FINAL PROJECT : EMOTION IDENTIFICATION SYSTEM DATA CITRA DENGAN JARINGAN SYARAF TIRUAN

PENGANTAR PEMROSESAN DATA MULTIMEDIA

DISUSUN OLEH :

I Kadek Bagus Deva Diga Dana Putra	**2108561013**
I Made Wahyu Purnama Putra	**2108561018**
Valentin Gea Affila Pradika	**2108561057**

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS UDAYANA

2023

DAFTAR ISI

BAB 1 : PENDAHULUAN

1.1 Latar Belakang

Pemrosesan data citra dan pengenalan emosi menjadi topik yang semakin menarik dalam penelitian *Computer Vision* dan Kecerdasan Buatan. Identifikasi emosi dari gambar atau citra memiliki banyak aplikasi potensial, seperti dalam pengenalan emoasi manusia, deteksi ekspresi wajah, pengawasan keamanan, sampai analisis media sosial.

Jaringan Syaraf Tiruan atau *Artificial Neural Network* telah terbukti efektif dalam pemrosesan data citra dan pengenalan pola kompleks. JST merupakan algoritma klasifikasi yang meniru prinsip kerja dari jaringan syaraf manusia. Algoritma ini memetakan data masukan pada layer masukan menuju target pada layer keluaran melalui neuron-neuron pada layer tersembunyi [1]. Identifikasi emosi bahagia dan sedih dari data citra adalah tugas yang menantang. Emosi diartikan sebagai reaksi terhadap situasi tertentu yang dilakukan oleh tubuh [2]. Emosi adalah konsep abstrak yang terkait dengan ekspresi wajah, pose tubuh, dan konteks visual lainnya.

Dalam penelitian ini, implementasi algoritma JST dapat menjadi pendekatan yang efektif. Dengan melatih JST menggunakan dataset citra yang telah diberi label dengan emosi yang sesuai, jaringan dapat mempelajari pola-pola visual yang menggambarkan emosi senang atau sedih. Dalam proses ini, bobot dan parameter jaringan akan disesuaikan untuk mengoptimalkan performa jaringan dalam pengenalan emosi.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, dapat diambil rumusan masalah sebagai berikut:

1.2.1 Bagaimana algoritma Jaringan Syaraf Tiruan dapat melakukan pemrosesan data citra.

1.2.2 Bagaimana penentuan nilai parameter yang digunakan

1.2.3 Bagaimana nilai akurasi, presisi, *recall*, dan F1-*score* nya.

1.3 Tujuan

Berdasarkan rumusan masalah tersebut, maka tujuan dari pembuatan program dan laporan ini adalah:

1.3.1 Mengetahui cara algoritma Jaringan Syaraf Tiruan dapat melakukan pemrosesan citra.

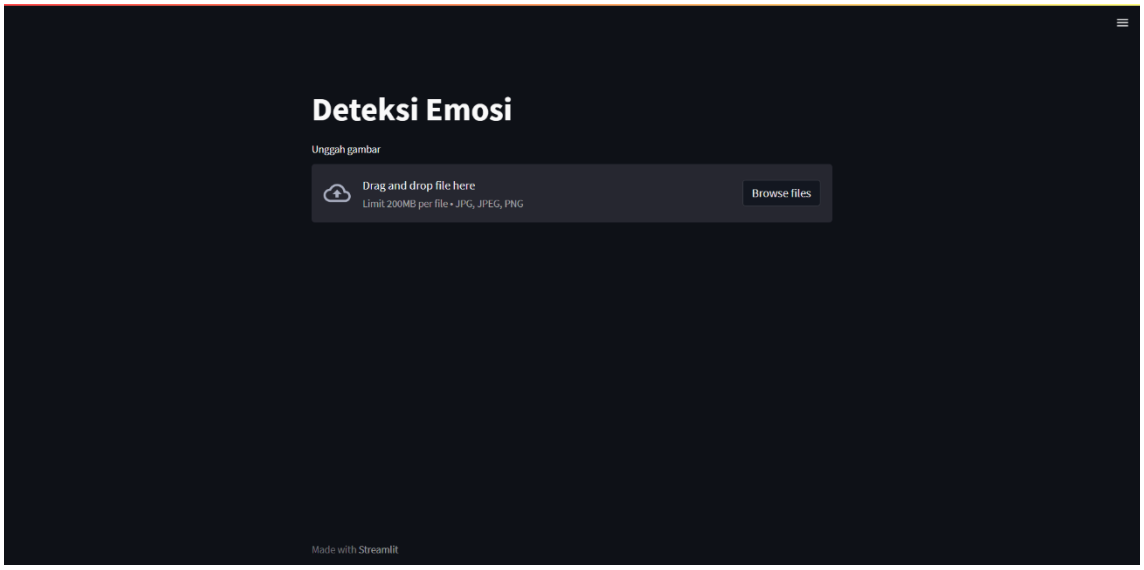
1.3.2 Mengetahui penentuan nilai parameter yang digunakan

