BAB II
LANDASAN TEORI

Bab ini menjelaskan analisis berbagai teori dan hasil penelitian yang relevan dengan masalah yang akan diteliti.

2.1 *Speech Recognition* (Pengenalan Suara)

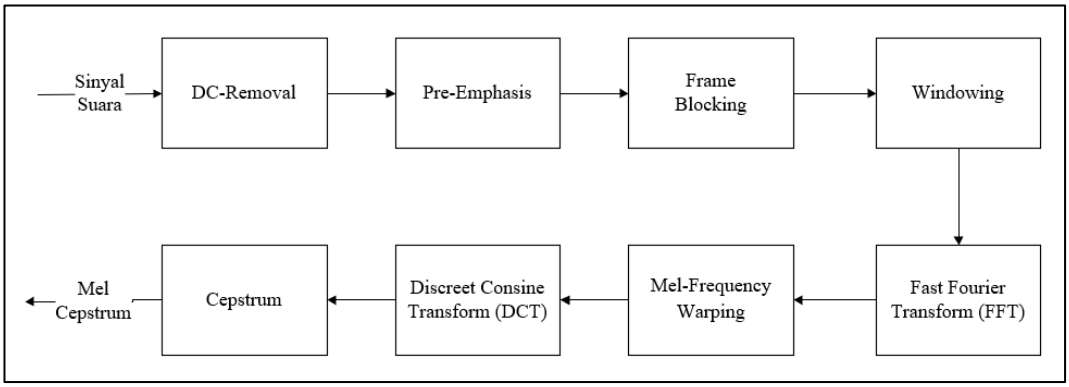
Speech recognition atau pengenalan suara memungkinkan sistem ataupun personal komputer untuk memahami ucapan pengguna dan mengubahnya menjadi serangkaian kata melalui program komputer, sehingga menciptakan semacam komunikasi alami antara manusia dan mesin . Pengenalan suara dapat diaplikasikan seperti pelayanan suara yang populer yaitu *Google Assistant* dan *Cortana* (Benkerzaz, Elmir, & Dennai, 2019).

Jenis-jenis pelafalan kata :

a. Kata yang terisolasi: Pelafalan kata jenis ini membutuhkan kondisi yang tenang (tidak adanya *noise*) diantara pelafalan antar kata.
b. Kata yang terhubung: Pelafalan kata ini mirip dengan kata yang terisolasi, perbedaannya jenis pelafalan ini hanya membutuhkan jeda minimum di antara kata-kata tersebut.
c. Kata yang berkelanjutan: Pelafalan kata ini berbicara hampir seperti normal dengan jeda waktu yang sangat sedikit atau bahkan tanpa jeda waktu.
d. Kata yang spontan: Pelafalan kata yang dapat mengenali kata-kata yang diucapkan secara alami tanpa adanya jeda waktu antar kata.

2.2 *Mel Frequency Cepstral Coefficient* (MFCC)

MFCC merupakan metode ekstraksi ciri yang digunakan untuk mengubah sinyal suara menjadi beberapa parameter dan merupakan metode yang populer digunakan sebagai ekstraksi ciri suara. Metode MFCC dapat menangkap karakteristik atau informasi-informasi yang penting dari suara untuk proses pengenalan suara dan menghasilkan data dari suara seminimal mungkin tanpa menghilangkan karakteristik atau informasi-informasi penting di dalamnya (Manunggal, 2005).



Gambar 2.2 Blok Diagram MFCC

2.2.1 *DC-Removal*

Tujuan dari *dc-removal* adalah menormalisasi data dari suara yang akan di proses (Indrawaty, Dewi, & Lukman, 2019). Persamaan proses *dc-removal* dapat dituliskan sebagai berikut :

$$y[n] = x[n] - \bar{x}, 0 \leq n \leq N - 1 \quad (2.1)$$

Dimana :

y[n] = sampel sinyal hasil *dc-removal*

x[n] = sampel sinyal asli

$\bar{x}$ = nilai rata-rata sampel sinyal asli

N = panjang sinyal

2.2.2 *Pre-emphasis*

Filter *pre-emphasis* berguna dalam menyeimbangkan spektrum frekuensi karena frekuensi tinggi biasanya memiliki magnitudo lebih kecil dibandingkan dengan frekuensi yang lebih rendah. Filter *pre-emphasis* juga berguna dalam mengurangi *noise* yang ada pada suara (Indrawaty, Dewi, & Lukman, 2019). Persamaan filter *pre-emphasis* dapat dituliskan sebagai berikut :

$$y(n) = x(n) - \alpha x(n-1) \quad (2.2)$$

Dimana :

y(n) = sinyal suara hasil proses *pre-emphasis*.

x(n) = sinyal suara sebelum proses *pre-emphasis*.

