

Speech Emotion Recognition

Speech Recognition

Kelompok: 16

- Kevyn Aprilyanto - 2602089793
- Riccardo Davinci - 2602089811
- Deverel Vieri - 2602094225

I. Latar Belakang

Emosi adalah reaksi yang dimiliki oleh manusia terhadap situasi tertentu. Emosi mencakup berbagai perasaan seperti senang, sedih, marah, dan takut. Masing-masing dari emosi tersebut dapat mempengaruhi cara seseorang menyampaikan pesan atau berkomunikasi, terutama dalam komunikasi verbal. Dalam komunikasi verbal, emosi membantu menyampaikan makna lebih dalam di balik kata-kata yang diucapkan. Tanpa adanya emosi, lawan bicara mungkin tidak dapat memahami konteks penuh dari pesan yang disampaikan. Hal yang sama berlaku untuk sebuah sistem speech recognition, sistem yang tidak dapat mendeteksi emosi dapat mengakibatkan output yang salah menginterpretasikan maksud pembicara, terutama dalam situasi dimana emosi dan nada suara sangat menentukan arti kata-kata.

Hadirnya Speech Emotion Recognition atau SER memungkinkan sistem untuk mengenali dan mengidentifikasi emosi dari sebuah ucapan. Hal ini dapat meningkatkan performa dari sebuah sistem speech recognition, termasuk meningkatkan akurasi dari output yang dihasilkan (Wang et al., 2015). Sebagai contoh, asisten virtual seperti Siri dapat merespons pengguna dengan lebih tepat dengan adanya SER yang membantu untuk memahami konteks pembicaraan. SER melibatkan penggunaan berbagai algoritma machine learning atau deep learning untuk menganalisis pola dalam sebuah ucapan. Algoritma-algoritma tersebut dapat mengklasifikasikan emosi ke dalam kategori-kategori yang telah ditentukan, seperti senang, sedih, marah, dan takut.

Teknologi SER tidak hanya bermanfaat untuk meningkatkan interaksi dengan asisten virtual, tetapi juga memiliki aplikasi yang luas dalam berbagai bidang. Misalnya, dalam bidang kesehatan mental, sistem SER dapat digunakan untuk mendeteksi tanda-tanda awal depresi atau kecemasan melalui analisis pola bicara pasien, memungkinkan intervensi yang lebih dini dan lebih tepat. Dalam sektor layanan pelanggan, perusahaan dapat menggunakan SER untuk menilai kepuasan pelanggan secara real-time, membantu agen layanan untuk menyesuaikan respons mereka dan meningkatkan pengalaman pelanggan.

Penelitian telah menunjukkan bahwa SER dapat mencapai tingkat akurasi yang tinggi dengan menggunakan jaringan saraf tiruan dan teknik pembelajaran mendalam lainnya. Penggunaan Convolutional Neural Networks (CNN) dan Recurrent Neural Networks (RNN) dalam SER dapat menghasilkan akurasi yang lebih baik dibandingkan metode tradisional. Data dari berbagai sumber, termasuk rekaman suara dan dataset emosi, digunakan untuk melatih

model SER, yang kemudian diuji dan divalidasi untuk memastikan kinerjanya dalam situasi dunia nyata.

Dengan kemajuan teknologi dan peningkatan kapasitas komputasi, pengembangan sistem SER menjadi lebih terjangkau dan lebih efisien. Masa depan SER menjanjikan peningkatan interaksi manusia-komputer yang lebih alami dan intuitif, membuka jalan bagi inovasi dalam berbagai sektor industri. Dalam dunia yang semakin digital, kemampuan untuk memahami dan merespons emosi manusia melalui teknologi menjadi aspek penting untuk menciptakan pengalaman pengguna yang lebih personal.

Referensi :

- Wang, K., An, N., Li, B.N., & Zhang, Y. (2015). Speech Emotion Recognition Using Fourier Parameters. *IEEE Transactions on Affective Computing*, 6(1), 69-75.
<http://dx.doi.org/10.1109/TAFFC.2015.2392101>

II. Rumusan Masalah

1. Bagaimana cara mendeteksi dan mengklasifikasikan emosi dari ucapan manusia secara akurat menggunakan teknologi Artificial Intelligence?
2. Apa saja tantangan utama yang dihadapi saat mengimplementasikan SER dalam sistem speech recognition?
3. Bagaimana implementasi sistem SER dapat membantu dalam berbagai bidang seperti kesehatan mental dan layanan pelanggan?

III. Tujuan Project

1. Mengembangkan sistem Speech Emotion Recognition (SER) yang mampu mendeteksi dan mengklasifikasikan emosi dari ucapan manusia dengan tingkat akurasi yang tinggi.
2. Meneliti dan mengidentifikasi algoritma deep learning yang paling efektif untuk digunakan dalam SER.
3. Menghasilkan model sistem speech recognition ke dalam proses pengenalan suara dengan akurasi yang tinggi dan efisien.

IV. Ruang Lingkup Project

Dataset yang kami gunakan dalam proyek ini berupa file audio dengan format "WAV". Dataset ini disimpan dalam folder bernama "Tess", yang berisi 14 folder subkategori emosi. Tujuan utama dari proyek ini adalah mengembangkan sistem Speech Emotion Recognition (SER) yang mampu mendeteksi dan mengklasifikasikan emosi dari ucapan manusia dengan tingkat akurasi yang tinggi. Untuk mencapai tujuan ini, kami menggunakan dua model deep learning: Long Short-Term Memory (LSTM) dan Convolutional Neural Network (CNN). Model LSTM digunakan untuk menangkap pola temporal dalam data audio, sementara model CNN diterapkan untuk mengekstraksi fitur spasial dari sinyal suara.
