1.3.3 Mengetahui nilai akurasi, presisi, *recall*, dan F1-*score* nya.

BAB 2 : ISI

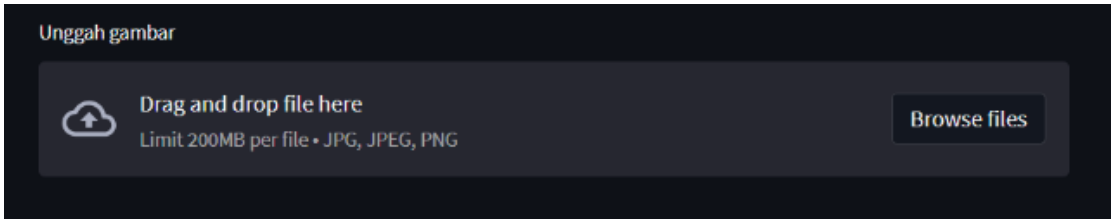
2.1 Manual Aplikasi

Program dibangun menggunakan *streamlit*. Untuk menjalankan program, dapat dilakukan dengan membuka CMD atau terminal, masuk ke folder dimana program tersimpan, lalu ketikkan "streamlit run *nama_file*" dan enter. Tampilan akan terbuka di browser utama komputer. Tampilan awal ketika membuka program akan seperti pada gambar 1 :



Gambar 1. Tampilan awal

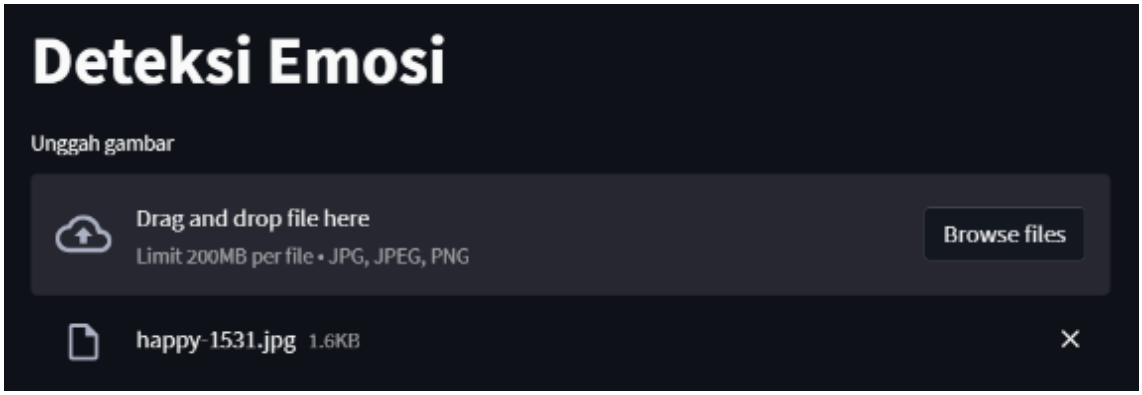
Pada bagian input, pengguna dapat memasukkan gambar yang ingin diidentifikasi emosinya dengan menekan tombol "Browse Files" dan memilih file gambar dari folder komputer atau melakukan "Drag and Drop" file gambar ke dalam kotak inputan, seperti gambar 2. Ukuran maksimal file yang dapat diterima adalah 200 MB dengan format JPG, JPEG, atau PNG.



Gambar 2. Bagian input

Setelah memasukkan gambar, program akan otomatis memulai identifikasi emosi. Nama dari file masukan akan terlihat di bawah kotak input, seperti gambar 3. Ditampilkan pula *preview* dari gambar masukan yang telah melalui tahapan preprocessing, yaitu

mengubah ukuran menjadi 48x48 pixel dan mengubah menjadi *grayscale*. Hasil identifikasi emosi berada tepat di bawah *preview* gambar dengan tulisan "Prediksi Emosi : ", seperti gambar 4.



Gambar 3. File yang diunggah

Gambar 4. Gambar yang diunggah dan hasil

2.2 Source Code

Source code lengkap dapat dilihat pada github [3]. Berikut adalah penjelasan lengkap source code dari tiap file :

2.2.1 preprocessing.ipynb

Berikut adalah source code dan penjelasan tiap shell dari file preprocessing :

1) Impor modul dan pustaka yang akan digunakan dalam program untuk analisis citra, manipulasi data, visualisasi, pembelajaran mesin, dan lain-lain

```
import matplotlib.pyplot as pls
import cv2
import os
from skimage.feature.texture import graycomatrix, graycoprops
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import precision_score, recall_score,
f1_score, accuracy_score
import joblib

pd.set_option('display.max_columns', None)
```

2) Mengatur path dan nama file gambar dalam folder "Happy" dan folder "Sad", kemudian menggabungkan daftar nama file gambar dari kedua folder tersebut. Setelah eksekusi baris ini, variabel `images_path` akan berisi daftar nama file gambar dari folder "Happy" dan folder "Sad" yang telah digabungkan

```
# Folder Happy
folder_path_happy = "data/happy"
images_file_happy = os.listdir(folder_path_happy)

# Folder Sad
folder_path_sad = "data/sad"
images_file_sad = os.listdir(folder_path_sad)

images_file_happy.remove("happy-0974.jpg") # image tidak terbaca
images_file_sad.remove("sad-0967.jpg") # image tidak terbaca
```

```
# Concate happy & sad
images_path = images_file_happy + images_file_sad
# images_path
```

3) Membaca dan mendapatkan dimensi gambar menggunakan modul `cv2` dari pustaka OpenCV. Baris-baris kode ini membaca gambar dari path yang diberikan, mendapatkan dimensi gambar, dan mencetaknya ke layar. Hasilnya adalah dimensi gambar (tinggi, lebar, dan jumlah kanal) yang ditampilkan sebagai output.

```
path = f"{folder_path_happy}/{images_file_happy[200]}"
image = cv2.imread(path)
image_height, image_width, num_channels = image.shape

print("Dimensi gambar:")
print("Tinggi:", image_height)
print("Lebar:", image_width)
print("Jumlah kanal:", num_channels)
```

