PENGENALAN UCAPAN HURUF HIJAIYAH MENGGUNAKAN MEL FREQUENCY CEPSTRAL COEFFICIENTS (MFCC) DAN HIDDEN MARKOV MODEL (HMM)

Rifan Muhamad Fauzi[1], Adiwijaya[2], Warih Maharani[3]

[1]Teknik Informatika, Fakultas Teknik Informatika, Universitas Telkom

Abstrak

Belajar huruf hijaiyah merupakan tahap awal seseorang untuk dapat membaca Al-Quran. Proses ini biasanya dilakukan seorang pembelajar beserta pembimbing yang berfungsi untuk mengenalkan dan mengajari pembelajaran huruf hijaiyah. Speech recognition merupakan sistem yang digunakan untuk memroses sinyal suara menjadi data sehingga dapat dikenali oleh komputer. Dengan memanfaatkan sistem ini, diharapkan peran seorang pembimbing dalam mengenalkan dan mengoreksi pelafalan huruf hijaiyah dapat tergantikan sehingga proses belajar membaca huruf hijaiyah dapat dilakukan lebih mandiri.

Permasalahan pengenalan huruf hijaiyah dapat diselesaikan dengan menggunakan Mel Frequency Cepstrum Coefficients (MFCC) untuk melakukan proses ekstraksi ciri setiap sinyal suara dan Hidden Markov Model (HMM) untuk membentuk model dan melakukan klasifikasi suara. Secara umum, sistem pengenalan suara memiliki dua proses yang penting yaitu testing dan training. Training merupakan proses dimana sistem melakukan pembentukan model suara dengan cara melakukan pelatihan menggunakan sejumlah sample suara. Testing merupakan proses dimana sistem melakukan klasifikasi/pelabelan pada sebuah sinyal suara.

Setelah dilakukan pengujian terhadap sistem dengan beberapa skenario, diperoleh akurasi terbaik yaitu 67.75% dalam mengenali 50 kata. Akurasi ini diperoleh dari hasil pengujian dengan sample rate 16kHz, ukuran codebook 64 dan 5 state HMM.

Kata Kunci : MFCC, HMM, Speech Recognition, Hijaiyah

Abstract

Learning hijaiyah letter is the initial stage for someone to be able to read the Quran. This process is usually done by a learner and their mentors that serve to introduce and teach lessons hijaiyah letters. Speech recognition is a system used to process voice signals into data that can be recognized by the computer. By utilizing this system, the expected role of a mentor in introducing and correct pronunciation of the letters can be replaced so hijaiyah learning to read letters hijaiyah do more independent. By utilizing this system, the role of a mentor in introducing and correct pronunciation of the Hijaiyah letters can be replaced so hijaiyah learning process can be performed more independent.

Hijaiyah letter recognition problems can be solved by using the Mel Frequency cepstrum Coefficients (MFCC) for extracting feature of voice signal and Hidden Markov Model (HMM) for building voice and performing voice classification. In general, the voice recognition system has two important processes, testing and training. Training is a process of building model from each voice. Testing is a process classification / labeling of a voice signal.

After testing the system with a several scenarios, the best accuracy obtained is 67.75% in recognizing 50 words. Accuracy is obtained from the testing with 16kHz sample rate, codebook 64 and 5 state in HMM.

Keywords : MFCC, HMM, Speech Recognition, Hijaiyah

1. Pendahuluan

1.1 Latar belakang masalah

Huruf hijaiyah merupakan huruf yang digunakan pada Al-Quran. seseorang perlu belajar huruf hijaiyah beserta ilmu tajwid untuk dapat membaca dan memahami Al-Quran. Belajar huruf hijaiyah merupakan tahap paling awal dalam proses pembelajaran Al-Quran. Dalam proses tersebut tentunya masih diperlukan pembimbing khusus yang sudah mampu membaca Al-Quran untuk mengenalkan dan mengajarkan huruf hijaiyah pada pembelajar. Atas dasar permasalahan ini penulis mencoba membuat sistem yang mampu menggantikan peran pembimbing sebagai korektor pembelajar dalam pembacaan huruf hijaiyah.

Speech recognition merupakan suatu teknologi yang digunakan untuk mengenali suara dan mengubahnya menjadi sebuah representasi data yang dapat dimengerti oleh komputer. Dengan *speech recognition system* ini diharapkan peran seorang pembimbing/korektor dalam pembelajaran huruf hijaiyah dapat tergantikan sehingga pembelajaran dapat dilakukan secara mandiri dan lebih fleksibel.