Proses pengembangan meliputi beberapa tahapan, dimulai dari preprocessing data, di mana fitur-fitur penting seperti Mel-Frequency Cepstral Coefficients (MFCC) diekstraksi dari file audio. Setelah itu, data dinormalisasi untuk memastikan konsistensi dan akurasi dalam proses pelatihan model. Model LSTM dan CNN kemudian dilatih menggunakan dataset yang telah

diproses ini, diikuti dengan validasi dan pengujian untuk memastikan kinerja model dalam mendeteksi berbagai emosi dengan tepat.

V. Deskripsi Dataset

Data Explorer

Version 1 (281.33 MB)

- ▼  TESS Toronto emotional s
 - ▶  OAF_Fear
 - ▶  OAF_Pleasant_surprise
 - ▶  OAF_Sad
 - ▶  OAF_angry
 - ▶  OAF_disgust
 - ▶  OAF_happy
 - ▶  OAF_neutral
 - ▶  YAF_angry
 - ▶  YAF_disgust
 - ▶  YAF_fear
 - ▶  YAF_happy
 - ▶  YAF_neutral
 - ▶  YAF_pleasant_surprise
 - ▶  YAF_sad

Dataset yang digunakan dalam proyek ini berupa file audio dalam format "WAV" yang terorganisir dalam folder bernama "Tess". Dataset ini terdiri dari rekaman suara yang telah dilabeli dengan berbagai jenis emosi seperti disgust, angry, sad, happy, surprise, neutral, dan lainnya, dengan total 2800 file audio. Dataset ini hanya berisi suara dari dua aktris wanita berusia 26 dan 64 tahun, memberikan kualitas audio yang sangat tinggi dan membantu mengatasi bias gender yang sering ditemukan dalam dataset lainnya. Setiap file audio mengandung kata target yang diucapkan dalam frasa "Say the word _", mencakup tujuh emosi yang berbeda. Penggunaan dataset ini memungkinkan model untuk dilatih dan diuji secara efektif dalam mendeteksi emosi dari suara, menggunakan algoritma LSTM dan CNN. Dengan karakteristik yang kaya dan representasi yang seimbang, dataset TESS ini memainkan peran penting dalam pengembangan sistem Speech Emotion Recognition yang akurat dan mampu melakukan generalisasi dengan baik.

VI. Tahap-tahap Eksperimen

1. Import library yang diperlukan

```
import pandas as pd
import numpy as np
import pathlib

import os
import sys

import librosa
import librosa.display
import seaborn as sns
import matplotlib.pyplot as plt
import scipy
from scipy.signal import resample

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

from IPython.display import Audio

import keras
from keras.callbacks import ReduceLROnPlateau, LearningRateScheduler
from keras.models import Sequential
from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten, Dropout, BatchNormalization, LSTM
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint

import warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Berikut adalah tabel yang menjelaskan secara singkat fungsi dari tiap library dan module yang dipakai:

Library/Module	Fungsi
pandas	Menyimpan dan memanipulasi data dalam bentuk DataFrame
numpy	Komputasi numerik dan manipulasi array
os	Berinteraksi dengan file di sistem
sys	Mengatur warnings
librosa	Memuat file audio, mengekstrak fitur-fitur audio, membuat waveplot dan spectrogram, augmentasi audio
seaborn	Membuat plot statistik, seperti count plot
matplotlib.pyplot	Visualisasi data
sklearn.preprocessing	Penskalaan data dan one-hot encoding label
sklearn.metrics	Mengevaluasi model dengan confusion matrix dan classification report
sklearn.model_selection	Membagi dataset menjadi train dan test set
IPython.display.Audio	Memutar audio di Jupyter Notebook
keras	Melatih model neural network, seperti Conv1D dan LSTM
warnings	Menonaktifkan warnings

2. Mempersiapkan dataset

```
Tess = "./Tess/"

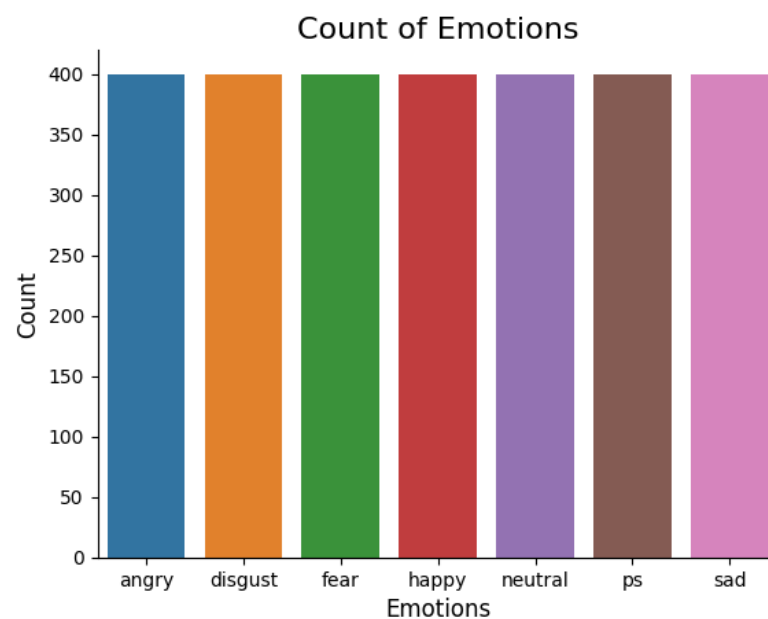
paths = []
labels = []

for dirname, _, filenames in os.walk(Tess):
    for filename in filenames:
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        paths.append(os.path.join(dirname, filename))
        label = filename.split('_')[-1]
        label = label.split('.')[0]
        labels.append(label.lower())
print('Dataset is Loaded')

df = pd.DataFrame({'path':paths, 'label':labels})
```

Dataset diambil dari direktori “Tess” dengan mengiterasi seluruh file dalam direktori tersebut dan menyimpan path file dan label emosi dari nama file ke dalam list *paths* dan *label*. Kemudian, kedua list tersebut digabungkan ke dalam sebuah DataFrame pandas.

3. Visualisasi Data Awal