Output nya :

```
Dimensi gambar:
Tinggi: 48
Lebar: 48
Jumlah kanal: 3
```

4) Menghitung fitur GLCM (Gray Level Co-occurrence Matrix) dari setiap gambar dalam `images_path`. Baris-baris kode ini membaca gambar, mengubah ukuran gambar, dan menghitung fitur GLCM dari setiap gambar dalam `images_path`. Fitur-fitur tersebut kemudian ditambahkan ke dalam daftar `features_list`.

```
import numpy as np
import cv2

def calculate_glcm_features(image):
    # Ubah gambar menjadi grayscale
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Hitung matriks GLCM
    # Jarak antar piksel untuk GLCM
    distances = [1]
    # Sudut untuk GLCM
    angles = [0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi]
```

```
    properties = ['dissimilarity', 'correlation', 'homogeneity',
                  'contrast', 'ASM', 'energy']
    glcm = graycomatrix(gray_image, distances, angles,
           symmetric=True, normed=True)

    # Ekstrak fitur dari matriks GLCM
    features = []
    for prop in properties:
        prop_values = graycoprops(glcm, prop)
        features.extend(prop_values.flatten())

    return features

# Contoh penggunaan pada setiap gambar
features_list = []


# Dimensi yang diinginkan
target_width = 48
target_height = 48

for image_path in images_path:
    try:
        if image_path[0:5] == 'happy':
            image_happy =
                 cv2.imread(f'{folder_path_happy}/{image_path}')
            # Resize gambar ke dimensi yang diinginkan
            resized_image_happy = cv2.resize(image_happy,
                                  (target_width, target_height))
            features_happy =
                     calculate_glcm_features(resized_image_happy)
            features_list.append(features_happy)
        else:
            image_sad =
                    cv2.imread(f'{folder_path_sad}/{image_path}')
            resized_image_sad = cv2.resize(image_happy,
                                (target_width, target_height))
            features_sad =
                      calculate_glcm_features(resized_image_sad)
            features_list.append(features_sad)
    except:
        print(image_path)
```

5) Mencetak atau menampilkan isi dari variabel features_list, yang berisi daftar fitur-fitur yang telah diekstraksi dari gambar-gambar menggunakan matriks GLCM.

```
features_list
```

Outputnya berupa list yang sangat panjang, berikut adalah sekilas tampilannya :

```
[[13.710992907801899,
  17.390221819827364,
  12.657801418440119,
  17.070167496604263,
  13.710992907801899,
  0.8934856868685247,
  0.8238089103496612,
  0.8951072362217792,
  0.8277037529054988,
  0.8934856868685247,
  0.12202527186103998,
  0.0934145289141387,
  0.13431353827529155,
  0.10193722212542444,
  0.12202527186103998,
  421.5496453900757,
  690.9728383883996,
  419.2003546099346,
  676.88637392484,
  421.5496453900757,
  0.0005533903287058143,
  0.00047605570937933356,
  0.0005472994001810925,
  0.0005080250122907338,
  0.0005533903287058143,
  0.0235242498011268,
  0.021818700909525608,
```

6) Mengubah daftar `features_list` menjadi array numpy (`np.array`) dan kemudian meratakan (`flatten()`) array tersebut. Dengan menggunakan `.flatten()` pada array numpy, fitur-fitur GLCM yang semula terdapat dalam array berdimensi lebih tinggi menjadi terkumpul dalam satu dimensi. Hal ini berguna jika ingin mengolah fitur-fitur tersebut menggunakan algoritma atau metode yang membutuhkan input berupa array satu dimensi.

```
features_list2 = np.array(features_list).flatten()
features_list2
```

Output :

```
array([13.71099291, 17.39022182, 12.65780142, ...,  0.02211653,
        0.02114374,  0.02218749])
```

7) Melakukan pemrosesan dan pengelompokkan fitur-fitur GLCM ke dalam kategori-kategori yang berbeda. Kode ini secara iteratif mengelompokkan fitur-fitur GLCM ke dalam kategori-kategori yang sesuai berdasarkan nilai `count`. Setelah pemrosesan, seluruh fitur akan digabungkan dalam `allFeature`, dan jumlah total fitur yang telah dikategorikan akan dihitung.

```
kolom = 5
features_list_copy = features_list2
diss = []
corr = []
homo = []
cont = []
asm = []
energ = []

count = 0
for i in range(len(features_list_copy)):
    if count < kolom:
        diss.append(features_list_copy[count])
    elif count >= kolom and count <kolom*2:
        corr.append(features_list_copy[count])
    elif count >= kolom*2 and count<kolom*3:
        homo.append(features_list_copy[count])
    elif count>=kolom*3 and count < kolom*4:
        cont.append(features_list_copy[count])
    elif count<=kolom*4 and count<kolom*5:
        asm.append(features_list_copy[count])
    else:
        energ.append(features_list_copy[count])
    count += 1
    if count == 30:
        features_list_copy = np.delete(features_list_copy,
np.s_[0:30])
        count = 0

allFeature = diss + corr + homo + cont + asm +energ
len(allFeature)
```

Outputnya : 89940

8) Mengorganisir fitur-fitur GLCM dalam struktur data yang disebut `feature`. Kode ini mengatur fitur-fitur GLCM dalam struktur data `feature` yang

mengelompokkan fitur-fitur berdasarkan angle dan nomor gambar. Struktur data ini digunakan untuk menyimpan dan mengatur fitur-fitur yang akan digunakan dalam analisis atau pemrosesan lebih lanjut.

```
feature = [[] for _ in range(14990)]

index = 14990
count = 0
count_itr = 1
flag = 5
angle = 0
idx = 0
img = 1
count_flag = 0

for i in range(len(allFeature)):
    feature[idx].append(allFeature[count])
    count += index
    if i == flag:
        feature[idx].append(angle)
        feature[idx].append(f"{img}")
        flag += 6
        count_flag += 1
        angle += 45
        count = count_itr
        count_itr += 1
        idx += 1
        if count_flag%5 == 0:
            angle = 0
            img += 1
feature
```

