00      1.00      1.00         2

    accuracy                           0.44        43
   macro avg       0.56      0.49      0.47        43
weighted avg       0.49      0.44      0.41        43



Pepatraan
              precision    recall  f1-score   support

              0       1.00      0.50      0.67         2
              1       0.33      0.43      0.38         7
              2       0.50      0.17      0.25         6
              3       0.83      0.71      0.77         7
              4       1.00      0.14      0.25         7
              5       0.67      0.57      0.62         7
              6       0.29      0.86      0.43         7
              7       0.67      0.29      0.40         7
              8       0.67      1.00      0.80         2

    accuracy                           0.48        52
   macro avg       0.66      0.52      0.51        52
weighted avg       0.63      0.48      0.47        52



Kekarangan
              precision    recall  f1-score   support

              0       1.00      0.50      0.67         2
              1       0.00      0.00      0.00         2
              2       0.00      0.00      0.00         2
              3       0.00      0.00      0.00         2
              4       0.00      0.00      0.00         8
              5       1.00      0.29      0.44         7
              6       0.00      0.00      0.00         2
              7       0.00      0.00      0.00         2
              8       0.23      1.00      0.38         7

    accuracy                           0.29        34
   macro avg       0.25      0.20      0.17        34
weighted avg       0.31      0.29      0.21        34



All
              precision    recall  f1-score   support
```

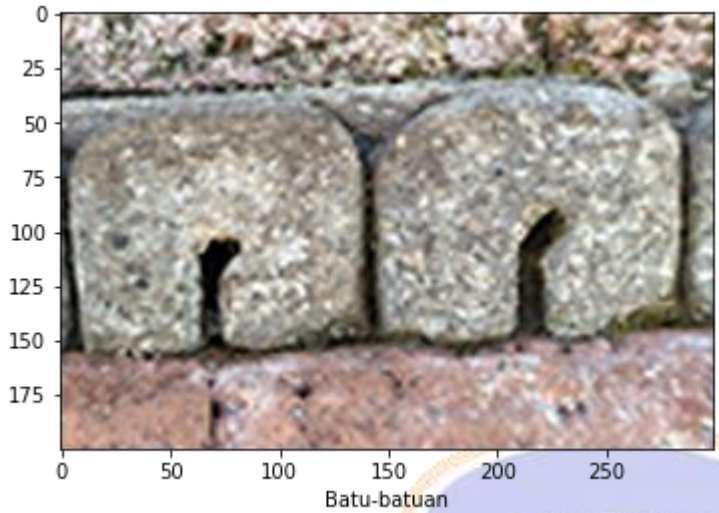
** Contoh Penggunaan Eksekusi Program **

Menentukan Class dari sebuah image yang akan dilakukan pengujian

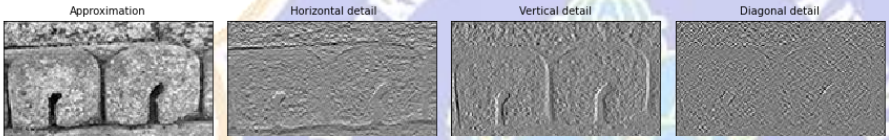
```
indeks_contoh_data = 6;
data_uji_coba = data_test_detail[indeks_contoh_data];
label_uji_coba = label_test_detail[indeks_contoh_data];
print("Tampilan Gambar yang di UJI COBA")
plt.imshow(data_uji_coba / 255.0)
plt.xlabel(nama_class_detail[label_uji_coba])
plt.show()

print("\n\nTampilan Gambar setelah dilakukan DWT")
x1 = rgb2gray(data_uji_coba);
gambar = pywt.dwt2(x1, 'bior1.3')
LL, (LH, HL, HH) = gambar;


titles = ['Approximation', ' Horizontal detail',
                'Vertical detail', 'Diagonal detail']
fig = plt.figure(figsize=(12, 3))
for i, a in enumerate([LL, LH, HL, HH]):
  ax = fig.add_subplot(1, 4, i + 1)
  ax.imshow(a, interpolation="nearest", cmap=plt.cm.gray)
  ax.set_title(titles[i], fontsize=10)
  ax.set_xticks([])
  ax.set_yticks([])

fig.tight_layout()
plt.show()

LH = np.array([LH]) /255.0
LL = np.array([LL]) /255.0
HL = np.array([HL]) /255.0
HH = np.array([HH]) /255.0
result = np.concatenate([LH, LL]);
result = np.concatenate([result, HL]);
result = np.concatenate([result, HH]);

result_dwt = np.array([result])
result_dwt = np.transpose(result_dwt, (0, 3,2,1));

print("\n\nHasil Pengenalan")
prediksi = model_all.predict(result_dwt);
class_prediksi = prediksi.argmax()
print("Class : " + nama_class_head[class_prediksi])
if(class_prediksi == 0):
  sub_prediksi = model_keketusan.predict(result_dwt);
  sub_class_prediksi = sub_prediksi.argmax();
  print("Sub Class : " + nama_class_keketusan[sub_class_prediksi])
elif(class_prediksi == 1):
  sub_prediksi = model_pepatraan.predict(result_dwt);
  sub_class_prediksi = sub_prediksi.argmax();
  print("Sub Class : " + nama_class_pepatraan[sub_class_prediksi])
elif(class_prediksi == 2):
  sub_prediksi = model_kekarangan.predict(result_dwt);
  sub_class_prediksi = sub_prediksi.argmax();
```

```
print("Sub Class : " + nama_class_kekarangan[sub_class_prediksi])
```

Tampilan Gambar yang di UJI COBA



Tampilan Gambar setelah dilakukan DWT




UNIVERSITAS PENDIDIKAN GANESHA
UNDIKSHA