

**EMOTION IDENTIFICATION SYSTEM**

**PENGANTAR PEMROSESAN DATA MULTIMEDIA**



**Tim Penyusun (Kelompok 1):**

**Ni Wayan Amanda Putri Astawa (2108561029/D)**

**I Komang Widia Pratama (2108561040/D)**

**Viencent (2108561068/D)**

**Ketut Agus Cahyadi Nanda (2108561079/D)**

**PROGRAM STUDI INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS UDAYANA**

**JIMBARAN**

**2023**

# BAB I

## PENDAHULUAN

### 1.1 Latar Belakang

Dalam era informasi yang semakin maju ini, analisis audio telah menjadi bagian integral dari banyak aplikasi, seperti pengenalan suara, pengenalan musik, pemrosesan bahasa alami berbasis audio, dan banyak lagi. Data audio yang dikumpulkan dari berbagai sumber, seperti mikrofon, rekaman, atau platform media sosial, seringkali dalam bentuk sinyal audio mentah yang kompleks dan sulit untuk dianalisis secara langsung. Oleh karena itu, preprocessing merupakan tahapan yang krusial dalam mengolah data audio agar dapat digunakan untuk analisis lebih lanjut.

Preprocessing bertujuan untuk mengubah data audio mentah menjadi representasi yang lebih sesuai untuk analisis dan memperoleh fitur-fitur penting dari sinyal audio tersebut [1]. Salah satu metode preprocessing yang populer adalah penggunaan Mel-Frequency Cepstral Coefficients (MFCC) [2]. Metode MFCC berhasil mengatasi beberapa tantangan analisis audio dengan menggabungkan informasi tentang karakteristik frekuensi dan temporal dari sinyal audio. Namun, dengan semakin kompleksnya tugas analisis audio, metode konvensional seperti MFCC mungkin memerlukan peningkatan kinerja untuk mencapai tingkat akurasi yang lebih tinggi. Inilah yang mendorong kemunculan jaringan syaraf tiruan (artificial neural networks/ANN) dalam pengolahan data audio.

Jaringan syaraf tiruan adalah model komputasi yang terinspirasi dari cara kerja sistem saraf manusia. Mereka terdiri dari jaringan neuron buatan yang bekerja secara paralel untuk memproses informasi dan mempelajari pola-pola yang rumit dari data input [3]. Dengan kemampuan adaptasi dan pembelajaran yang kuat, jaringan syaraf tiruan telah membawa revolusi dalam berbagai bidang, termasuk analisis audio. Penggunaan jaringan syaraf tiruan dalam analisis audio memberikan peluang untuk meningkatkan kemampuan analisis, termasuk pengenalan pola suara, klasifikasi musik, atau bahkan peningkatan kualitas audio. Integrasi MFCC dengan jaringan syaraf tiruan memungkinkan kombinasi keunggulan fitur ekstraksi MFCC dengan kemampuan adaptasi dan

generalisasi jaringan syaraf tiruan, sehingga hasil analisis audio dapat ditingkatkan dengan signifikan.

## **1.2 Rumusan Masalah**

Rumusan masalah yang akan dibahas dalam laporan ini adalah:

1. Bagaimana implementasi metode Multi-layer Neural Network dengan menggunakan Mel-Frequency Cepstral Coefficients dalam mendeteksi sentiment berdasarkan citra audio?
2. Bagaimana mengembangkan aplikasi analisis sentiment berdasarkan citra audio berbasis web?

## **1.3 Tujuan**

Tujuan dari dibuatnya program ini adalah sebagai berikut:

1. Mengetahui sentiment atau emosi seseorang berdasarkan citra audio.
2. Mengetahui kombinasi hyperparameter dengan tingkat akurasi paling tinggi.

## **1.4 Manfaat**

Manfaat dari dibuatnya program ini adalah sebagai berikut:

1. Memberikan wawasan yang lebih dalam mengenai analisis sentiment untuk citra audio.
2. Memberikan wawasan mengenai kombinasi hyperparameter yang memiliki tingkat akurasi paling tinggi dalam preprocessing citra audio.

## BAB II

### ISI

## 2.1 Manual Aplikasi

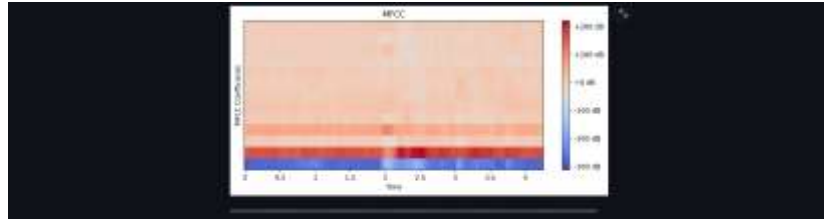
### 2.1.1 Fitur Sistem dan Antar Muka



Gambar 2.1.1.1 Tampilan Awal Aplikasi.

Berikut merupakan tampilan awal aplikasi. User dapat menginputkan data audio dengan format .wav





*Gambar 2.1.1.2 Tampilan Hasil Preprocessing.*

Setelah user menginput file audio, maka program akan secara otomatis melakukan identifikasi sentimen dari file tersebut. Analisis yang dilakukan terdiri dari akurasi, presisi, recal, dan juga F1-Score. Selain itu juga ditampilkan juga cepstral coefficients dan juga chart analysis dari MFCC.