```
plt.title('Count of Emotions', size=16)
sns.countplot(x = df.label)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)
sns.despine(top=True, right=True, left=False, bottom=False)
plt.show()
```

Bar chart digunakan untuk menampilkan jumlah kemunculan setiap emosi dalam dataset, dengan sumbu y mewakili *count* dan sumbu x mewakili *emotions*.

4. Visualisasi gelombang dan spektrogram

```
def create_waveplot(data, sr, e):
    plt.figure(figsize=(10, 3))
    plt.title('Waveplot for audio with {} emotion'.format(e), size=15)
    librosa.display.waveshow(data, sr=sr)
    plt.show()

def create_spectrogram(data, sr, e):
    # stft function converts the data into short term fourier transform
    X = librosa.stft(data)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(12, 3))
    plt.title('Spectrogram for audio with {} emotion'.format(e), size=15)
    librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
    #librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
    plt.colorbar()
```

Function *create_waveplot* menerima data audio, sample rate, dan emotion sebagai argumen dan menghasilkan waveplot, sedangkan function *create_spectrogram* menggunakan Short-Time Fourier Transform (STFT) untuk mengonversi data audio menjadi spektrogram.

```
emotion='fear'
path = np.array(df.path[df.label==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)
```

Path file audio yang sesuai dengan label emosi yang ditentukan dipilih, lalu function *create_waveplot* dan *create_spectrogram* dipanggil untuk membuat waveplot dan spektrogram dari audio tersebut. Proses ini akan diulang untuk setiap jenis emosi, yaitu angry, disgust, fear, happy, neutral, sad, dan pleasant surprise.

VII. Preprocessing

1. Augmentasi Data

```
def noise(data):
    noise_amp = 0.035*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(data, rate=0.8):
    return librosa.effects.time_stretch(data, rate=rate)

def shift(data):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.7):
    n_steps = 12 * pitch_factor # Assuming 12 steps per octave
    return librosa.effects.pitch_shift(data, sr=sampling_rate, n_steps=n_steps)
```

Beberapa proses augmentasi data dilakukan pada project ini:

- Function *noise* digunakan untuk menambahkan noise pada audio dengan menghitung amplitudo noise berdasarkan nilai maksimum dan menambahkannya menggunakan secara acak.
- Function *stretch* digunakan untuk meregangkan audio dengan memanfaatkan library librosa.
- Function *shift* digunakan untuk menggeser audio secara acak.
- Function *pitch* digunakan untuk mengubah pitch audio dengan memanfaatkan library librosa.

2. Feature Extraction

```

def extract_features(data):
    result = np.array([])
    # MFCC
    mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate, n_mfcc=40).T, axis=0)
    result = np.hstack((result, mfcc))

    return result

def get_features(path):
    data, sample_rate = librosa.load(path, duration=2.5, offset=0.6)

    # without augmentation
    res1 = extract_features(data)
    result = np.array(res1)

    # data with noise
    noise_data = noise(data)
    res2 = extract_features(noise_data)
    result = np.vstack((result, res2))

    # data with stretching and pitching
    new_data = stretch(data)
    data_stretch_pitch = pitch(new_data, sample_rate)
    res3 = extract_features(data_stretch_pitch)
    result = np.vstack((result, res3))

    return result

```

Function *extract_features* untuk mengekstraksi fitur-fitur dari data audio menggunakan Mel-Frequency Cepstral Coefficients (MFCC). Kemudian dilanjutkan dengan function *get_features* digunakan untuk mengambil fitur-fitur dari sebuah file audio dengan memuat dari path yang diberikan serta variasinya yang telah di augment.

```

X, Y = [], []
for path, emotion in zip(df['path'], df['label']):
    print(path, emotion)
    feature = get_features(path)
    for ele in feature:
        X.append(ele)
        Y.append(emotion)