α = nilai yang paling sering digunakan ialah 0.95 atau 0.97

Setelah diketahui hasil *pre-emphasis*, lalu data sinyal suara sebelum proses *pre-emphasis* ditambah dengan data sinyal suara setelah proses *pre-emphasis*. Dapat dituliskan dalam persamaan sebagai berikut :

$$Xn = Nn + Yn \quad (2.3)$$

Dimana :

Xn = hasil *pre-emphasis* baru.

Nn = hasil *pre-emphasis* lama.

Yn = hasil sinyal dari *pre-emphasis*.

2.2.3 *Frame Blocking*

Setelah *pre-emphasis*, perlu membagi sinyal menjadi *frame* waktu singkat. Alasan di balik ini adalah frekuensi dalam sinyal berubah dari waktu ke waktu, sehingga sinyal suara diproses secara *short segments*. Panjang *frame* yang digunakan untuk kebutuhan ekstraksi ciri umumnya sekitar 10-30ms. Proses *frame blocking* ini dilakukan secara terus menerus sampai seluruh sinyal dapat diproses dan dilakukan secara *overlapping* untuk setiap *body frame*-nya. Dilakukannya o*verlapping* ini berfungsi untuk menghindari hilangnya ciri atau karakteristik dari suara pada perbatasan perpotongan setiap *frame* (Nurhasanah, Barmawi, & David, 2016). Persamaan proses *frame blocking* ini dapat dituliskan sebagai berikut :

$$Jumlah\ frame = ((I - N)/M + 1) \quad (2.4)$$

Dimana :

I = *sample rate*

N = *sample point* (*sample rate**waktu *framing*)

M = N/2

2.2.4 *Windowing*

Proses *windowing* dilakukan untuk menghindari adanya kebocoran spektral atau disebut juga dengan *aliasing*. *Aliasing* ini merupakan sinyal baru yang memiliki frekuensi berbeda dengan sinyal aslinya. Hal ini bisa terjadi dikarenakan rendahnya jumlah dari *sampling rate* atau bisa juga terjadi karena proses *frame*

blocking sebelumnya yang dapat menyebabkan sinyal menjadi *discontinue* (Nurhasanah, Barmawi, & David, 2016).

Pertama, dilakukan proses *windowing* menggunakan jendela *Hamming* dengan persamaan sebagai berikut :

$$w(n) = 0.54 - 0.46\cos = \frac{2\pi n}{M-1} \quad (2.5)$$

Dimana:

n = 0,1,2,3,…, M-1

M = jumlah sample pada masing- masing *frame*

Setelah menggunakan *Hamming*, dilakukan proses *windowing* sebagai berikut :

$$x(n) = xi(n) * w(n)\,; n = 0.1, \dots, N-1 \quad (2.6)$$

Dimana:

x(n) = nilai sampel sinyal

xi(n) = nilai sampel sinyal *frame* 1

w(n) = fungsi *windowing*

2.2.5 *Fast Fourier Transform (FFT)*

Fast Fourier Transform (FFT) bertujuan untuk merubah data sinyal suara dari domain waktu menjadi sinyal suara di domain frekuensi untuk kebutuhan analisa *fourier* (Dewi, Zulkarnain, & Lestari, 2018). Proses FFT dapat dituliskan sebagai berikut berikut:

$$F(k) = \sum_{n=1}^{N} f[n]\cos\left(\frac{2\pi nkT}{N}\right) - j\sum_{n=1}^{N} f[n]\,sin\left(\frac{2\pi nkT}{N}\right) \quad (2.7)$$

Dimana :

F(k)= *Fourier Form Transform*

F(n) = sampel data

k = 0,1,2,…,(N-1)

T = hasil *windowing*

2.2.6 *Mel-Frequency Warping*

Mel-Frequency Warping menggunakan *filterbank* untuk dapat mengetahui ukuran energi dari *frequency band* tertentu dengan menentukan batas atas dan bawah filter dalam sinyal suara dan untuk menyaring sinyal suara yang telah diubah dari domain waktu ke dalam bentuk domain frekuensi (Nurhasanah, Dewi, & Saputro, 2018).