## 2.2 Source Code Modul

### 2.2.1 Source Code

#### 2.2.1.1 Audio\_Preprocessing.py

```
def preprocess_audio(audio_path, target_sample_rate):
    """
    Melakukan preprocessing audio dengan menghitung fitur MFCC dari file audio.
    Mengubah sampling rate audio menjadi target_sample_rate.
    """
    signal, sample_rate = sf.read(audio_path)
    signal_resampled = librosa.resample(signal, orig_sr=sample_rate, target_sr=target_sample_rate)
    signal_normalized = librosa.util.normalize(signal_resampled)
    mfccs = librosa.feature.mfcc(y=signal_normalized, sr=target_sample_rate, n_mfcc=13)
    mfccs_mean = np.mean(mfccs, axis=1)
    return mfccs_mean, mfccs
```

*Gambar 2.2.1.1.1 Audio\_Preprocessing.py.*

Fungsi **preprocess\_audio** digunakan untuk melakukan preprocessing audio dengan beberapa langkah. Pertama-tama, fungsi ini akan membaca file audio dari **audio\_path** menggunakan library **soundfile** dan menyimpan sinyal audio serta sample rate-nya. Selanjutnya, sinyal audio diubah menjadi target sample rate yang diinginkan menggunakan fungsi **resample** dari library **librosa**. Kemudian, sinyal audio yang telah diubah sample rate-nya dinormalisasi menggunakan fungsi **normalize** dari library **librosa.util**. Selanjutnya, fitur MFCC (Mel-frequency cepstral coefficients) dihitung dari sinyal audio yang telah dinormalisasi menggunakan fungsi **feature.mfcc** dari

library **librosa**. Terakhir, nilai rata-rata dari setiap koefisien MFCC diambil menggunakan **np.mean** dan dikembalikan bersama dengan seluruh koefisien MFCC.

### 2.2.1.2 GUI.py

```
def deploy_web_app(best_model, x_test, y_test, target_sample_rate):
    """
    Mendeploy model terbaik ke dalam aplikasi web menggunakan Streamlit.
    """
    st.title("Identifikasi Sentimen dari Suara")
    st.write("Aplikasi ini menggunakan model neural network untuk mengidentifikasi sentimen atau mood dari suara.")

    audio_files = st.file_uploader("Upload satu atau beberapa file audio", type="wav",
                                    accept_multiple_files=True)

    if audio_files is not None:
        for audio_file in audio_files:
            mfcc_mean, mfccs = preprocess_audio(audio_file, target_sample_rate)
            input_data = np.array([mfcc_mean])
            sentiment = "Positive" if np.argmax(best_model.predict(input_data)) == 0 else "Negative"

            accuracy, precision, recall, f1 = evaluate_model(best_model, x_test, y_test)
            st.write("Hasil identifikasi sentimen:")
            st.write(f"File: {audio_file.name}")
            st.write(f"Sentimen: {sentiment}")
            st.write(f"Akurasi: {accuracy}")
            st.write(f"Presisi: {precision}")
            st.write(f"Recall: {recall}")
            st.write(f"F1-Score: {f1}")
            st.write("=====")

            st.write("Spectral Coefficients:")
            for i, coef in enumerate(mfccs):
                st.write(f"MFCC {i+1}: {coef}")

            st.write("MFCC Chart Analysis:")
            plt.figure(figsize=(10, 4))
            librosa.display.melogram(mfccs, sr=target_sample_rate, show=True)
            plt.colorbar(format="%+2.0f dB")
            plt.title("MFCC")
            plt.xlabel("Time")
            plt.ylabel("MFCC Coefficients")
            st.pyplot(plt)

            st.write("=====")
```

Gambar 2.2.1.2.1 GUI.py.

Fungsi **deploy\_web\_app** mengimplementasikan deploy model ke dalam aplikasi web menggunakan Streamlit. Disini user dapat mengunggah satu atau beberapa file audio dalam format WAV menggunakan fitur **file\_uploader** dari Streamlit. Setelah user mengunggah file audio, fungsi **preprocess\_audio** akan memproses audio tersebut untuk mendapatkan fitur MFCC (Mel-frequency cepstral coefficients). Fitur MFCC ini kemudian digunakan sebagai input untuk model neural network yang telah dilatih (**best\_model**). Hasil dari prediksi model akan menentukan sentimen dari suara tersebut, yaitu "Positive" atau "Negative". Selanjutnya, fungsi **evaluate\_model** akan menghitung metrik evaluasi seperti akurasi, presisi, recall, dan

F1-score dari model menggunakan data uji (**X\_test** dan **y\_test**). Hasil evaluasi ini juga ditampilkan dalam aplikasi.

### 2.2.1.3 Machine\_Learning.py

- **create\_model**

```
def create_model(input_shape, num_classes, num_hidden_layers, num_hidden_neurons, activation):  
    """  
    Membuat model neural network dengan jumlah hidden layer, jumlah neuron, dan fungsi aktivasi yang  
    ditentukan.  
    """  
    model = Sequential()  
    model.add(Dense(num_hidden_neurons, activation=activation, input_shape=input_shape))  
    for i in range(num_hidden_layers - 1):  
        model.add(Dense(num_hidden_neurons, activation=activation))  
    model.add(Dense(num_classes, activation='softmax'))  
    return model
```

*Gambar 2.2.1.3.1 create\_model.*

Fungsi **create\_model** digunakan untuk membuat model jaringan saraf dengan arsitektur yang dapat disesuaikan. Model tersebut terdiri dari layer input, beberapa hidden layer, dan output layer. Jumlah hidden layer, jumlah neuron dalam setiap hidden layer, dan fungsi aktivasi dapat ditentukan sebagai argumen fungsi.