Features = pd.DataFrame(X)
Features['labels'] = Y

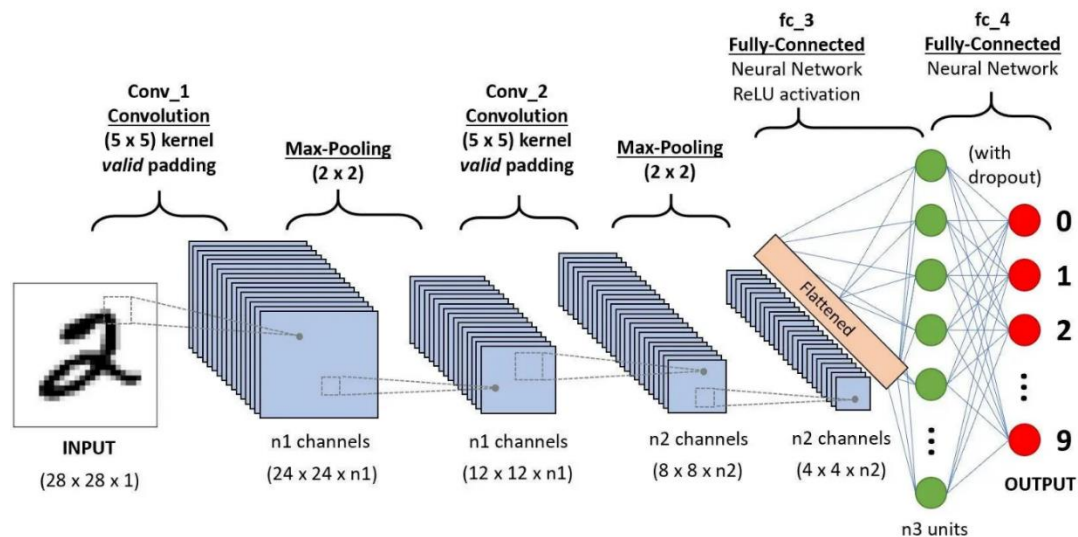
```

Iterasi dilakukan melalui setiap path dan label emosi dalam DataFrame. Untuk setiap path, fungsi *get_features* digunakan untuk mengekstraksi fitur-fitur audio. Fitur-fitur ini kemudian ditambahkan ke dalam list X, sementara label emosi ditambahkan ke dalam list Y. Kedua list tersebut digunakan untuk membuat DataFrame baru, dimana fitur-fitur audio disimpan sebagai data kolom, dan label emosi disimpan sebagai kolom terpisah dengan nama *labels*.

VIII. Rancangan Model

Dalam proyek ini, kami menggunakan dua model deep learning untuk mengenali emosi dari suara, yaitu Convolutional Neural Network (CNN) dan Long Short-Term Memory (LSTM).

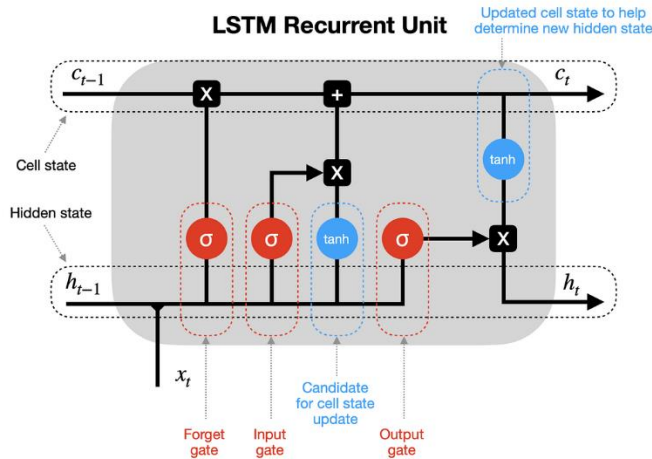
- Convolutional Neural Network (CNN)



Convolutional Neural Network adalah jenis jaringan saraf tiruan yang dirancang untuk memproses data yang memiliki struktur grid, seperti gambar atau data audio. CNN terdiri dari beberapa lapisan konvolusi yang bertujuan untuk mengekstraksi fitur lokal dari input, diikuti oleh lapisan pooling untuk mengurangi dimensi fitur. Dalam proyek ini, CNN kami memiliki beberapa lapisan konvolusi dengan filter yang semakin berkurang jumlahnya, yang kemudian diikuti oleh lapisan pooling dan dropout untuk mengurangi overfitting. Setelah fitur diekstraksi, data diflatten dan diteruskan ke lapisan fully connected untuk klasifikasi. CNN bekerja dengan memindai seluruh input menggunakan kernel konvolusi untuk mendeteksi fitur penting dan menghasilkan prediksi berdasarkan fitur-fitur tersebut. Kelebihan CNN terletak pada kemampuannya untuk secara otomatis mengidentifikasi fitur penting dari data tanpa memerlukan ekstraksi fitur manual.

- Long Short-Term Memory (LSTM)

LONG SHORT-TERM MEMORY NEURAL NETWORKS



Long Short-Term Memory adalah jenis Recurrent Neural Network (RNN) yang khusus dirancang untuk mengatasi masalah vanishing gradient pada RNN tradisional. LSTM memiliki sel memori yang dapat menyimpan informasi dalam jangka waktu yang panjang, yang sangat berguna dalam analisis data sekuensial seperti sinyal audio. Model LSTM dalam proyek ini terdiri dari beberapa lapisan LSTM yang saling berhubungan, diikuti oleh lapisan fully connected untuk klasifikasi. LSTM bekerja dengan menangkap dependensi temporal dalam data sekuensial, memungkinkan model untuk mengenali pola emosi yang tersebar sepanjang sinyal audio. Kelebihan LSTM termasuk kemampuannya untuk menangani data sekuensial yang panjang dan menangkap informasi temporal yang kompleks, menjadikannya ideal untuk tugas-tugas seperti pengenalan emosi dari suara.

IX. Proses Pelatihan Model

- Convolutional Neural Networks (CNN)
 1. Split train-test data

```

X = Features.iloc[:, :-1].values
Y = Features['labels'].values

encoder = OneHotEncoder()
Y = encoder.fit_transform(np.array(Y).reshape(-1, 1)).toarray()

x_train, x_test, y_train, y_test = train_test_split(X, Y,
random_state=0, shuffle=True)

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)

x_train = np.expand_dims(x_train, axis=2)
x_test = np.expand_dims(x_test, axis=2)

```

Fitur-fitur audio dan label dipisah menjadi dua array berbeda dan label emosi diubah dengan OneHotEncoder. Kemudian, data dibagi menjadi train dan test set dan fitur-fitur dinormalisasi menggunakan StandardScaler. Dimensi diubah untuk memenuhi persyaratan input model menggunakan library numpy.

2. Training model