Secara umum proses sistem pengenalan suara dapat dibagi menjadi dua bagian, yaitu *training* dan *testing*[6]. *Training* merupakan proses pembentukkan model kata dari sejumlah sinyal suara, sedangkan *testing* merupakan proses pelabelan sinyal suara yang dilakukan oleh sistem sesuai dengan model yang telah dibentuk. Tahapan penting yang dilakukan pada proses *training* dan *testing* diantaranya adalah *preprocessing*, *feature extraction*, *modeling* dan *recognition*. *Preprocessing* merupakan proses pengolahan data suara agar data sesuai dengan spesifikasi yang diperlukan sistem. *Feature extraction* merupakan proses pengambilan karakteristik unik dari setiap sinyal suara. *Modeling* merupakan proses pembentukkan model kata dari sejumlah sinyal suara. *Recognition* merupakan proses untuk mengenali suara berdasarkan model suara yang ada.

Mel-frequency cepstral coefficients (MFCC) merupakan suatu metode yang digunakan untuk melakukan *feature extraction*. Metode ini mengadopsi cara kerja dari organ pendengaran manusia, sehingga mampu untuk menangkap karakteristik suara yang sangat penting[11]. *Hidden Markov Model* (HMM) merupakan sebuah model statistik dimana sistem yang dimodelkan diasumsikan sebagai Markov proses dengan parameter yang tidak diketahui, dan bertujuan untuk menentukan parameter-parameter tersembunyi (*hidden*) tersebut berdasarkan parameter-parameter yang dapat diketahui[9]. HMM digunakan pada proses *modeling* dan *recognition*. MFCC digunakan karena kemampuannya untuk menangkap karakter dari suara yang memiliki perbedaan cukup signifikan seperti suara pada pelafalan huruf hijaiyah, sedangkan HMM digunakan karena metode ini sangat cocok digunakan untuk bidang *speech recognition* dan sudah banyak digunakan dalam banyak penelitian[7].

1.2 Perumusan Masalah

Tugas akhir ini berangkat dari konsep dasar bahwa suara dapat dikenali dengan cara mengambil karakteristiknya dengan metode *Mel Frequency Cepstrum Coefficients* (MFCC) yang selanjutnya karakteristik tersebut dapat dimodelkan dengan metode *Hidden Markov Model* (HMM). Adapun permasalahan yang timbul dari latar belakang pembuatan Tugas Akhir ini ialah:

1. Bagaimana merancang dan mengimplementasikan suatu sistem pengenalan huruf hijaiyah berbasis *speech recognition* menggunakan *Mel Frequency Cepstral Coefficients* (MFCC) dan *Hidden Markov Model* (HMM)
2. Bagaimana mendapatkan faktor-faktor (parameter) yang dapat mempengaruhi akurasi dan performansi sistem.
3. Bagaimana mengukur tingkat akurasi berdasar jumlah kebenaran respon yang diberikan sistem pada *input* yang dilakukan oleh sejumlah user, serta performansi sistem berdasarkan waktu komputasi.

1.3 Batasan Masalah

Ruang lingkup dalam perancangan dan pembuatan *speech recognition sistem* untuk aplikasi pengenalan huruf hijaiyah ini meliputi:

1. Masukan suara yang dikenali adalah pengucapan huruf hijaiyah 3-6 huruf dengan harkat dan tanda baca tertentu yang ada didalam daftar kata pada *dataset*.
2. Proses memasukkan suara kedalam sistem ini menggunakan *microphone.*
3. Metode *clustering* yang digunakan saat *vector quantization* ialah *K-Means Clustering.*
4. Sistem bersifat *close set*, artinya hanya mengenali kosakata yang terdapat dalam *database*.
5. Sistem tidak ditujukan untuk mengenali suara secara *real time*
6. Sistem hanya dapat mengenali suara secara satu persatu (tidak *continue*).
7. Sistem tidak menangani *noise* yang ada pada sinyal suara.

1.4 Tujuan

Tujuan yang ingin dicapai dari pengerjaan Tugas Akhir ini ialah

1. Merancang dan mengimplementasikan suatu sistem pengenalan huruf hijaiyah berbasis *speech recognition sistem* menggunakan *Mel Frequency Cepstral Coefficients* (MFCC) dan *Hidden Markov Model* (HMM)
2. Menganalisis faktor-faktor (parameter) yang mempengaruhi akurasi dan performansi sistem.