Outputnya berupa list yang sangat panjang, berikut adalah sekilas tampilannya :

```
[[13.710992907801899,
  0.8934856868685247,
  0.12202527186103998,
  421.5496453900757,
  0.0005533903287058143,
  0.020025455670348802,
  0,
  '1'],
 [17.390221819827364,
  0.8238089103496612,
  0.0934145289141387,
  690.9728383883996,
  0.0005596777387958489,
  0.01923814337788586,
  45,
  '1'],
 [12.657801418440119,
  0.8951072362217792,
  0.13431353827529155,
  419.2003546099346,
  0.0006266779525678256,
  0.02130889244986203,
  90,
  '1'],
```

9) Mengubah struktur data `feature` menjadi DataFrame menggunakan modul `pandas`. Kemudian, DataFrame tersebut diubah menjadi tabel pivot dengan menggunakan metode `pivot_table()` dari `pandas`. Hasilnya adalah tabel pivot yang mengorganisir fitur-fitur GLCM berdasarkan angka dan gambar. `pivot_table_df.head()` digunakan untuk mencetak lima baris pertama dari tabel pivot yang telah dibentuk.

```
df = pd.DataFrame(feature, columns=['dissimilarity',
     'correlation', 'homogeneity', 'contrast', 'ASM', 'energy',
     'angles', 'images'])

pivot_table_df = pd.pivot_table(df,
                index='images',columns='angles',
                values=['dissimilarity', 'correlation',
                'homogeneity', 'contrast', 'ASM', 'energy'])
pivot_table_df.head()
```

10) Melakukan beberapa operasi pada suatu DataFrame yang disebut pivot_table_df. Kode ini menghasilkan DataFrame baru yang telah di-reset

indeks dan kolom 'images' telah dihapus, kemudian mencetak atau menampilkan DataFrame tersebut.

```
flatenned_table = pivot_table_df.reset_index().drop('images',
                  axis=1)
flatenned_table
```

11) Melakukan perhitungan statistik deskriptif pada DataFrame flatenned_table menggunakan metode .describe(). Kode ini menghasilkan ringkasan statistik deskriptif dari DataFrame flatenned_table, memberikan informasi penting tentang distribusi dan karakteristik data dalam kolom-kolom numerik dalam DataFrame tersebut.

```
flatenned_table.describe()
```

12) Melakukan normalisasi pada fitur-fitur dalam DataFrame flatenned_table menggunakan metode Min-Max Scaling. Kode ini melakukan normalisasi fitur-fitur dalam DataFrame flatenned_table_transform menggunakan metode Min-Max Scaling, dan hasilnya disimpan dalam X_scaled. Normalisasi ini berguna untuk memastikan bahwa skala nilai fitur-fitur seragam, sehingga memudahkan dalam analisis dan pemodelan data.

```
# Normalisasi fitur
flatenned_table_transform = flatenned_table.copy()
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(flatenned_table_transform)
#Seluruh data
```

13) Menghasilkan histogram untuk setiap fitur dalam DataFrame yang telah dinormalisasi (X_scaled). Kode ini menghasilkan tampilan histogram untuk setiap fitur dalam DataFrame X_scaled, dengan masing-masing histogram ditampilkan dalam subplot pada grid 6x5. Hal ini membantu untuk memvisualisasikan distribusi data pada setiap fitur dan memperoleh wawasan tentang karakteristik data yang dinormalisasi.

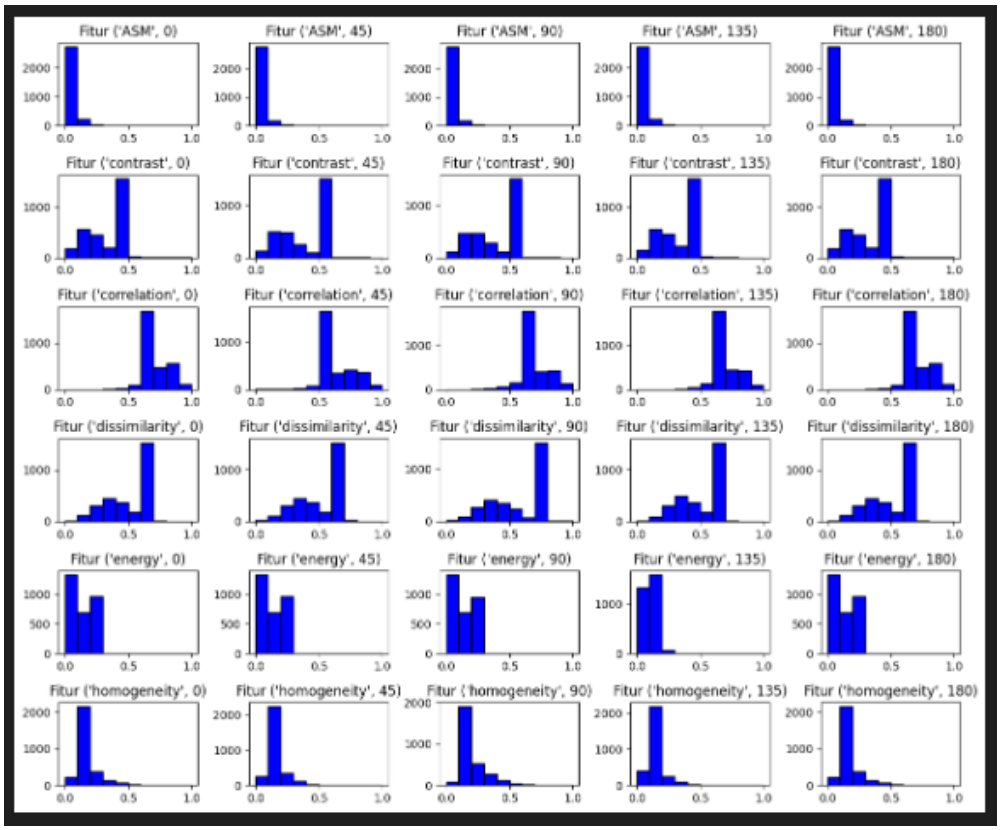
```
df = np.array(X_scaled)

fig, axs = plt.subplots(6, 5, figsize=(12,10)) #grid 6x5
count = 0
for i in range(6):
    for j in range(5):
        index_feature = i * 5 + j # indeks fitur dalam data
```

```
        axs[i, j].hist(df[:, index_feature], bins=10,
color='blue', edgecolor='black')
        axs[i, j].set_title('Fitur ' +
str(flatenned_table.columns[count]))
        count += 1

# Menampilkan histogram
plt.tight_layout()
plt.show()
```

