TUGAS BESAR
IMAGE PROCESSING SHAPE DETECTION

Disusun Untuk Memenuhi Salah Satu Tugas Mata Kuliah Pengolahan Citra Digital Yang Diampu Oleh

Leni Fitriani, ST. M.Kom

Oleh :

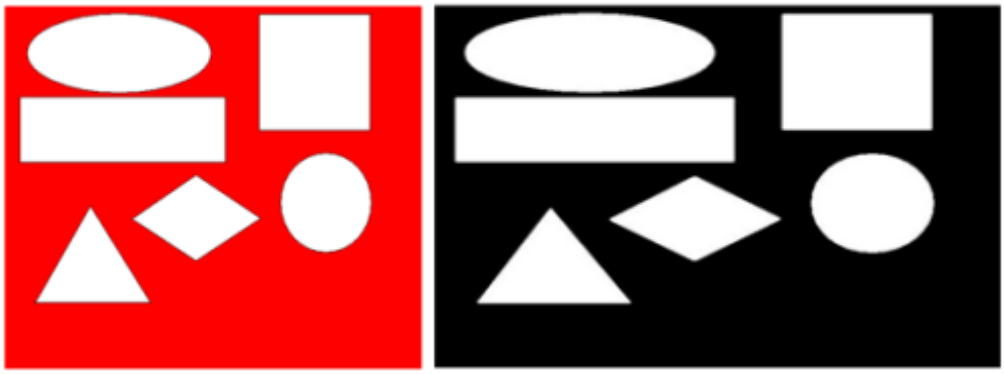
Nisrina Khaerunisa

(2006154)

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI TEKNOLOGI GARUT

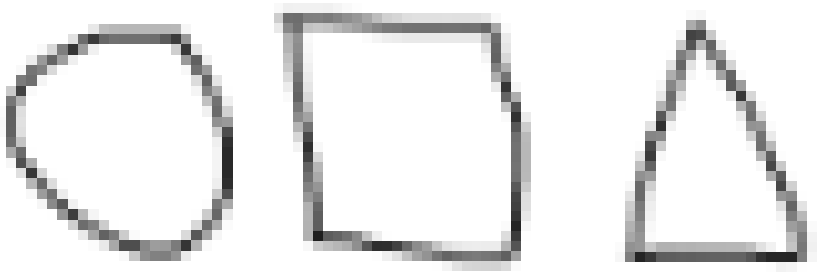
2023

Dalam project ini digunakan sumber artikel rujukan yang berjudul ‘A New Algorithm For Shape Detection”. Yang mana artikel terbitan tahun 2017 tersebut menjelaskan mengenai proses pendeteksian suatu objek sederhana seperti lingkaran, segitiga, persegi, dll dalam suatu gambar berwarna maupun pada gambar hitam putih, warna pada background foto tersebut nantinya akan mewakili bentuk atau objek yang tertera.



Gambar 1 Dataset Artikel Rujukan

Untuk dataset penulis menggunakan data publik yang diakses dari laman kaggle.com yaitu berupa gambar segitiga, lingkaran dan persegi yang keseluruhannya berjumlah kurang lebih 300 gambar. Namun perbedaanya dengan dataset yang digunakan pada artikel rujukan yaitu, pada dataset ini gambar yang ada tidak memiliki warna background sehingga background tersebut tidak mewakili bentuk atau objek yang ada digambar, namun pola yang terdapat pada gambarlah yang nantinya akan mewakili bentuk atau objek yang tertera pada gambar.



Gambar 2 Dataset Project

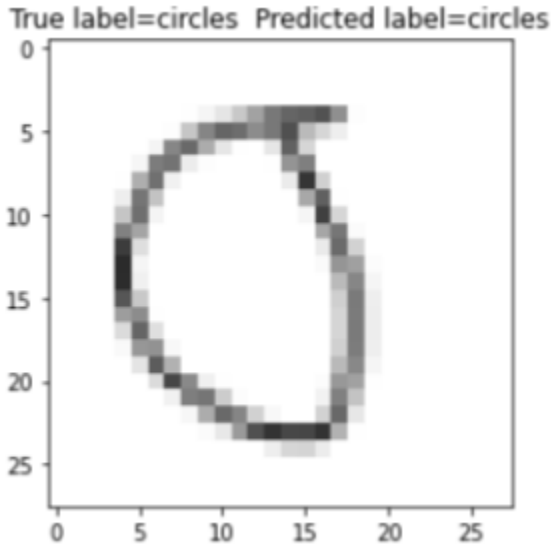
Atau dengan kata lain, proses pendeteksian bentuk/objek tersebut didapat berdasrkan pada pola garis yang terdapat dalam gambar dataset. Setelah itu digunakan model CNN untuk mengolah dataset tersebut, yang kemudian menghasilkan accurary sebesar 95%