3. Menguji dan Mengukur tingkat akurasi berdasar jumlah kebenaran respon yang diberikan sistem, serta menganalisis performansi sistem berdasarkan waktu komputasi.

1.5 Metodologi Penyelesaian Masalah

Dalam melakukan menyelesaikan masalah-masalah yang terdapat pada Tugas Akhir ini, penulis menggunakan beberapa metodologi yaitu:

1. Studi literatur
   Studi literatur digunakan untuk memahami konsep mengenai metode MFCC dan HMM, serta memahami konsep dasar dari *speech recognition*. Studi ini dilakukan dengan cara membaca referensi berupa buku, jurnal ilmiah, serta dari sumber relevan lain yang terkait.
2. Observasi
   Observasi yang dilakukan ialah dengan cara melakukan diskusi dengan pembimbing, rekan mahasiswa, maupun orang-orang lain yang memiliki kompetensi di bidang *speech recognition*.
3. Perekaman dan *Preprocessing* suara
   Proses ini merupakan proses merekam lafal pengucapan huruf hijaiyah dari beberapa orang relawan yang dan kemudian suara tersebut dilakukan *preprocessing* agar sesuai dengan kebutuhan dan batasan sistem.
4. Desain sistem
   Sebelum melakukan implementasi, dilakukan desain sistem mulai dari arsitektur hingga modul-modul yang akan digunakan.
5. Pembuatan perangkat lunak
   Hasil keluaran dari proses desain sistem akan digunakan untuk proses pembuatan sistem. Jika ditemukan kesalahan pada desain, maka desain sistem akan di-*update* sesuai dengan implementasi yang sesuai.
6. Pengujian sistem
   Pengujian sistem dilakukan untuk mengetahui seberapa besar akurasi yang dihasilkan oleh sistem, serta performansi sistem dilihat dari *response time*-nya.
7. Analisis
   Analisis dilakukan untuk mengetahui pengaruh setiap parameter dan faktor-faktor lain yang ada pada sistem. Sehingga *output* dari proses ini dapat digunakan untuk memperbaiki sistem agar lebih baik.
8. Pembuatan laporan tugas akhir
   Pembuatan laporan dilakukan untuk mendokumentasikan seluruh proses pengerjaan tugas akhir beserta pustaka yang digunakan. Selain itu, laporan ini juga menjadi syarat sidang Tugas Akhir.

1.7 Sistematika Penulisan Tugas Akhir

BAB I. PENDAHULUAN

Berisi latar belakang, perumusan masalah, batasan masalah, tujuan pembahasan, metodologi penyelesaian masalah dan sistematika penulisan.

BAB II. LANDASAN TEORI

Berisi penjelasan singkat mengenai konsep-konsep yang mendukung dikembangkannya sistem pada Tugas Akhir ini. Konsep-konsep yang digunakan untuk mendukung sistem ini adalah sinyal suara, *feature extraction* menggunakan *Mel Frequency Cepstrum Coefficients* (MFCC), *vector quantization* dan *Hidden Markov Model* (HMM).

BAB III. ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM

Berisi analisis kebutuhan perangkat lunak yang dibangun serta rincian perancangan sistem untuk proses *training* dan *testing* pada sistem pengenalan ucapan huruf hijaiyah.

BAB IV. IMPLEMENTASI DAN ANALISIS HASIL PENGUJIAN

Berisi rincian pengujian terhadap sistem yang telah dibangun disertai analisis terhadap hasil pengujian sistem tersebut. Pada bab ini akan dilakukan analisis terhadap factor-faktor (parameter) yang mempengaruhi performansi sistem dalam mengenali suara.

BAB V. KESIMPULAN DAN SARAN

Berisi tentang kesimpulan yang didapat dari pelaksanaan Tugas Akhir ini dan saran saran yang diperlukan untuk perbaikan maupun pengembanganya lebih lanjut.

4. Implementasi dan Analisis Hasil Pengujian

Pada bab ini dijelaskan mengenai implementasi dan analisis hasil pengujian terhadap sistem *speech recognition* untuk pengenalan suara menggunakan *Mel Frequency Cepstrum Coefficients* dan *Hidden Markov Model* yang telah dibangun. Pembahasan pada bab ini meliputi tujuan pengujian, strategi pengujian serta analisis hasil pengujian.