```

model = Sequential()
model.add(Conv1D(256, kernel_size=5, strides=1, padding='same',
activation='relu', input_shape=(x_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=5, strides=2, padding='same'))
model.add(Conv1D(256, kernel_size=5, strides=1, padding='same',
activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides=2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv1D(128, kernel_size=5, strides=1, padding='same',
activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides=2, padding='same'))
model.add(Dropout(0.2))
model.add(Conv1D(64, kernel_size=5, strides=1, padding='same',
activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides=2, padding='same'))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(units=32, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(units=7, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])
model.summary()

rlrp = ReduceLRonPlateau(monitor='loss', factor=0.4, verbose=0,
patience=2, min_lr=0.0000001)
history = model.fit(x_train, y_train, batch_size=64, epochs=50,
validation_data=(x_test, y_test), callbacks=[rlrp])

```

Kode di atas adalah implementasi sebuah Convolutional Neural Network (CNN) menggunakan framework Keras di Python. CNN adalah jenis arsitektur jaringan saraf tiruan yang umumnya digunakan untuk tugas-tugas pengenalan pola pada data spasial seperti citra.

Pertama-tama, kita mendefinisikan model sebagai sebuah objek Sequential, yang berarti kita akan membuat model secara berurutan, lapisan demi lapisan. Lapisan pertama adalah Conv1D, yang merupakan lapisan konvolusi satu dimensi. Ini digunakan untuk mengekstraksi fitur-fitur dari input. Di sini, kita memiliki 256 filter dengan kernel size 5, activation 'relu', dan padding 'same', yang berarti kita mempertahankan dimensi input. Input shape diatur sesuai dengan bentuk data input kita, yaitu x_train.shape[1] yang merepresentasikan panjang data dan 1 untuk jumlah saluran (channel).

Setelah itu, kita tambahkan MaxPooling1D layer untuk mengurangi dimensi output dari lapisan konvolusi sebelumnya. MaxPooling1D digunakan untuk mengekstraksi fitur yang paling penting dengan mengambil nilai maksimum dalam jendela yang ditentukan. Di sini, kita menggunakan pool_size 5, strides 2, dan padding 'same' untuk mempertahankan dimensi.

Proses di atas diulangi dengan menambahkan beberapa lapisan Conv1D dan MaxPooling1D lagi. Dropout layer juga diterapkan setelah beberapa lapisan konvolusi untuk mencegah overfitting. Dropout secara acak mengabaikan sebagian unit selama pelatihan. Kemudian, kita mendatar (flatten) output dari lapisan konvolusi terakhir untuk mempersiapkannya sebagai input ke lapisan-lapisan berikutnya yang sepenuhnya terhubung. Setelah mendatar, kita tambahkan Dense layers, yang merupakan lapisan-lapisan sepenuhnya terhubung. Ini bertanggung jawab untuk menghubungkan setiap neuron dari lapisan sebelumnya ke setiap neuron pada lapisan berikutnya. Activation function 'relu' digunakan di lapisan-lapisan ini untuk memperkenalkan non-linearitas ke dalam model.

Terakhir, kita tambahkan lapisan Dense dengan unit keluaran sesuai dengan jumlah kelas yang ingin diprediksi (dalam kasus ini, 7 kelas), dengan activation 'softmax'. Softmax digunakan untuk menghasilkan probabilitas keluaran untuk setiap kelas. Ringkasnya, kode ini menggambarkan pembangunan CNN dengan beberapa lapisan konvolusi dan pooling untuk mengekstraksi fitur dari data input, diikuti oleh lapisan-lapisan sepenuhnya terhubung untuk melakukan klasifikasi pada keluaran yang dihasilkan.

```
rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0, patience=2, min_lr=0.000001)
history=model.fit(x_train, y_train, batch_size=64, epochs=50, validation_data=(x_test, y_test), callbacks=[rlrp])
```

Kode tersebut merupakan bagian dari proses pelatihan model Convolutional Neural Network (CNN). Digunakan callback ReduceLROnPlateau untuk memantau dan menyesuaikan laju pembelajaran saat loss tidak menurun signifikan setelah 2 epoch. Proses pelatihan dilakukan menggunakan data pelatihan (x_train, y_train) dengan fungsi fit, sementara evaluasi model dilakukan pada data validasi (x_test, y_test). Callback ReduceLROnPlateau dijalankan selama pelatihan untuk mengatur laju pembelajaran.

- Long Short-Term Memory (LSTM)

1. Split train-test data

```
def extract_mfcc(filename):
    y, sr = librosa.load(filename, duration = 3, offset = 0.5)
    mfcc = np.mean(librosa.feature.mfcc(y = y, sr = sr, n_mfcc = 40).T, axis = 0)
    return mfcc

X_mfcc = df['path'].apply(lambda x: extract_mfcc(x))

X = [x for x in X_mfcc]
X = np.array(X)