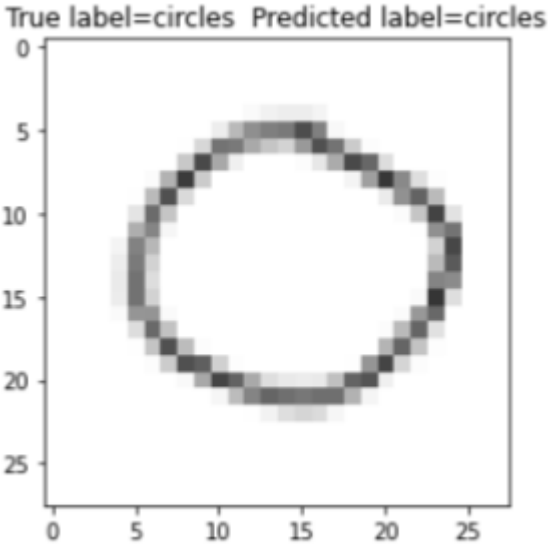
```
2/2 [==============================] - 0s 18ms/step - loss: 0.0904 - accuracy: 0.9556
[0.09042235463857651, 0.9555555582046509]
```

Gambar 3 Hasil Accuracy

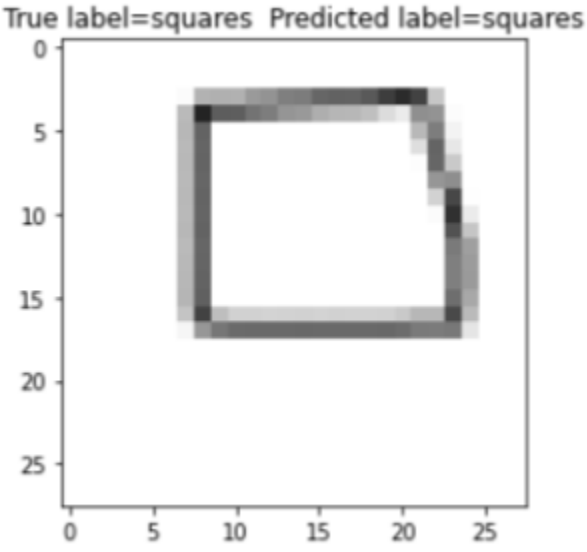
Kemudian hasil dari projek ini snediri berupa sebagai berukut :