- **train\_model**

```
def train_model(model, X_train, y_train, lr, num_epochs):  
    """  
    Melatih model dengan data train menggunakan learning rate dan jumlah epochs yang ditentukan.  
    """  
    optimizer = SGD(learning_rate=lr)  
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])  
    model.fit(X_train, y_train, epochs=num_epochs, batch_size=32, verbose=1)  
    return model
```

*Gambar 2.2.1.3.2 train\_model.*

Fungsi **train\_model** digunakan untuk melatih model jaringan saraf menggunakan train data dengan learning rate dan jumlah epochs (jumlah iterasi) yang ditentukan. Optimizer diinisialisasi dengan menggunakan SGD (Stochastic Gradient Descent) dengan learning rate yang diberikan. Selanjutnya, model di-compile dengan menggunakan optimizer yang telah diinisialisasi, fungsi

loss diatur sebagai **'categorical\_crossentropy'** (untuk masalah klasifikasi multikelas), dan metrik yang digunakan adalah akurasi (accuracy). Setelah melakukan kompilasi, model dilatih menggunakan metode `fit()`. Data latihan (**X\_train** dan **y\_train**) digunakan untuk melatih model dengan jumlah epochs yang ditentukan, menggunakan batch size sebesar 32. Parameter `verbose` diatur sebagai 0 untuk menghindari output yang terlalu banyak saat melatih model.

- **evaluate\_model**



```
def evaluate_model(model, X_test, y_test):  
    """  
    Mengevaluasi model menggunakan data test dan menghitung metrik evaluasi seperti akurasi, presisi,  
    recall, dan F1-Score.  
    """  
    with warnings.catch_warnings():  
        warnings.filterwarnings("ignore")  
        y_pred = model.predict(X_test).astype(int)  
        y_test = y_test.astype(int)  
  
        accuracy = accuracy_score(y_test, y_pred)  
        precision = precision_score(y_test, y_pred, zero_division=0)  
        recall = recall_score(y_test, y_pred)  
        f1 = f1_score(y_test, y_pred)  
  
    return accuracy, precision, recall, f1
```

*Gambar 2.2.1.3.3 evaluate\_model.*

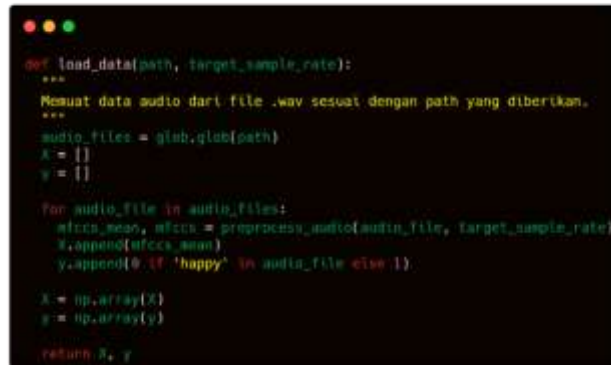
Fungsi **evaluate\_model** digunakan untuk mengevaluasi model jaringan saraf menggunakan data uji (test data) dan menghitung beberapa metrik evaluasi seperti akurasi, presisi, recall, dan F1-Score. Di dalam fungsi tersebut, ada pengaturan untuk mengabaikan peringatan (warnings) yang muncul selama evaluasi model. Selanjutnya, menggunakan model yang telah diberikan, fungsi **predict** digunakan untuk menghasilkan prediksi model terhadap data uji (**X\_test**). Kemudian, fungsi **argmax** dari library NumPy digunakan untuk mengambil indeks nilai maksimum dari prediksi model dan label sebenarnya (**y\_test**). Setelah itu, beberapa metrik evaluasi dihitung berdasarkan hasil prediksi dan



label sebenarnya. Metrik-metrik tersebut meliputi accuracy, precision, recall, dan F1-Score.

#### 2.2.1.4 Main.py

- **load\_data**



```
def load_data(path, target_sample_rate):  
    """  
    Memuat data audio dari file .wav sesuai dengan path yang diberikan.  
    """  
    audio_files = glob.glob(path)  
    X = []  
    y = []  
  
    for audio_file in audio_files:  
        mfccs_mean, mfccs = preprocess_audio(audio_file, target_sample_rate)  
        X.append(mfccs_mean)  
        y.append(0 if 'happy' in audio_file else 1)  
  
    X = np.array(X)  
    y = np.array(y)  
  
    return X, y
```

Gambar 2.2.1.4.1 load\_data.

Fungsi **load\_data** digunakan untuk memuat data audio dari file dengan format .wav. Fungsi ini menerima dua parameter, yaitu **path** yang merupakan path file audio yang ingin dimuat, dan **target\_sample\_rate** yang merupakan sample rate yang diinginkan. **glob.glob** berfungsi untuk mencari semua file audio dengan format .wav yang sesuai dengan path yang diberikan. Kemudian, fungsi melakukan proses preprocessing pada setiap file audio dengan memanggil fungsi **preprocess\_audio** yang mengembalikan nilai rata-rata MFCCs (Mel-frequency cepstral coefficients) dan MFCCs itu sendiri. Nilai rata-rata MFCCs dari setiap file audio ditambahkan ke dalam array **X**, sedangkan label kelas (0 jika terdapat kata 'happy' dalam nama file audio, dan 1 jika tidak) ditambahkan ke dalam array **y**. Setelah itu, array **X** dan **y** diubah menjadi array numpy

menggunakan **np.array** sebelum dikembalikan sebagai output dari fungsi.