X = np.expand_dims(X, -1)

enc = OneHotEncoder()
y = enc.fit_transform(df[['label']])

y = y.toarray()

x_train, x_test, y_train, y_test = train_test_split(X, y, random_state=0, shuffle=True)
```

Kode tersebut mempersiapkan data untuk melatih model LSTM dengan mengekstraksi fitur MFCC (Mel Frequency Cepstral Coefficients) dari file audio, menggunakan librosa untuk memuat dan memproses audio, serta menghitung rata-rata MFCC dari 40 komponen pertama.

Ekstraksi ini diterapkan pada setiap jalur file audio dalam DataFrame, menghasilkan array numpy yang kemudian diperluas dimensinya agar sesuai dengan input model LSTM. Label kategori diubah menjadi format one-hot encoded untuk kompatibilitas dengan model.

Data fitur dan label dibagi menjadi set pelatihan dan pengujian menggunakan `train_test_split`, dengan pengacakan untuk memastikan distribusi acak dan hasil yang dapat direproduksi. Proses ini menghasilkan data siap pakai untuk melatih dan menguji model LSTM dalam tugas klasifikasi atau prediksi berbasis audio.

2. Training-model

```
model = Sequential([
    LSTM(256, return_sequences=False, input_shape=(40,1)),
    Dropout(0.2),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(64, activation='relu'),
    Dropout(0.2),
    Dense(32, activation='relu'),
    Dropout(0.3),
    Dense(7, activation='softmax')
])

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

checkpoint_filepath = 'best_model.keras'
model_checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_filepath,
    save_best_only=True,
    monitor='val_accuracy',
    mode='max',
    verbose=1)

def scheduler(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * tf.math.exp(-0.1)

lr_scheduler_callback = LearningRateScheduler(scheduler)

history = model.fit(x_train, y_train, validation_split=0.2, epochs=50, batch_size=64
```

Setelah data siap, model LSTM dibangun menggunakan `Sequential` dengan lapisan LSTM berukuran 256-unit yang tidak mengembalikan urutan (`return_sequences=False`), diikuti oleh beberapa lapisan `Dense` dengan aktivasi ReLU dan lapisan `Dropout` untuk mencegah overfitting. Lapisan output menggunakan aktivasi softmax dengan 7-unit untuk klasifikasi multikategori.

Model dikompilasi dengan loss `categorical_crossentropy`, optimizer `adam`, dan metrik `accuracy`, kemudian ditampilkan arsitekturnya dengan `model.summary()`. Untuk menyimpan model terbaik, digunakan callback `ModelCheckpoint` yang menyimpan model dengan akurasi validasi tertinggi di jalur `'best_model.keras'`. Sebuah fungsi scheduler didefinisikan untuk menyesuaikan laju pembelajaran setelah 10 epoch, yang diimplementasikan melalui `LearningRateScheduler`.

Model kemudian dilatih menggunakan data pelatihan dengan validasi split 20%, selama 50 epoch, dan batch size 64, serta callback untuk checkpoint dan scheduler diaktifkan selama pelatihan untuk memastikan model optimal dan laju pembelajaran yang sesuai. Hasil pelatihan disimpan dalam variabel `history` yang berisi informasi tentang performa model selama pelatihan.

X. Hasil Evaluasi dan Analisis

- Convolutional Neural Networks (CNN)

1. Accuracy dan loss model

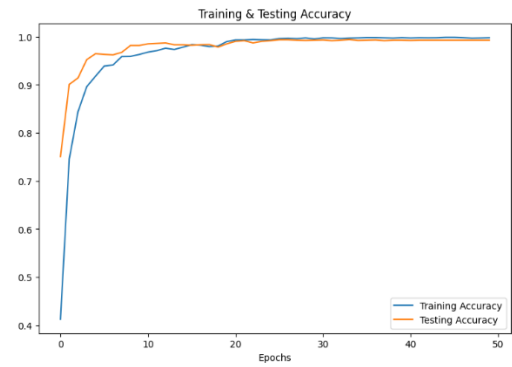
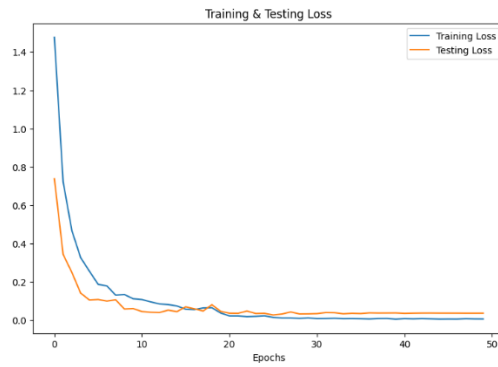
```
print("Accuracy of our model on test data : " ,
      model.evaluate(x_test,y_test)[1]*100 , "%")

epochs = [i for i in range(50)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
test_acc = history.history['val_accuracy']
test_loss = history.history['val_loss']

fig.set_size_inches(20,6)
ax[0].plot(epochs , train_loss , label = 'Training Loss')
ax[0].plot(epochs , test_loss , label = 'Testing Loss')
ax[0].set_title('Training & Testing Loss')
ax[0].legend()
ax[0].set_xlabel("Epochs")

ax[1].plot(epochs , train_acc , label = 'Training Accuracy')
ax[1].plot(epochs , test_acc , label = 'Testing Accuracy')
ax[1].set_title('Training & Testing Accuracy')
ax[1].legend()
ax[1].set_xlabel("Epochs")
plt.show()
```

Akurasi model dihitung menggunakan function `evaluate`. Kemudian, grafik yang menampilkan perubahan nilai loss dan akurasi pada set pelatihan dan pengujian dari setiap epoch pelatihan model dibuat menggunakan `plot`.



2. Prediksi model