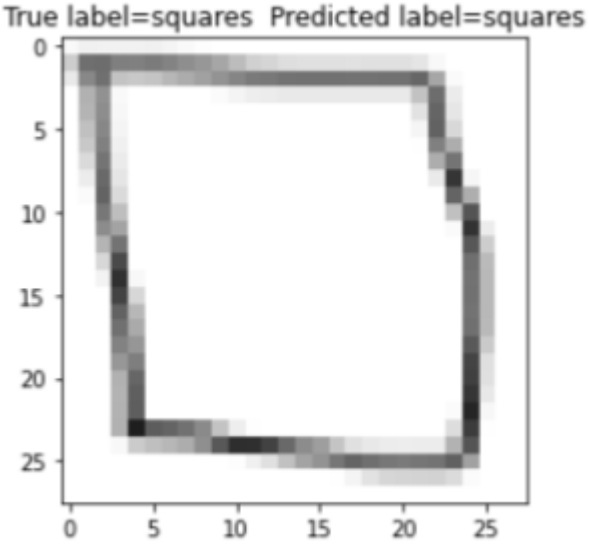
a. **Pengujian pada lingkaran**



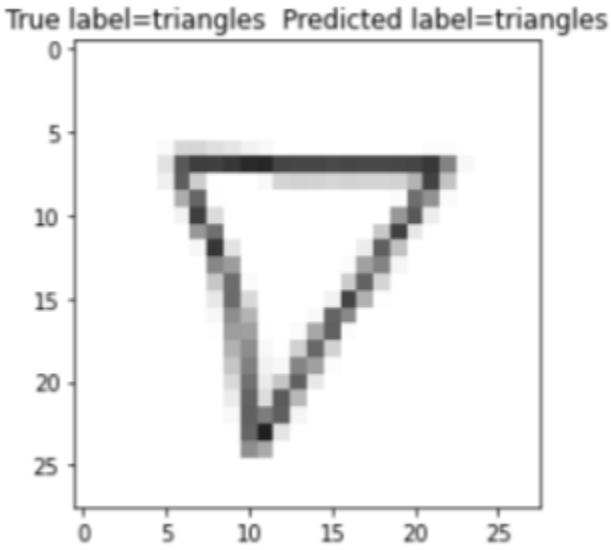


b. **Pengujian pada Persegi**





c. Pengujian pada Segitiga



Source code yang digunakan

```
# Import statements

import tensorflow as tf
import pandas as pd
import numpy as np
import cv2
import os
import shutil
from zipfile import ZipFile
import matplotlib.pyplot as plt
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns


# Split dataset into Train, Validation & Test sets

parent_folder = '../input/basicshapes/shapes/shapes'
size = 75   # Resize all images to (size,size)
bs = 32      # Batch size


# Data augmentation on train dataset only
train_data_gen = ImageDataGenerator(width_shift_range = 0.1,
height_shift_range = 0.1, zoom_range=0.1, shear_range=0.1,
brightness_range=[0.8,1.2], validation_split=0.15,
preprocessing_function=preprocess_input)
train_data = train_data_gen.flow_from_directory(parent_folder,
class_mode='categorical', target_size=(size,size), color_mode='rgb',
batch_size=bs, seed=42, subset='training')

validation_data_gen = ImageDataGenerator(validation_split=0.15,
preprocessing_function=preprocess_input)
validation_data = validation_data_gen.flow_from_directory(parent_folder,
class_mode='categorical', target_size=(size,size), color_mode='rgb',
batch_size=bs, seed=42, subset='validation')
```

```
test_data_gen = ImageDataGenerator(validation_split=0.10,
preprocessing_function=preprocess_input)
test_data = test_data_gen.flow_from_directory(parent_folder,
class_mode='categorical', target_size=(size,size), color_mode='rgb',
subset='validation', shuffle=False)
Found 255 images belonging to 3 classes.
Found 45 images belonging to 3 classes.
Found 30 images belonging to 3 classes.

# Assign essential variables

shape = train_data.image_shape
print(shape)
k = train_data.num_classes
train_samples = train_data.samples
validation_samples = validation_data.samples


# Build the model


input = Input(shape=shape)

basemodel = InceptionV3(include_top=False, weights='imagenet',
input_shape=shape, pooling='avg')
basemodel.trainable = False

x = basemodel(input)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.2)(x)
output = Dense(k, activation='softmax')(x)

model = Model(input,output)



# Compile the model

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

# Initialize callbacks

stop = EarlyStopping(monitor='val_loss', patience=4, mode='min',
restore_best_weights=True)
checkpoint = ModelCheckpoint(filepath='{val_loss:.4f}-weights-
{epoch:02d}.hdf5', monitor='val_loss', mode='min', save_best_only=True)



# Model Summary
model.summary()
```

```
# Train the model

ep = 50
spe = train_samples/bs
vs = validation_samples/bs

r = model.fit(train_data, validation_data=validation_data,
steps_per_epoch=spe, validation_steps=vs, epochs=ep,
callbacks=[stop,checkpoint])



# Evaluate the model

model.evaluate(validation_data)


# Plot training history

plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
plt.plot(r.history['accuracy'], label='accuracy')
plt.plot(r.history['val_accuracy'], label='val_accuracy')
plt.legend()
plt.show()




# Predictions on the test data

pred = model.predict(test_data).argmax(axis=1)
labels = list(train_data.class_indices.keys())


# Get the F1 score on test data prediction

print(classification_report(test_data.classes, pred))



# Plot confusion matrix

n = (test_data.num_classes)*2

cm = confusion_matrix(test_data.classes, pred)
plt.figure(figsize=(n,n))
sns.heatmap(cm, annot=True, fmt='g', xticklabels=labels, yticklabels=labels)
plt.title('Confusion Matrix')
plt.show()
```

```
# Visualize random predictions on the test data

rand = np.random.randint(low=0, high=test_data.samples, size=5)

for n in rand:
  true_index = test_data.classes[n]
  predicted_index = pred[n]
  img = cv2.imread(test_data.filepaths[n])
  img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  plt.imshow(img)
  plt.title('True label={}  Predicted label={}'.format(labels[true_index],
labels[predicted_index]))
  plt.show()
```