- main

```
def main(target_sample_rate):
    """
    Fungsi utama yang menjalankan alur aplikasi.
    """
    project_directory = r"C:\OperasiBI-Mayan
    AnonimDownloadAudioClassificationWithMultiLayerNeuralNetwork
    main_audio_classification_with_multi_layer_neural_network-main"
    main_samples_directory = os.path.join(project_directory, "main_samples")

    X, y = load_data(os.path.join(main_samples_directory, "data"), target_sample_rate)

    num_classes = 2 # Jumlah kelas (positive sentiment dan negative sentiment)
    input_shape = (15,) # Bentuk data input (filter 1000)

    num_samples = len(X)
    num_test_samples = int(0.2*num_samples)

    if num_samples > 0 and num_test_samples > 0:
        test_size = num_test_samples / num_samples
        x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=test_size,
        y_train = y_train, y_test = y_test, random_state=0)
        x_test = x_test[:num_test_samples]
        y_test = y_test[:num_test_samples]

    learning_rate = [0.01, 0.001, 0.0001]
    num_epochs_list = [10, 50, 100]
    hidden_layers_list = [2, 3, 4]
    hidden_neurons_list = [10, 50, 100]
    optimizers = ['adam', 'rmsprop', 'sgd']

    if os.path.exists('best_model.h5'):
        # Load the saved model
        best_model = load_model('best_model.h5')
        display_score(best_model, x_test, y_test, target_sample_rate)

    else:
        best_accuracy = 0
        best_model = None
        hyperparameter_data = []

        for i in range(len(learning_rate)):
            for num_hidden_neurons in hidden_neurons_list:
                for num_hidden_layers in hidden_layers_list:
                    model = create_model(input_shape, num_hidden_layers, num_hidden_neurons,
                    activation)

                    model = train_model(model, x_train, y_train, lr, num_epochs)
                    accuracy, precision, recall, f1 = evaluate_model(model, x_test, y_test)
                    combination_row_data = [lr, num_hidden_neurons, num_hidden_layers, num_hidden_neurons,
                    activation, accuracy, precision, recall, f1]
                    hyperparameter_data.append(combination_row_data)

                    if accuracy > best_accuracy:
                        best_accuracy = accuracy
                        best_model = model

        print(f"Learning Rate: {lr}, Epoch: {num_epochs}, Hidden Layers: {num_hidden_layers},
        Hidden Neurons: {num_hidden_neurons}, Activation: {activation}")
        print(f"Accuracy: {accuracy}, Precision: {precision}, Recall: {recall}, F1-Score: {f1}")
        print("*****")

    # Save the best model to a file
    save_model(best_model, 'best_model.h5')

    # Load the saved model
    loaded_model = load_model('best_model.h5')

    # Summary object methods
    summary = ModelSummary()
    sheet = summary.create

    # Summary data to file excel
    sheet['A1'] = 'Learning Rate'
    sheet['B1'] = 'Epoch'
    sheet['C1'] = 'Hidden Layers'
    sheet['D1'] = 'Hidden Neurons'
    sheet['E1'] = 'Activation'
    sheet['F1'] = 'Accuracy'
    sheet['G1'] = 'Precision'
    sheet['H1'] = 'Recall'
    sheet['I1'] = 'F1-Score'

    for row_index, row_data in enumerate(hyperparameter_data, start=1):
        # Menentukan style ke data dengan format paling tinggi
        for col_index in range(1, len(sheet.columns) + 1):
            cell = sheet.cell(row_index, col_index, row_data[col_index])
            cell.value = format(cell.value, ".4f")

    # Menyimpan file Excel
    workbook.save(filename="Data Kombinasi Hyper Parameter.xlsx")

    display_score(loaded_model, x_test, y_test, target_sample_rate)

    except:
        print("Error: Model Terbaik Tidak Ditemukan.")

    except:
        print("Error: Data tidak cukup untuk dibagi menjadi data training dan data testing.")

# Contoh penggunaan
if __name__ == '__main__':
    target_sample_rate = 44100 # Set the target sampling rate to 44100 for gait, sound, etc.
```

Gambar 2.2.1.4.2 main.

Fungsi **main** merupakan fungsi utama yang menjalankan alur aplikasi. Fungsi ini memiliki beberapa langkah untuk memproses data audio dan melatih model jaringan saraf. Pertama, direktori proyek dan direktori sampel audio diinisialisasi menggunakan **project\_directory** dan **audiosamples\_directory** dengan menggunakan modul `os` untuk menggabungkan path. Selanjutnya, data audio (**X**) dan label (**y**) dimuat menggunakan fungsi **load\_data** dengan mengambil file **.wav** dari direktori sampel audio. Setelah itu, beberapa variabel seperti **num\_classes**, **input\_shape**, **num\_samples**, dan **num\_test\_samples** diatur sesuai dengan konfigurasi yang telah ditentukan. Jika jumlah sampel dan jumlah sampel uji memenuhi syarat, maka data akan dibagi menjadi data latihan (**X\_train** dan **y\_train**) dan data pengujian (**X\_test** dan **y\_test**) menggunakan fungsi **train\_test\_split**. Selanjutnya, label dikonversi menjadi format one-hot encoding menggunakan fungsi **to\_categorical**. Terdapat beberapa list yang berisi variasi parameter yang akan digunakan dalam eksperimen, seperti **learning\_rates**, **num\_epochs\_list**, **hidden\_layers\_list**, **hidden\_neurons\_list**, dan **activations**. Selanjutnya, program akan memeriksa apakah file model terbaik (**best\_model.h5**) sudah ada. Jika ada, maka model tersebut akan dimuat dan aplikasi web akan dideploy menggunakan model tersebut. Jika tidak, maka akan dilakukan percobaan dengan variasi parameter yang telah ditentukan. Model-model akan dibuat menggunakan fungsi **create\_model** dan dilatih menggunakan fungsi **train\_model**. Setiap model akan dievaluasi menggunakan fungsi **evaluate\_model** untuk mendapatkan metrik evaluasi seperti akurasi, presisi,

recall, dan F1-score. Model dengan akurasi tertinggi akan disimpan sebagai **best\_model**. Setelah selesai melakukan percobaan, model terbaik akan disimpan dalam file (**best\_model.h5**) dan kemudian dimuat kembali. Aplikasi web akan dideploy menggunakan model yang telah disimpan. Jika tidak ada model terbaik yang ditemukan, akan dicetak pesan error. Dan jika jumlah sampel tidak memenuhi syarat, akan dicetak pesan error yang sesuai. Program juga akan menuliskan data secara otomatis ke file excel, dan menandai kombinasi hyperparameter dengan tingkat akurasi yang paling tinggi.