4.1 Pengujian Sistem

Setelah dibangun sistem berdasarkan analisis kebutuhan dan perancangan sistem pada Bab 3, selanjutnya dilakukan pengujian terhadap sistem tersebut. Pengujian dilakukan untuk mengetahui kinerja dari proses normalisasi, *feature extraction*, *training*, dan pengenalan suara pada sistem yang telah dibangun.
Tujuan pengujian yang dilakukan pada bab ini adalah:

1. Menganalisis pengaruh normalisasi sinyal suara terhadap akurasi sistem.
2. Menganalisis *cepstral* suara hasil *feature extraction* (MFCC) dan pengaruhnya terhadap sistem.
3. Menganalisis pengaruh ukuran *codebook* dan jumlah state pada proses *training* (HMM).
4. Mengetahui nilai *Sum Square Error* yang terbaik untuk pemilihan ukuran *codebook* optimal.
5. Mengetahui *response time* untuk proses *training* dan *testing*.

4.2 Strategi Pengujian

Dalam pengujian sistem ini, sinyal suara yang digunakan adalah suara pelafalan huruf hijayah yang berjumlah 3-6 huruf dengan harkat dan tanda baca tertentu dengan spesifikasi 16-bit mono 44100Hz/48000Hz bertipe *.wav* dan berdurasi 0.4-1.3 detik. Pengujian akan dilakukan pada proses *training* dan *testing*. *Sample* suara yang digunakan pada proses training berjumlah 1091 *audio file* yang berasal dari 16 orang (delapan laki-laki dan delapan perempuan), sementara *sample* yang digunakan pada proses testing berjumlah 493 *audio file* yang berasal dari sepuluh orang (lima laki-laki dan lima perempuan).

Strategi pengujian dilakukan dengan menggunakan beberapa skenario berikut.

1. **Skenario 1 – normalisasi**
   Pada skenario pertama dilakukan pengujian terhadap proses normalisasi suara. Pengujian dilakukan dengan cara mengatur nilai *sample rate* pada proses *resampling* dan mengamati pengaruhnya terhadap akurasi sistem.
2. **Skenario 2 – *codebook* dan *state***
   Pada scenario ini akan dilakukan pengujian dengan cara melakukan observasi pada ukuran *codebook* pada proses *vector quantization* dan jumlah *state* pada HMM terhadap akurasi dan *response time* sistem.

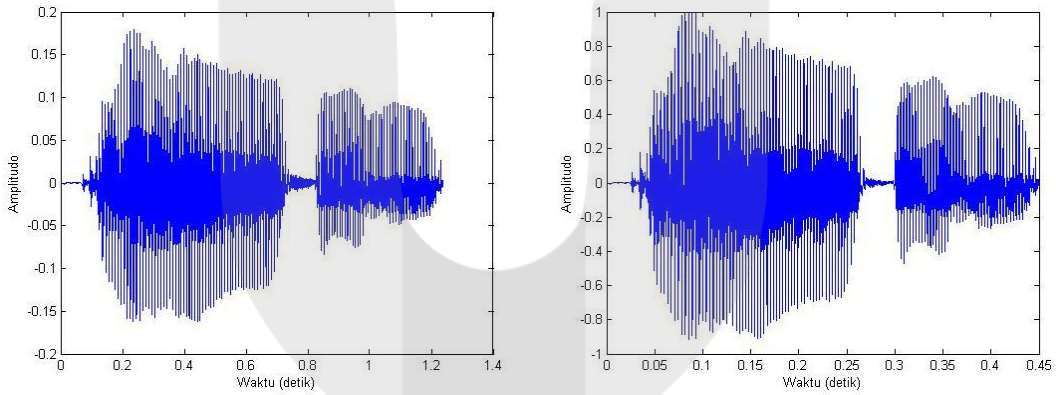
3. **Skenario 3 – jumlah kata**
   Pada skenario ini akan dilakukan pengujian dengan cara melakukan observasi jumlah kata (model) yang dapat dikenali oleh sistem. Pada skenario ini akan dilakukan sejumlah *training* dengan jumlah kata yang berbeda namun dengan *dataset* dalam ruang lingkup yang serupa.