Sebelum melakukan proses *filterbank*, pertama-tama harus mencari nilai-nilai koefisien dari *filterbank* terlebih dahulu sebagai berikut :

$$H_i = 2595 * \log(1 + 1000/700)/S_i/2 \quad (2.8)$$

Dimana :

Hi = koefisien *filterbank*

Si = nilai hasil dari proses FFT

Lalu menghitung *filterbank* :

$$Y[t] = \sum_{j=1}^{N} S[j]H_i[j] \quad (2.9)$$

Dimana :

N = jumlah *magnitude spectrum*

S[j] = *magnitude spectrum* pada frekuensi j

H_i[j] = koefisien *filterbank* pada frekuensi j

M = jumlah *channel* dalam *filterbank*

2.2.7 *Discreet Consine Transform (DCT)*

Konsep dasar dari DCT adalah agar nilai *mel* dapat di konversikan kembali ke dalam domain waktu (Widodo, Siswanto, & Sudjana, 2016).

$$C_n = \sum_{k=1}^{K} (\log S_k) \cos\left[n(k - \frac{1}{2})\frac{\pi}{K}\right] \; n = 1,2, \dots, \mathrm{K} \quad (2.10)$$

Dimana :

Sk = keluaran proses *filterbank* pada index k

K = jumlah koefisien yang diharapkan

2.2.8 *Cepstrum*

Cepstrums merupakan hasil akhir dari proses MFCC, digunakan untuk memperoleh informasi dari sinyal suara ucapan dan meningkatkan kualitas pengenalan suara, maka dari itu *cepstrums* harus melewati proses *cepstral liftering* (Nurhasanah, Dewi, & Saputro, 2018).

$$W(n) = \{C_n + \frac{L}{2}\ sin\ (\frac{n\pi}{L}), n = 1,2, \dots, L \quad (2.11)$$

Dimana :

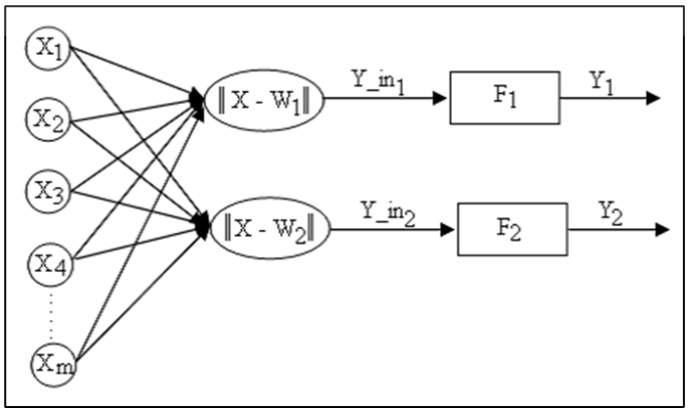
L = jumlah *cepstral coefficients*

N = indeks dari *cepstral coefficients*

C_n = nilai hasil DCT

2.3 *Learning Vector Quantization*

Metode *Learning Vector Quantization* bertujuan untuk melakukan pembelajaran pada lapisan kompetitif yang terawasi. Lapisan kompetitif ini akan dapat secara otomatis mempelajari inputan untuk mengklasifikasikan vektor-vektor berdasarkan jarak antara vektor-vektor input dengan bobot menggunakan *euclidean distance*. Jika jarak antara vektor input dan vektor bobot diketahui mendekati sama, maka lapisan kompetitif tersebut akan meletakkan kedua vektor tersebut ke dalam *codebook* yang sama.