## 2.2.2 Tabel Data Kombinasi Hyperparameter

Learning Rate	Epochs	Hidden Layers	Hidden Neurons	Activation	Accuracy	Precision	Recall	F1-Score
0,01	50	1	32	sigmoid	0,806452	0,784232	0,855204	0,818182
0,01	50	1	32	softmax	0,81106	0,897143	0,710407	0,792929
0,01	50	1	32	tanh	0,792627	0,78355	0,819005	0,800885
0,01	50	1	64	sigmoid	0,831797	0,885417	0,769231	0,823245
0,01	50	1	64	softmax	0,815668	0,893855	0,723982	0,8
0,01	50	1	64	tanh	0,737327	0,680135	0,914027	0,779923
0,01	50	1	128	sigmoid	0,836406	0,797619	0,909502	0,849894
0,01	50	1	128	softmax	0,776498	0,942857	0,597285	0,731302
0,01	50	1	128	tanh	0,75576	0,697595	0,918552	0,792969
0,01	50	2	32	sigmoid	0,808756	0,792373	0,846154	0,818381
0,01	50	2	32	softmax	0,815668	0,794979	0,859729	0,826087
0,01	50	2	32	tanh	0,762673	0,707746	0,909502	0,79604
0,01	50	2	64	sigmoid	0,652074	0,972973	0,325792	0,488136
0,01	50	2	64	softmax	0,490783	1	0	0
0,01	50	2	64	tanh	0,670507	0,953488	0,371041	0,534202
0,01	50	2	128	sigmoid	0,776498	0,709459	0,950226	0,812379
0,01	50	2	128	softmax	0,490783	1	0	0
0,01	50	2	128	tanh	0,612903	0,570667	0,968326	0,718121
0,01	50	3	32	sigmoid	0,767281	0,730769	0,859729	0,790021
0,01	50	3	32	softmax	0,490783	1	0	0
0,01	50	3	32	tanh	0,799539	0,89881	0,683258	0,77635
0,01	50	3	64	sigmoid	0,822581	0,918605	0,714932	0,804071
0,01	50	3	64	softmax	0,490783	1	0	0
0,01	50	3	64	tanh	0,785714	0,81068	0,755656	0,782201
0,01	50	3	128	sigmoid	0,785714	0,728571	0,923077	0,814371
0,01	50	3	128	softmax	0,490783	1	0	0
0,01	50	3	128	tanh	0,698157	0,95	0,429864	0,5919
0,01	100	1	32	sigmoid	0,81106	0,7603	0,918552	0,831967
0,01	100	1	32	softmax	0,490783	1	0	0
0,01	100	1	32	tanh	0,509217	0,509217	1	0,674809
0,01	100	1	64	sigmoid	0,847926	0,827004	0,886878	0,855895
0,01	100	1	64	softmax	0,490783	1	0	0
0,01	100	1	64	tanh	0,806452	0,796537	0,832579	0,814159
0,01	100	1	128	sigmoid	0,841014	0,833333	0,859729	0,846325
0,01	100	1	128	softmax	0,509217	0,509217	1	0,674809
0,01	100	1	128	tanh	0,78341	0,734317	0,900452	0,808943
0,01	100	2	32	sigmoid	0,794931	0,735714	0,932127	0,822355
0,01	100	2	32	softmax	0,824885	0,822222	0,837104	0,829596
0,01	100	2	32	tanh	0,806452	0,796537	0,832579	0,814159
0,01	100	2	64	sigmoid	0,85023	0,882353	0,81448	0,847059
0,01	100	2	64	softmax	0,490783	1	0	0
0,01	100	2	64	tanh	0,781106	0,740458	0,877828	0,803313
0,01	100	2	128	sigmoid	0,817972	0,938272	0,687783	0,793734
0,01	100	2	128	softmax	0,490783	1	0	0
0,01	100	2	128	tanh	0,806452	0,915152	0,683258	0,782383
0,01	100	3	32	sigmoid	0,847926	0,881773	0,809955	0,84434
0,01	100	3	32	softmax	0,490783	1	0	0
0,01	100	3	32	tanh	0,815668	0,813333	0,828054	0,820628
0,01	100	3	64	sigmoid	0,827189	0,820175	0,846154	0,832962