Output :



14) Menghasilkan histogram untuk setiap fitur dalam DataFrame X_scaled setelah dilakukan transformasi akar kuadrat. Kode ini menghasilkan tampilan histogram untuk setiap fitur dalam DataFrame X_scaled setelah dilakukan transformasi akar kuadrat. Hal ini berguna untuk melihat efek transformasi tersebut terhadap distribusi data dan mungkin membantu dalam mengatasi asimetri atau pengaruh ekstrem pada data.

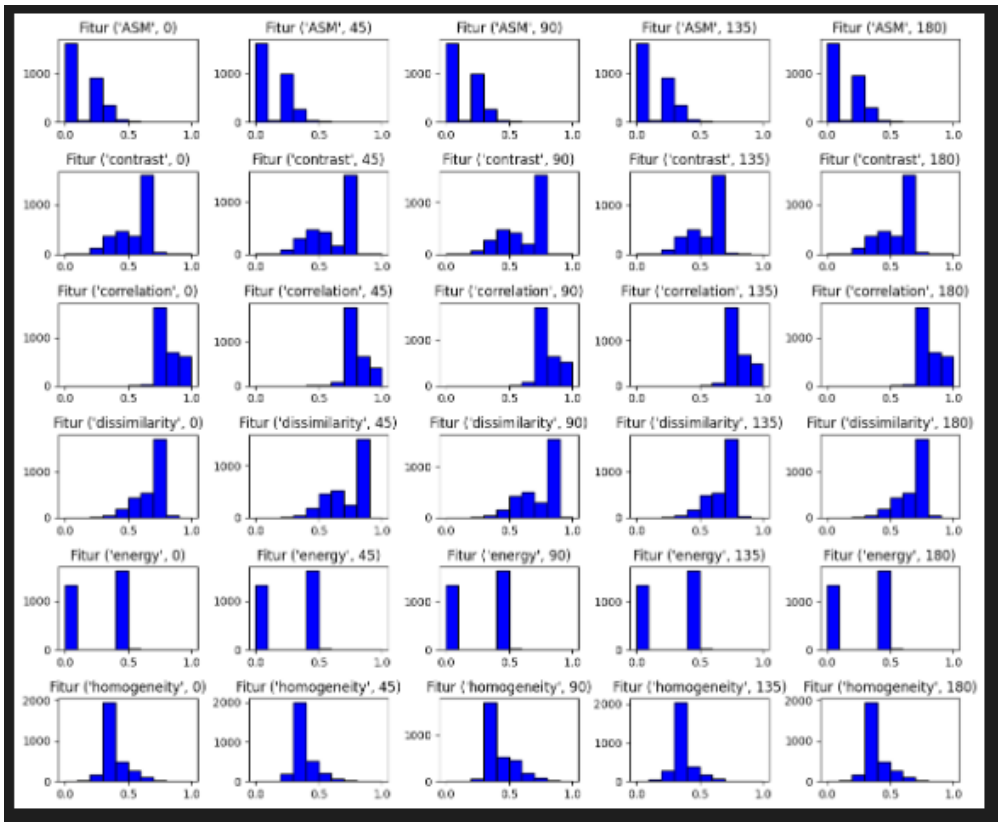
```
df = np.array(X_scaled)
df = np.sqrt(df)

fig, axs = plt.subplots(6, 5, figsize=(12,10)) #grid 6x5
```

```
count = 0
for i in range(6):
    for j in range(5):
        index_feature = i * 5 + j # indeks fitur dalam data
        axs[i, j].hist(df[:, index_feature], bins=10,
color='blue', edgecolor='black')
        axs[i, j].set_title('Fitur ' +
str(flatenned_table.columns[count]))
        count += 1

# Menampilkan histogram
plt.tight_layout()
plt.show()
```

Output :



15) Membuat DataFrame baru test menggunakan array NumPy df dan menampilkan DataFrame tersebut. Kode ini menghasilkan tampilan DataFrame baru test yang dibuat berdasarkan array NumPy df. Hal ini berguna untuk memeriksa struktur dan isi DataFrame setelah dilakukan transformasi atau operasi lainnya.

```
test = pd.DataFrame(df)
test
```