Gambar 2.3 Arsitektur LVQ

Arsitektur pada Gambar 2.2 merupakan arsitektur *Learning Vector Quantization* yang memiliki vektor *input* X_1, X_2, X_3, …, X_m dengan 2 buah kelas. W_1 dan W_2 adalah bobot untuk menghitung jarak antara vektor input dan vektor

bobot. Pada F_1, jika $|X\text{-}W_1| < |X\text{-}W_2|$ maka vektor akan dipetakan ke $Y_1=1$ dan dipetakan ke $Y_1=0$ jika sebaliknya. Seperti persamaan sebagai berikut :

$$C_j = \sqrt{\sum_{i=1}^{n}(x_i - w_j)^2} \quad (2.12)$$

Dimana :

C_j = nilai jarak *euclidean*

X_i = nilai vektor *input*

W_j = nilai vektor bobot

Metode ini juga mempelajari sebagian kecil vektor corak untuk menandai karekteristik spesifik pembicara agar lebih efesien dan tidak bergantung pada seluruh dataset. Sebagian kecil vektor corak tersebut dikelempokan menjadi sebuah *codebook* untuk masing-masing pembicara. Dalam hal pengujian, suara yang diujikan dibandingkan dengan *codebook* dari tiap pembicara dan mengukur perbedaan dari suara yang di ujikan dengan *codebook*. Setelah diketahui perbedaannya, kemudian dilakukan keputusan dari identifikasi suara berdasarkan perbedaan tersebut.

Selain itu LVQ selalu memperbaharui bobotnya sesuai jumlah maksimal *epoch* yang ditetapkan. Epoch yaitu satu siklus pengujian yang melibatkan semua pola. Rumus bobot LVQ yang digunakan untuk memperbaharui nilai bobot sebagai berikut :

$$\overrightarrow{W_m} \leftarrow \overrightarrow{W_m} + \alpha \cdot (\vec{X} - \overrightarrow{W_m}) \quad (2.13)$$

Dimana :

$\overrightarrow{W_m}$ = bobot

α = *learning rate*

$\vec{X}$ = *input*

2.4 Pengujian Kinerja Sistem

Pengujian kinerja sistem dilakukan dengan cara mengukur *accuracy*, *precision*, *recall* dan *F Measure*. *Accuracy* merupakan prediksi benar, baik benar positif maupun benar negatif dengan keseluruhan data. *Precision* merupakan data

benar positif dibandingkan dengan kesulurahan hasil yang diprediksi positif, *recall* merupakan data benar positif dibandingkan dengan keseluruhan data yang benar positif, *F Measure* merupakan ukuran akurasi uji dari perhitungan *precision* dan *recall* (Pardede & Husada, 2016).

Tabel 2.1 Tabel *Confussion Matrix*

		Hasil Sebenarnya	
		Positive	Negative
Hasil Prediksi	Positive	TP (True Positive)	FP (False Positive)
	Negative	FN (False Negative)	TN (True Negative)

Untuk mengukur kinerja sistem menggunakan persamaan berikut :

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \; x \; 100\% \quad (2.14)$$

$$Precision = \frac{TP}{TP + FP} \; x \; 100\% \quad (2.15)$$

$$Recall = \frac{TP}{TP + FN} \; x \; 100\% \quad (2.16)$$

$$F\ Measure = 2 \; x \; \frac{(Precision \; x \; Recall)}{(Precision + Recall)} \quad (2.17)$$

Dimana :

TP (*True Positive*) = memprediksi jumlah aksen yang benar sesuai pengujian

FP (*False Positive*) = memprediksi jumlah aksen yang salah terdeteksi benar

FN (*False Negative*) = memprediksi jumlah aksen yang salah klasifikasi

TN (*True Negative*) = memprediksi jumlah aksen yang salah dari pengujian

Pada penelitian ini memiliki 2 pengujian, pengujian aksen British dan pengujian aksen non British. Pengujian pertama menggunakan sampel suara aksen

British, *true positive* yang digunakan merupakan aksen British dan *true negative* merupakan aksen non British. Pengujian kedua menggunakan sampel suara aksen non British, *true positive* yang digunakan merupakan aksen non British dan *true negative* merupakan aksen British.