```

pred_test = model.predict(x_test)
y_pred = encoder.inverse_transform(pred_test)

y_test = encoder.inverse_transform(y_test)

output = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])
output['Predicted Labels'] = y_pred.flatten()
output['Actual Labels'] = y_test.flatten()

output.head(10)

```

Prediksi dilakukan menggunakan model yang telah dilatih, kemudian hasil prediksi disimpan dalam bentuk label asli dan disimpan dalam DataFrame baru.

	Predicted Labels	Actual Labels
0	neutral	neutral
1	ps	ps
2	neutral	neutral
3	happy	happy
4	ps	ps
5	fear	fear
6	fear	fear
7	sad	sad
8	disgust	disgust
9	fear	fear

Hasil prediksi menunjukkan bahwa model berhasil memprediksi 10 dari 10 baris pertama dari DataFrame tersebut.

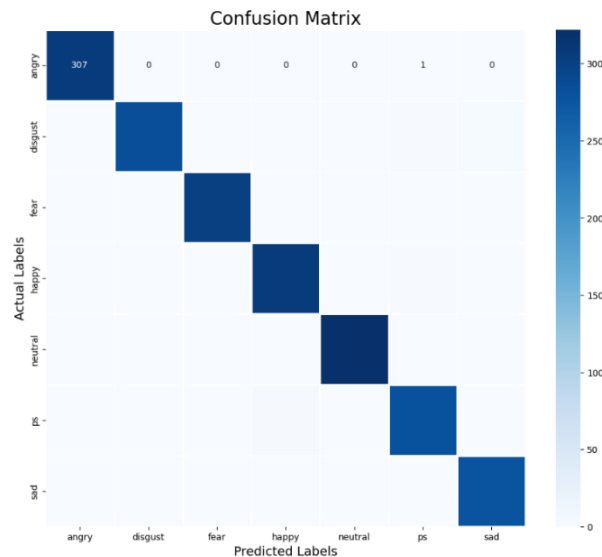
3. Confusion Matrix

```

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize = (12, 10))
cm = pd.DataFrame(cm , index = [i for i in encoder.categories_] ,
columns = [i for i in encoder.categories_])
sns.heatmap(cm, linecolor='white', cmap='Blues', linewidth=1,
annot=True, fmt='')
plt.title('Confusion Matrix', size=20)
plt.xlabel('Predicted Labels', size=14)
plt.ylabel('Actual Labels', size=14)
plt.show()

```

Confusion matrix dihitung menggunakan data uji dan prediksi model. Kemudian, confusion matrix tersebut diubah menjadi DataFrame dengan label kategori sebagai indeks dan kolom. Confusion matrix kemudian divisualisasikan sebagai heatmap menggunakan seaborn.



Hasil confusion matrix memberikan gambaran tentang prediksi kelas-kelas emosi pada data uji. Pada kelas *angry*, terlihat bahwa model berhasil memprediksi 307 data dengan benar sesuai dengan label aslinya. Model hanya salah memprediksi 1 data *angry* menjadi *pleasant surprise*. Hal ini menunjukkan bahwa model memiliki tingkat akurasi yang tinggi dalam memprediksi kelas *angry*, namun masih ada kesalahan prediksi yang terjadi, meskipun jumlahnya sangat kecil.

4. Classification Report

	precision	recall	f1-score	support
angry	1.00	1.00	1.00	308
disgust	1.00	0.98	0.99	291
fear	1.00	1.00	1.00	303
happy	0.98	0.99	0.99	310
neutral	1.00	1.00	1.00	322
ps	0.98	0.98	0.98	287
sad	0.99	1.00	0.99	279
accuracy			0.99	2100
macro avg	0.99	0.99	0.99	2100
weighted avg	0.99	0.99	0.99	2100

Hasil dari classification report menunjukkan performa model dalam mengklasifikasikan emosi dari ucapan manusia pada tujuh kategori: *angry*, *disgust*, *fear*, *happy*, *neutral*, *pleasant surprise*, dan *sad*. Secara keseluruhan, model menunjukkan akurasi yang sangat tinggi (0.99) dengan rata-rata precision, recall,

dan F1-score yang hampir sempurna untuk semua kelas. Hal ini menunjukkan bahwa model memiliki kinerja yang sangat baik dalam mendeteksi dan mengklasifikasikan emosi dari ucapan manusia.

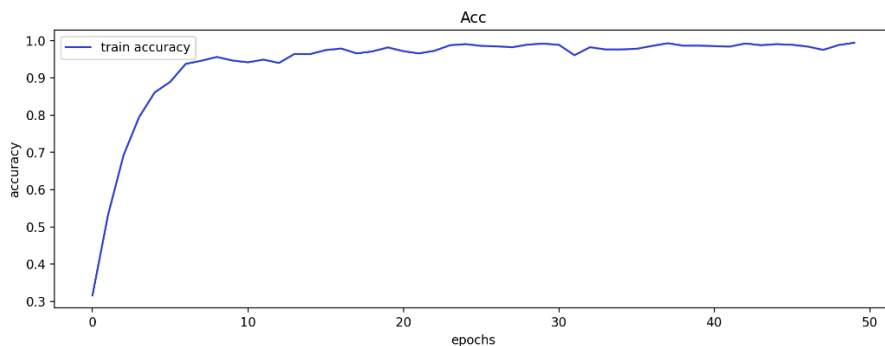
- Long Short-Term Memory (LSTM)

1. Accuracy dan loss model