16) Melakukan persiapan data untuk pemodelan dengan menggunakan fitur-fitur GLCM yang telah dinormalisasi. Kode ini melakukan persiapan data dengan menghapus file gambar yang tidak terbaca, mengubah DataFrame fitur GLCM menjadi array NumPy, dan mempersiapkan matriks fitur (X) dan array label (y) untuk pemodelan berikutnya.

```
## Menyiapkan data
# Folder Happy
folder_path_happy = "data/happy"
images_file_happy = os.listdir(folder_path_happy)
images_file_happy.remove("happy-0974.jpg") # image tidak terbaca

# Folder Sad
folder_path_sad = "data/sad"
images_file_sad = os.listdir(folder_path_sad)
images_file_sad.remove("sad-0967.jpg") # image tidak terbaca

df = np.array(X_scaled)

X = df  # Masukkan fitur GLCM di sini
y = np.array([1] * len(images_file_happy) + [0] *
len(images_file_sad))  # Label: 1 untuk happy, 0 untuk sad
```

17) Membuat DataFrame baru a menggunakan array NumPy df dan menampilkan DataFrame tersebut. Kode ini menghasilkan tampilan DataFrame baru a yang dibuat berdasarkan array NumPy df. Hal ini berguna untuk memeriksa struktur dan isi DataFrame setelah dilakukan operasi atau transformasi sebelumnya.

```
a = pd.DataFrame(df)
a
```

18) Menggunakan modul joblib untuk menyimpan objek scaler ke dalam file dengan nama "scaler.pkl". Kode ini menyimpan objek scaler ke dalam file "scaler.pkl" menggunakan modul joblib. Hal ini berguna jika Anda ingin menggunakan kembali objek scaler tersebut di waktu yang akan datang tanpa harus melakukan normalisasi ulang atau mengatur ulang konfigurasi.

```
joblib.dump(scaler, 'scaler.pkl')
```

19) Membagi data menjadi set pelatihan (training set) dan set pengujian (testing set) menggunakan fungsi train_test_split() dari modul sklearn.model_selection. Kode ini melakukan pemisahan data menjadi set pelatihan dan set pengujian dengan membagi matriks fitur (X) dan array label

(y) menggunakan fungsi train_test_split(). Hal ini memungkinkan kita untuk melatih model pada set pelatihan dan menguji performa model pada set pengujian untuk mengevaluasi kinerjanya.

```
# Membagi data menjadi set pelatihan dan set pengujian
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

20) Membuat objek model dengan menggunakan kelas Sequential dari TensorFlow's Keras API. Kode ini hanya menginisialisasi objek model, dan tidak melakukan konfigurasi atau penambahan layer pada model tersebut. Dalam tahap selanjutnya, Anda perlu menambahkan layer-layer neural network ke dalam model untuk membangun arsitektur model yang spesifik.

```
# Membangun model
model = tf.keras.models.Sequential()
```

21) Kombinasi 1 dari Tuning Hyper Parameter

```
# Menentukan kombinasi hyperparameter
## Kombinasi 1
learning_rate = 0.005
epochs = 50
num_hidden_layers = 2
hidden_neurons = [64, 32]  # Jumlah neuron pada setiap hidden
layer
activation_functions = ['relu', 'relu']  # Fungsi aktivasi untuk
setiap hidden layer

# Membangun model
model = tf.keras.models.Sequential()

# Hidden layers
for i in range(num_hidden_layers):
    model.add(tf.keras.layers.Dense(hidden_neurons[i],
activation=activation_functions[i]))

# Output layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Mengompilasi model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=le
arning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
# Melatih model
model.fit(X_train, y_train, epochs=epochs, batch_size=32,
verbose=1)

# Evaluasi model pada set pengujian
_, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Accuracy:', accuracy)
```

22) Evaluasi kombinasi 1 Hyper Parameter

```
# Evaluasi model pada set pengujian
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)  # Konversi prediksi
menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-Score:', f1)
```

23) Kombinasi 2 Tuning Hyper Parameter

```
# Menentukan kombinasi hyperparameter
## Kombinasi 2
learning_rate = 0.001
epochs = 50
num_hidden_layers = 3
hidden_neurons = [64, 32, 16]  # Jumlah neuron pada setiap hidden
layer
activation_functions = ['relu', 'relu', 'relu']  # Fungsi
aktivasi untuk setiap hidden layer

# Membangun model
model = tf.keras.models.Sequential()

# Hidden layers
for i in range(num_hidden_layers):
    model.add(tf.keras.layers.Dense(hidden_neurons[i],
activation=activation_functions[i]))

# Output layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Mengompilasi model
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=le
arning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Melatih model
model.fit(X_train, y_train, epochs=epochs, batch_size=32,
verbose=1)

# Evaluasi model pada set pengujian
_, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Accuracy:', accuracy)
```

24) Evaluasi kombinasi 2 Hyper Parameter

```
# Evaluasi model pada set pengujian
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)  # Konversi prediksi
menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-Score:', f1)
```

25) Kombinasi 3 Tuning Hyper Parameter

```
# Menentukan kombinasi hyperparameter
## Kombinasi 3
learning_rate = 0.01
epochs = 50
num_hidden_layers = 2
hidden_neurons = [64, 32]  # Jumlah neuron pada setiap hidden
layer
activation_functions = ['relu', 'relu']  # Fungsi aktivasi untuk
setiap hidden layer

# Membangun model
model = tf.keras.models.Sequential()

# Hidden layers
for i in range(num_hidden_layers):
    model.add(tf.keras.layers.Dense(hidden_neurons[i],
activation=activation_functions[i]))
```

```
# Output layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Mengompilasi model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=le
arning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Melatih model
model.fit(X_train, y_train, epochs=epochs, batch_size=32,
verbose=1)

# Evaluasi model pada set pengujian
_, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Accuracy:', accuracy)
```

26) Evaluasi kombinasi 2 Hyper Parameter

```
# Evaluasi model pada set pengujian
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)  # Konversi prediksi
menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-Score:', f1)
```

27) Melakukan prediksi menggunakan model neural network pada data pengujian (X_test) untuk satu data input dan menyimpan hasil prediksi tersebut dalam variabel a. Hal ini berguna untuk melihat hasil prediksi dari model terhadap satu data input tertentu.