0,01	100	3	64	softmax	0,490783	1	0	0
0,01	100	3	64	tanh	0,815668	0,902857	0,714932	0,79798
0,01	100	3	128	sigmoid	0,852535	0,894472	0,80543	0,847619
0,01	100	3	128	softmax	0,490783	1	0	0
0,01	100	3	128	tanh	0,78341	0,739623	0,886878	0,806584
0,01	200	1	32	sigmoid	0,820276	0,923077	0,705882	0,8
0,01	200	1	32	softmax	0,490783	1	0	0
0,01	200	1	32	tanh	0,799539	0,770161	0,864253	0,814499
0,01	200	1	64	sigmoid	0,845622	0,826271	0,882353	0,853392
0,01	200	1	64	softmax	0,490783	1	0	0
0,01	200	1	64	tanh	0,799539	0,781513	0,841629	0,810458
0,01	200	1	128	sigmoid	0,852535	0,852018	0,859729	0,855856
0,01	200	1	128	softmax	0,824885	0,816594	0,846154	0,831111
0,01	200	1	128	tanh	0,827189	0,857843	0,791855	0,823529
0,01	200	2	32	sigmoid	0,836406	0,826087	0,859729	0,842572
0,01	200	2	32	softmax	0,808756	0,7875	0,855204	0,819957
0,01	200	2	32	tanh	0,831797	0,885417	0,769231	0,823245
0,01	200	2	64	sigmoid	0,845622	0,866667	0,823529	0,844548
0,01	200	2	64	softmax	0,827189	0,778626	0,923077	0,84472
0,01	200	2	64	tanh	0,829493	0,876923	0,773756	0,822115
0,01	200	2	128	sigmoid	0,836406	0,8	0,904977	0,849257
0,01	200	2	128	softmax	0,820276	0,862944	0,769231	0,813397
0,01	200	2	128	tanh	0,751152	0,682848	0,954751	0,796226
0,01	200	3	32	sigmoid	0,841014	0,868932	0,809955	0,838407
0,01	200	3	32	softmax	0,490783	1	0	0
0,01	200	3	32	tanh	0,804147	0,769841	0,877828	0,820296
0,01	200	3	64	sigmoid	0,829493	0,906077	0,742081	0,81592
0,01	200	3	64	softmax	0,490783	1	0	0
0,01	200	3	64	tanh	0,827189	0,854369	0,79638	0,824356
0,01	200	3	128	sigmoid	0,834101	0,811715	0,877828	0,843478
0,01	200	3	128	softmax	0,490783	1	0	0
0,01	200	3	128	tanh	0,822581	0,836449	0,809955	0,822989
0,001	50	1	32	sigmoid	0,774194	0,766234	0,800905	0,783186
0,001	50	1	32	softmax	0,490783	1	0	0
0,001	50	1	32	tanh	0,817972	0,81982	0,823529	0,82167
0,001	50	1	64	sigmoid	0,806452	0,791489	0,841629	0,815789
0,001	50	1	64	softmax	0,490783	1	0	0
0,001	50	1	64	tanh	0,718894	0,660194	0,923077	0,769811
0,001	50	1	128	sigmoid	0,822581	0,790323	0,886878	0,835821
0,001	50	1	128	softmax	0,788018	0,732852	0,918552	0,815261
0,001	50	1	128	tanh	0,83871	0,903743	0,764706	0,828431
0,001	50	2	32	sigmoid	0,762673	0,773148	0,755656	0,764302
0,001	50	2	32	softmax	0,509217	0,509217	1	0,674809
0,001	50	2	32	tanh	0,78341	0,832461	0,719457	0,771845
0,001	50	2	64	sigmoid	0,764977	0,764444	0,778281	0,7713
0,001	50	2	64	softmax	0,509217	0,509217	1	0,674809
0,001	50	2	64	tanh	0,785714	0,725352	0,932127	0,815842
0,001	50	2	128	sigmoid	0,781106	0,754032	0,846154	0,797441
0,001	50	2	128	softmax	0,490783	1	0	0
0,001	50	2	128	tanh	0,820276	0,771863	0,918552	0,838843

0,001	50	3	32	sigmoid	0,822581	0,821429	0,832579	0,826966
0,001	50	3	32	softmax	0,490783	1	0	0
0,001	50	3	32	tanh	0,808756	0,813636	0,809955	0,811791
0,001	50	3	64	sigmoid	0,790323	0,815534	0,760181	0,786885
0,001	50	3	64	softmax	0,490783	1	0	0
0,001	50	3	64	tanh	0,797235	0,88	0,696833	0,777778
0,001	50	3	128	sigmoid	0,769585	0,776256	0,769231	0,772727
0,001	50	3	128	softmax	0,490783	1	0	0
0,001	50	3	128	tanh	0,836406	0,850467	0,823529	0,836782
0,001	100	1	32	sigmoid	0,83871	0,847926	0,832579	0,840183
0,001	100	1	32	softmax	0,490783	1	0	0
0,001	100	1	32	tanh	0,843318	0,843049	0,850679	0,846847
0,001	100	1	64	sigmoid	0,824885	0,80083	0,873303	0,835498
0,001	100	1	64	softmax	0,822581	0,770677	0,927602	0,841889
0,001	100	1	64	tanh	0,831797	0,873737	0,782805	0,825776
0,001	100	1	128	sigmoid	0,836406	0,850467	0,823529	0,836782
0,001	100	1	128	softmax	0,776498	0,708054	0,954751	0,813102
0,001	100	1	128	tanh	0,845622	0,823529	0,886878	0,854031
0,001	100	2	32	sigmoid	0,762673	0,770642	0,760181	0,765376
0,001	100	2	32	softmax	0,490783	1	0	0
0,001	100	2	32	tanh	0,808756	0,776	0,877828	0,823779
0,001	100	2	64	sigmoid	0,797235	0,792952	0,81448	0,803571
0,001	100	2	64	softmax	0,490783	1	0	0
0,001	100	2	64	tanh	0,769585	0,941606	0,58371	0,72067
0,001	100	2	128	sigmoid	0,799539	0,79646	0,81448	0,805369
0,001	100	2	128	softmax	0,490783	1	0	0
0,001	100	2	128	tanh	0,845622	0,846847	0,850679	0,848758
0,001	100	3	32	sigmoid	0,792627	0,788546	0,809955	0,799107
0,001	100	3	32	softmax	0,490783	1	0	0
0,001	100	3	32	tanh	0,753456	0,959677	0,538462	0,689855
0,001	100	3	64	sigmoid	0,785714	0,778261	0,809955	0,793792
0,001	100	3	64	softmax	0,490783	1	0	0
0,001	100	3	64	tanh	0,827189	0,768382	0,945701	0,84787
0,001	100	3	128	sigmoid	0,804147	0,809091	0,80543	0,807256
0,001	100	3	128	softmax	0,490783	1	0	0
0,001	100	3	128	tanh	0,801843	0,747253	0,923077	0,825911
0,001	200	1	32	sigmoid	0,845622	0,818182	0,895928	0,855292
0,001	200	1	32	softmax	0,490783	1	0	0
0,001	200	1	32	tanh	0,829493	0,84507	0,81448	0,829493
0,001	200	1	64	sigmoid	0,854839	0,891089	0,81448	0,851064
0,001	200	1	64	softmax	0,490783	1	0	0
0,001	200	1	64	tanh	0,824885	0,808511	0,859729	0,833333
0,001	200	1	128	sigmoid	0,847926	0,835498	0,873303	0,853982
0,001	200	1	128	softmax	0,824885	0,939394	0,701357	0,803109
0,001	200	1	128	tanh	0,845622	0,818182	0,895928	0,855292
0,001	200	2	32	sigmoid	0,813364	0,830189	0,79638	0,812933
0,001	200	2	32	softmax	0,813364	0,853535	0,764706	0,806683
0,001	200	2	32	tanh	0,843318	0,846154	0,846154	0,846154
0,001	200	2	64	sigmoid	0,824885	0,779923	0,914027	0,841667
0,001	200	2	64	softmax	0,490783	1	0	0