```
epochs = list(range(50))
acc = history.history['accuracy']

plt.figure(figsize=(12, 4), dpi=200)
plt.plot(epochs, acc, label='train accuracy', color='#2E40CB')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.title('Acc')
plt.legend()
plt.show()
```

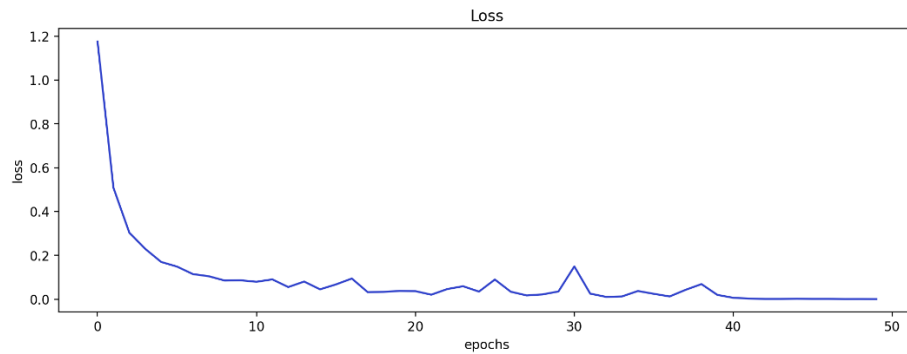
Sebuah daftar epoch dari 0 hingga 49 dibuat, lalu akurasi pelatihan diambil dari history pelatihan. Epoch dan akurasi tersebut kemudian diplot dengan label dan sumbu yang sesuai.



```
loss = history.history['loss']

plt.figure(figsize=(12, 4), dpi=200)
plt.plot(epochs, loss, label='train loss', color='#2E40CB')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('Loss')
plt.show()
```

Proses yang sama juga diterapkan untuk membuat grafik loss model. Loss pelatihan diambil dari history pelatihan dan juga diplot dalam grafik serupa.



```
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=2)
print(f'Test Accuracy: {test_accuracy}')

22/22 - 1s - loss: 0.0764 - accuracy: 0.9786 - 668ms/epoch - 30ms/step
Test Accuracy: 0.9785714149475098
```

Model dievaluasi menggunakan function *evaluate*, yang menghitung loss dan akurasi model menggunakan data *x_test* dan *y_test*. Hasil menunjukkan bahwa evaluasi dilakukan dalam 22 batch dan membutuhkan 1 detik untuk diselesaikan. Nilai loss pada data uji adalah 0.0764, dan akurasi model pada data uji adalah 97.86%. Waktu yang dibutuhkan untuk setiap epoch adalah 668 milidetik dan setiap langkahnya memerlukan 30 milidetik. Secara keseluruhan, hasil ini menunjukkan bahwa model memiliki kinerja yang sangat baik pada data uji dengan akurasi yang tinggi.

XI. Kesimpulan dan Saran

Dalam project ini, deteksi dan klasifikasi emosi dari ucapan manusia telah berhasil dilakukan dengan menggunakan teknologi Artificial Intelligence, khususnya model CNN dan LSTM. Proses ini melibatkan ekstraksi fitur MFCC (Mel Frequency Cepstral Coefficients) dari rekaman suara, yang kemudian digunakan sebagai input untuk model-model tersebut. Hasilnya sangat mengesankan, dengan akurasi mencapai 99% untuk model CNN dan 97% untuk model LSTM, menunjukkan bahwa pendekatan ini sangat efektif dalam mendeteksi dan mengklasifikasikan emosi secara akurat. Model CNN menunjukkan performa yang lebih baik dibandingkan dengan LSTM, yang bisa disebabkan oleh kemampuan CNN dalam menangkap fitur spasial dari data audio dengan lebih efisien. CNN juga cenderung memiliki arsitektur yang lebih sederhana dan lebih cepat dalam proses pelatihan dibandingkan dengan LSTM yang kompleks dan membutuhkan lebih banyak waktu untuk melatih karena sifatnya yang berurutan dan penggunaan memori jangka panjang. Namun, implementasi Speech Emotion Recognition (SER) dalam sistem pengenalan suara menghadapi beberapa tantangan utama. Tantangan ini termasuk variabilitas emosi antar individu, kualitas rekaman suara yang bervariasi, serta kebisingan lingkungan yang dapat mengganggu akurasi deteksi

emosi. Selain itu, kompleksitas emosi manusia yang seringkali campuran dan tidak mudah didefinisikan secara jelas menjadi kendala dalam melatih model yang akurat dan generalis.

Implementasi sistem SER dapat membawa manfaat signifikan di berbagai bidang. Dalam bidang kesehatan mental, sistem SER dapat digunakan untuk memantau kondisi emosional pasien secara real-time, membantu dalam diagnosis, dan memberikan intervensi yang lebih tepat waktu. Dalam layanan pelanggan, SER dapat meningkatkan kualitas interaksi dengan mendeteksi emosi pelanggan, memungkinkan penyedia layanan untuk merespons dengan lebih empatik dan efektif, yang pada akhirnya meningkatkan kepuasan pelanggan dan efisiensi layanan.

Untuk meningkatkan performa dan generalisasi model, di masa yang akan datang disarankan untuk menggunakan dataset yang lebih besar dan lebih beragam, termasuk berbagai bahasa dan dialek serta kondisi rekaman yang berbeda. Mempertimbangkan penggunaan teknologi terbaru seperti Transformer atau model attention-based yang telah menunjukkan hasil yang mengesankan dalam tugas-tugas pengenalan suara dan analisis teks juga disarankan. Selain itu, menggabungkan SER dengan analisis visual (ekspresi wajah) dan biometrik lainnya dapat meningkatkan akurasi dan reliabilitas sistem secara keseluruhan.

XII. Lampiran

Gituhub : https://github.com/Evnx26/AOL-Speech-Recognition/blob/main/AOL_SpeechRec.ipynb

Kaggle : <https://www.kaggle.com/datasets/ejlok1/toronto-emotional-speech-set-tess/data>