```
a = model.predict(X_test[0:1])
a
```

28) Melakukan evaluasi model pada set pengujian (X_test) dengan menghitung beberapa metrik evaluasi, yaitu akurasi (accuracy), presisi (precision), recall, dan nilai F1 (F1-score).

```
# Evaluasi model pada set pengujian
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)  # Konversi prediksi
menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-Score:', f1)
```

29) Pemodelan

```
# Membangun model
model = tf.keras.models.Sequential()
```

30) Penentuan hyperparameter

```
# Menentukan hyperparameter
learning_rate = 0.005
epochs = 50
num_hidden_layers = 2
hidden_neurons = [64, 32]  # Jumlah neuron pada setiap hidden
layer
activation_functions = ['relu', 'relu']  # Fungsi aktivasi untuk
setiap hidden layer

# Membangun model
model = tf.keras.models.Sequential()

# Hidden layers
for i in range(num_hidden_layers):
    model.add(tf.keras.layers.Dense(hidden_neurons[i],
activation=activation_functions[i]))

# Output layer
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

# Mengompilasi model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=le
arning_rate),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Melatih model
```

```
model.fit(X_train, y_train, epochs=epochs, batch_size=32,
verbose=1)

# Evaluasi model pada set pengujian
_, accuracy = model.evaluate(X_test, y_test, verbose=0)
print('Accuracy:', accuracy)
```

31) Evaluasi model

```
# Evaluasi model pada set pengujian
y_pred = model.predict(X_test)
y_pred_binary = (y_pred > 0.5).astype(int)  # Konversi prediksi
menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y_test, y_pred_binary)
precision = precision_score(y_test, y_pred_binary)
recall = recall_score(y_test, y_pred_binary)
f1 = f1_score(y_test, y_pred_binary)

print('Accuracy:', accuracy)
print('Precision:', precision)
print('Recall:', recall)
print('F1-Score:', f1)
```

32) Mencoba model pada masing-masing data

```
data_happy_sad = pd.DataFrame(df)
```

33) Evaluasi pada data happy

```
y_pred_happy = model.predict(happy)
y_pred_happy_binary = (y_pred_happy > 0.55).astype(int)  #
Konversi prediksi menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y[0:1536], y_pred_happy_binary)
precision = precision_score(y[0:1536], y_pred_happy_binary)
recall = recall_score(y[0:1536], y_pred_happy_binary)
f1 = f1_score(y[0:1536], y_pred_happy_binary)

print('All happy Accuracy:', accuracy)
print('All happy Precision:', precision)
print('All happy Recall:', recall)
print('All happy F1-Score:', f1)
```

34) Evaluasi pada data sad

```
y_pred_sad = model.predict(sad)
y_pred_sad_binary = (y_pred_sad > 0.55).astype(int)  # Konversi
prediksi menjadi nilai biner (0 atau 1)

accuracy = accuracy_score(y[1536::], y_pred_sad_binary)
```

```
precision = precision_score(y[1536::], y_pred_sad_binary)
recall = recall_score(y[1536::], y_pred_sad_binary)
f1 = f1_score(y[1536::], y_pred_sad_binary)

print('All sad Accuracy:', accuracy)
print('All sad Precision:', precision)
print('All sad Recall:', recall)
print('All sad F1-Score:', f1)
```

35) Menyimpan model

```
model.save('model.h5')
```

36) Menampilkan ringkasan darii arsitektur model

```
model.summary()
```

2.2.2 app.py

Gambar merupakan *source code* yang digunakan untuk tampilan antarmuka dari program yang dibuat. Dengan menggunakan *streamlit* program dapat berjalan di web. Pada saat menjalankan program, model dan scaler (untuk normalisasi fitur) dimuat ke dalam memori. Ketika gambar diunggah, menggunakan OpenCV gambar tersebut diubah ukurannya menjadi 48x48 pixel. Fitur GLCM (*Gray-Level Co-occurrence Matrix*) diekstraksi dari gambar yang telah diubah ukurannya yang mencerminkan tekstur gambar. Fitur-fitur GLCM kemudian dihitung dan dinormalisasi menggunakan scaler yang telah dilatih sebelumnya. Setelah itu, model melakukan prediksi menggunakan fitur-fitur yang diberikan, dan hasil prediksi ditampilkan di halaman web kepada pengguna.