0,001	200	2	64	tanh	0,85023	0,827731	0,891403	0,858388
0,001	200	2	128	sigmoid	0,827189	0,809322	0,864253	0,835886
0,001	200	2	128	softmax	0,490783	1	0	0
0,001	200	2	128	tanh	0,827189	0,804167	0,873303	0,83731
0,001	200	3	32	sigmoid	0,808756	0,82243	0,79638	0,809195
0,001	200	3	32	softmax	0,490783	1	0	0
0,001	200	3	32	tanh	0,801843	0,768924	0,873303	0,817797
0,001	200	3	64	sigmoid	0,788018	0,772152	0,828054	0,799127
0,001	200	3	64	softmax	0,490783	1	0	0
0,001	200	3	64	tanh	0,794931	0,73913	0,923077	0,820926
0,001	200	3	128	sigmoid	0,769585	0,761905	0,79638	0,778761
0,001	200	3	128	softmax	0,490783	1	0	0
0,001	200	3	128	tanh	0,845622	0,826271	0,882353	0,853392
0,0001	50	1	32	sigmoid	0,728111	0,726872	0,746606	0,736607
0,0001	50	1	32	softmax	0,509217	0,509217	1	0,674809
0,0001	50	1	32	tanh	0,774194	0,780822	0,773756	0,777273
0,0001	50	1	64	sigmoid	0,78341	0,763485	0,832579	0,796537
0,0001	50	1	64	softmax	0,490783	1	0	0
0,0001	50	1	64	tanh	0,792627	0,791111	0,80543	0,798206
0,0001	50	1	128	sigmoid	0,776498	0,776786	0,78733	0,782022
0,0001	50	1	128	softmax	0,781106	0,746094	0,864253	0,800839
0,0001	50	1	128	tanh	0,797235	0,780591	0,837104	0,80786
0,0001	50	2	32	sigmoid	0,751152	0,738397	0,791855	0,764192
0,0001	50	2	32	softmax	0,493088	1	0,004525	0,009009
0,0001	50	2	32	tanh	0,799539	0,788793	0,828054	0,807947
0,0001	50	2	64	sigmoid	0,493088	1	0,004525	0,009009
0,0001	50	2	64	softmax	0,490783	1	0	0
0,0001	50	2	64	tanh	0,813364	0,833333	0,791855	0,812065
0,0001	50	2	128	sigmoid	0,767281	0,745902	0,823529	0,782796
0,0001	50	2	128	softmax	0,490783	1	0	0
0,0001	50	2	128	tanh	0,813364	0,804348	0,837104	0,820399
0,0001	50	3	32	sigmoid	0,509217	0,509217	1	0,674809
0,0001	50	3	32	softmax	0,490783	1	0	0
0,0001	50	3	32	tanh	0,771889	0,790476	0,751131	0,770302
0,0001	50	3	64	sigmoid	0,509217	0,509217	1	0,674809
0,0001	50	3	64	softmax	0,509217	0,509217	1	0,674809
0,0001	50	3	64	tanh	0,769585	0,778802	0,764706	0,771689
0,0001	50	3	128	sigmoid	0,467742	0,416667	0,113122	0,177936
0,0001	50	3	128	softmax	0,490783	1	0	0
0,0001	50	3	128	tanh	0,778802	0,77533	0,79638	0,785714
0,0001	100	1	32	sigmoid	0,59447	0,578947	0,746606	0,652174
0,0001	100	1	32	softmax	0,490783	1	0	0
0,0001	100	1	32	tanh	0,737327	0,731602	0,764706	0,747788
0,0001	100	1	64	sigmoid	0,806452	0,791489	0,841629	0,815789
0,0001	100	1	64	softmax	0,509217	0,509217	1	0,674809
0,0001	100	1	64	tanh	0,71659	0,704167	0,764706	0,733189
0,0001	100	1	128	sigmoid	0,792627	0,78355	0,819005	0,800885
0,0001	100	1	128	softmax	0,509217	0,509217	1	0,674809
0,0001	100	1	128	tanh	0,820276	0,823529	0,823529	0,823529
0,0001	100	2	32	sigmoid	0,767281	0,780374	0,755656	0,767816



0,0001	100	2	32	softmax	0,490783	1	0	0
0,0001	100	2	32	tanh	0,799539	0,779167	0,846154	0,81128
0,0001	100	2	64	sigmoid	0,74424	0,72541	0,800905	0,76129
0,0001	100	2	64	softmax	0,486175	0,25	0,004525	0,008889
0,0001	100	2	64	tanh	0,771889	0,758475	0,809955	0,78337
0,0001	100	2	128	sigmoid	0,762673	0,770642	0,760181	0,765376
0,0001	100	2	128	softmax	0,490783	1	0	0
0,0001	100	2	128	tanh	0,827189	0,804167	0,873303	0,83731
0,0001	100	3	32	sigmoid	0,578341	0,548718	0,968326	0,700491
0,0001	100	3	32	softmax	0,509217	0,509217	1	0,674809
0,0001	100	3	32	tanh	0,78341	0,741445	0,882353	0,805785
0,0001	100	3	64	sigmoid	0,725806	0,738318	0,714932	0,726437
0,0001	100	3	64	softmax	0,509217	0,509217	1	0,674809
0,0001	100	3	64	tanh	0,804147	0,790598	0,837104	0,813187
0,0001	100	3	128	sigmoid	0,774194	0,766234	0,800905	0,783186
0,0001	100	3	128	softmax	0,509217	0,509217	1	0,674809
0,0001	100	3	128	tanh	0,808756	0,8	0,832579	0,815965
0,0001	200	1	32	sigmoid	0,758065	0,75	0,78733	0,768212
0,0001	200	1	32	softmax	0,511521	0,510393	1	0,675841
0,0001	200	1	32	tanh	0,822581	0,813043	0,846154	0,829268
0,0001	200	1	64	sigmoid	0,760369	0,76	0,773756	0,766816
0,0001	200	1	64	softmax	0,490783	1	0	0
0,0001	200	1	64	tanh	0,824885	0,80083	0,873303	0,835498
0,0001	200	1	128	sigmoid	0,781106	0,769231	0,81448	0,791209
0,0001	200	1	128	softmax	0,490783	1	0	0
0,0001	200	1	128	tanh	0,824885	0,811159	0,855204	0,832599
0,0001	200	2	32	sigmoid	0,753456	0,752212	0,769231	0,760626
0,0001	200	2	32	softmax	0,490783	1	0	0
0,0001	200	2	32	tanh	0,797235	0,787879	0,823529	0,80531
0,0001	200	2	64	sigmoid	0,78341	0,777293	0,80543	0,791111
0,0001	200	2	64	softmax	0,490783	1	0	0
0,0001	200	2	64	tanh	0,788018	0,789238	0,79638	0,792793
0,0001	200	2	128	sigmoid	0,806452	0,796537	0,832579	0,814159
0,0001	200	2	128	softmax	0,490783	1	0	0
0,0001	200	2	128	tanh	0,85023	0,854545	0,850679	0,852608
0,0001	200	3	32	sigmoid	0,573733	0,663636	0,330317	0,441088
0,0001	200	3	32	softmax	0,490783	1	0	0
0,0001	200	3	32	tanh	0,797235	0,780591	0,837104	0,80786
0,0001	200	3	64	sigmoid	0,813364	0,893258	0,719457	0,796992
0,0001	200	3	64	softmax	0,490783	1	0	0
0,0001	200	3	64	tanh	0,797235	0,798206	0,80543	0,801802
0,0001	200	3	128	sigmoid	0,790323	0,806604	0,773756	0,789838
0,0001	200	3	128	softmax	0,509217	0,509217	1	0,674809
0,0001	200	3	128	tanh	0,799539	0,768	0,868778	0,815287

*Tabel 2.2.2.1 Data Kombinasi Hyperparameter*

Berdasarkan tabel tersebut, tingkat akurasi tertinggi yakni 85% dan terdapat pada learning rate 0,001, epocsh 200, hidden layers 1, hidden neurons 64, activation sigmoid, accuracy 0,854838709677419, precision 0,891089108910891, recall 0,81447963800905, dan F1-Score 0,851063829787234.

### **BAB III**

#### **PENUTUP**

Pada sistem ini digunakan metode ekstraksi fitur MFCC untuk menentukan nilai fitur dari data audio. Terdapat dua sentiment yang diuji pada percobaan ini yaitu sentimen positif (atau emosi happy) dan sentimen negatif (atau emosi sad). Dataset tersebut dibagi menjadi dua bagian, yakni 80% untuk dataset training dan 20% untuk dataset testing. Berdasarkan uji coba yang telah dilakukan, sistem ini mencapai tingkat akurasi sekitar 85% pada learning rate 0,001, epoch 200, hidden layers 1, hidden neurons 64, activation sigmoid, accuracy 0,854838709677419, precision 0,891089108910891, recall 0,81447963800905, dan F1-Score 0,851063829787234.

## REFERENSI

- [1] M. N. Fauzan, "Identifikasi Audio Ancaman Menggunakan Metode Convolutional Neural Network," *Jurnal Sistem dan Teknologi Informasi*, vol. 10, no. 4, pp. 446–452, 2022.
- [2] S. Amalia, "Pengenalan Digit 0 Sampai 9 Menggunakan Ekstraksi Ciri MFCC dan Jaringan Syaraf Tiruan Backpropagation," *Jurnal Teknik Elektro ITP*, vol. 6, no. 1, pp. 1–8, Jan. 2017, doi: 10.21063/jte.2017.3133601.
- [3] Jayanta, Henki Bayu Seta, and Ridho Zulfahmi, "PENERAPAN PEMISAHAN SUARA BERDASARKAN CIRI SUARA MENGGUNAKAN JARINGAN SARAF TIRUAN," *Jurnal Ilmiah MATRIK*, vol. 23, no. 2, pp. 209–218, 2021.