```
import streamlit as st
import cv2
import numpy as np
from skimage.feature.texture import graycomatrix, graycoprops
from sklearn.preprocessing import StandardScaler
import tensorflow as tf
import joblib


# Fungsi untuk menghitung fitur GLCM
def calculate_glcm_features(image):
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    distances = [1]
    angles = [0, np.pi / 4, np.pi / 2, 3 * np.pi / 4, np.pi]
    properties = [
        "dissimilarity",
        "correlation",
        "homogeneity",
```

```
        "contrast",
        "ASM",
        "energy",
    ]
    glcm = graycomatrix(gray_image, distances, angles, symmetric=True,
           normed=True)
    features = []
    for prop in properties:
        prop_values = graycoprops(glcm, prop)
        features.extend(prop_values.flatten())
    return features


# Fungsi untuk memuat model dan melakukan prediksi
def predict_emotion(features):
    model = tf.keras.models.load_model("model.h5")
    scaler = joblib.load("scaler.pkl")
    scaled_features = scaler.transform(features)
    predictions = model.predict(scaled_features)
    return predictions


# Judul halaman web
st.title("Deteksi Emosi")

# Upload gambar
uploaded_file = st.file_uploader("Unggah gambar", type=["jpg", "jpeg",
                "png"])

if uploaded_file is not None:
    # Baca gambar yang diunggah
    image = cv2.imdecode(np.fromstring(uploaded_file.read(), np.uint8),
            1)
    # Resize gambar
    target_width = 48
    target_height = 48
    resized_image = cv2.resize(image, (target_width, target_height))
    # Hitung fitur GLCM
    features = calculate_glcm_features(resized_image)
    # Prediksi emosi
    predictions = predict_emotion([features])
    emotion = "Happy" if predictions[0] > 0.5 else "Sad"

    # Tampilkan gambar dan hasil prediksi
    st.image(image, caption="Gambar yang diunggah",
             use_column_width=True)
    st.write("Prediksi Emosi:", emotion)
```

2.3 Evaluasi Sistem

Program ini dibuat dengan menggunakan 3 (tiga) kombinasi parameter untuk mencari model terbaik seperti yang ada pada source code. Tabel 1 menunjukkan tuning hyperparameter yang digunakan.

Tabel 1. Tuning hyperparameter

	Learning Rate	**Epoch**	**Jumlah *hidden layer***	**Jumlah *hidden neuron***	**Fungsi aktivasi**	**Akurasi**
Kombinasi 1	0.005	50	2	64, 32	Relu, Relu	0.62166…
Kombinasi 2	0.001	50	3	64, 32, 16	Relu, Relu, relu	0.58999…
Kombinasi 3	0.01	50	2	64, 32	Relu, Relu	0.60000…

Dari tabel 1 dapat diambil kesimpulan bahwa kombinasi pertama mendapatkan hasil akurasi terbaik. Karena itu, digunakan nilai hyperparameter pada kombinasi 1 yang akan seterusnya digunakan dalam model. Berikut adalah matrik evaluasi lengkap dari kombinasi 1 :

Tabel 2. Matrik evaluasi kombinasi 1

Accuracy	***Precision***	***Recall***	***F1-Score***
0. 633333333333..	0. 609071274298..	0. 878504672897..	0. 719387755102..

Selain matrik evaluasi dari seluruh data, perhitungan matrik untuk masing-masing data juga perlu dihitung untuk mengetahui apakah terdapat ketimpangan akurasi antara satu data dengan data lainnya. Tabel 3 menunjukkan matrik evaluasi dari data *happy* dan *sad*.

Tabel 3. Matrik Evaluasi per data

	Accuracy	***Precision***	***Recall***	***F1-Score***
Data *Happy*	0.7884114583..	1.0	0.7884114583..	0.8816891153..
Data *Sad*	0.4090287277..	0.0	0.0	0.0

BAB 3 : PENUTUP

3.1 Kesimpulan

Dalam laporan ini, telah dibangun sebuah sistem untuk mengidentifikasi sentimen atau emosi dari sebuah citra atau gambar. Algoritma yang digunakan adalah Jaringan Saraf Tiruan (JST). Tujuan akhir dari pembuatan program sistem ini adalah untuk menerapkan metode JST dalam melakukan pemrosesan sebuah citra untuk diidentifikasi sentimen atau emosi dari citra tersebut dengan pembagian emosi bahagia (Happy) dan sedih (sad). Sistem melakukan normalisasi fitur terlebih dahulu yang akan digunakan sebagai input pada model. Setelah model jadi, akan masuk ke bagian preprocessing dan terakhir menampilkan hasilnya.

Dalam pengujian yang dilakukan, sistem mampu melakukan klasifikasi emosi terbaik dengan nilai akurasi 63.3%, presisi 60.9%, *recall* 87.8%, dan *F1-Score* 71.9%. Hal ini menunjukkan bahwa model masih memerlukan beberapa perbaikan untuk dapat meningkatkan nilai akurasinya.

3.2 Saran

Saran yang dapat penulis berikan untuk pengembangan sistem selanjutnya adalah :

a. Memperbaiki model maupun dataset yang digunakan untuk mendapatkan akurasi yang lebih baik
b. Menambahkan kelas emosi lain dalam klasifikasi agar emosi yang terdeteksi lebih beragam
c. Mengembangkan tampilan antarmuka agar lebih menarik dan mudah digunakan

DAFTAR PUSTAKA

[1] teguh Setadi, “Dasar Jaringan Syaraf Tiruan,” *Universitas STEKOM*, Apr. 19, 2022. https://sistem-komputer-s1.stekom.ac.id/informasi/baca/Dasar-Jaringan-Syaraf-Tiruan/de13b15343f6d3895ee1468d7eb0e76c061ec9c5 (accessed Jun. 28, 2023).

[2] R. Adinda, “Pengertian Emosi, Macam-Macam Emosi, & Emosi Positif Negatif,” *gramedia*, Mar. 29, 2021. https://www.gramedia.com/best-seller/pengertian-emosi/#Pengertian_Emosi (accessed Jun. 28, 2023).

[3] “Source Code,” *github*. https://github.com/valentingea/final_project_PPDM (accessed Jul. 10, 2023).