4.3 Analisis Hasil Pengujian

Pada sub bab ini dijelaskan tentang analisis hasil pengujian terhadap sistem yang telah dibangun yang meliputi analisis hasil normalisasi, analisis hasil proses ekstraksi ciri, analisis nilai SSE, analisis ukuran *codebook* dan jumlah *state*, analisis jumlah model dan analisis *response time* sistem.

4.3.1 Analisis Hasil Normalisasi

Pada sistem yang telah dibangun ini, proses normalisasi sinyal meliputi konversi stereo ke mono, *resampling*, dan normalisasi amplitudo. Untuk melihat perubahan apa yang terjadi pada sinyal setelah melewati ini, maka dapat ditunjukkan dengan Gambar 4.1.



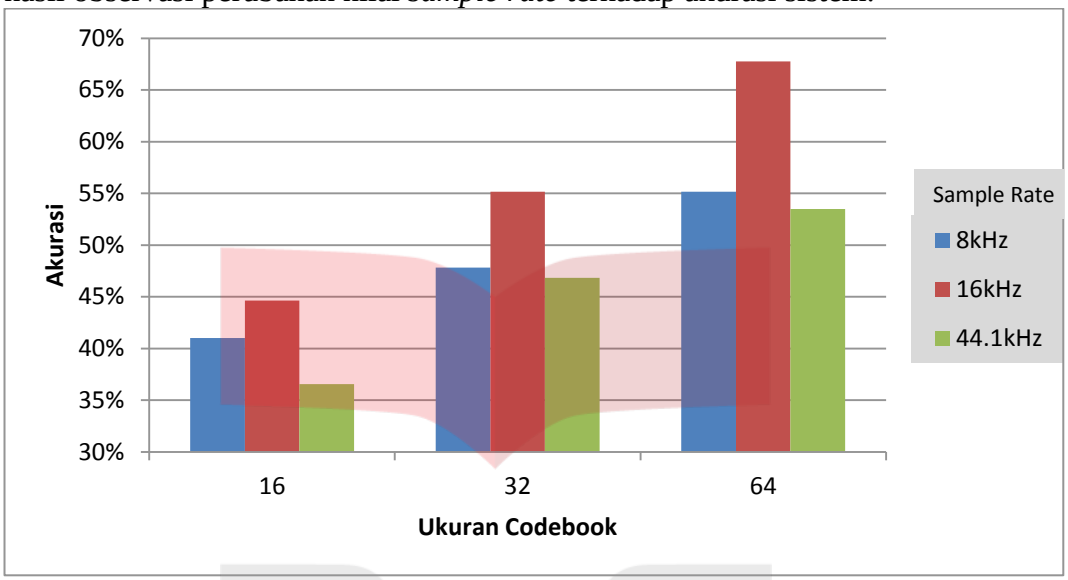
(a) **(b)**

Gambar 4.1 Perbandingan audio hasil normalisasi
(a) sinyal suara sebelum normalisasi; (b) sinyal suara setelah normalisasi

Gambar 4.1 (a) merupakan representasi sinyal suara mono 16bit 44100kHz yang berlafal وَ عَافِنِ (*wa 'aafini*). Sedangkan Gambar 4.1 (b) merupakan hasil normalisasi sinyal (a). Pada Gambar 4.1 dapat dilihat bahwa ada perbedaan yang cukup siginifikan antara Gambar (a) dan (b), yaitu terletak pada *range* nilai amplitudonya. Bila didengarkan oleh pendengaran manusia, maka suara pada Gambar (b) akan terdengar lebih keras dan lebih tegas. Proses normalisasi amplitude ini dapat memudahkan sistem untuk mengenali suara tanpa dipengaruhi oleh nilai amplitudonya. Adapun efek lain dari hasil normalisasi seperti *resampling* dan konversi mono tidak terlihat pada Gambar 4.1, namun dapat dilihat pada representasi audio secara binary.

Sample rate menunjukkan nilai sinyal audio yang diambil dalam satu detik ketika melakukan rekaman suara. *Resampling* merupakan proses untuk menormalisasi *sample rate* pada sinyal suara. Semakin tinggi nilai sample rate ini kualitas audio yang dimainkan akan semakin baik. *Resampling* dilakukan agar semua sinyal berada dalam *sample rate* yang sama agar pada proses selanjutnya

tidak dipengaruhi oleh perubahan *sample rate* setiap sinyal. Berikut ini merupakan hasil observasi perubahan nilai *sample rate* terhadap akurasi sistem.

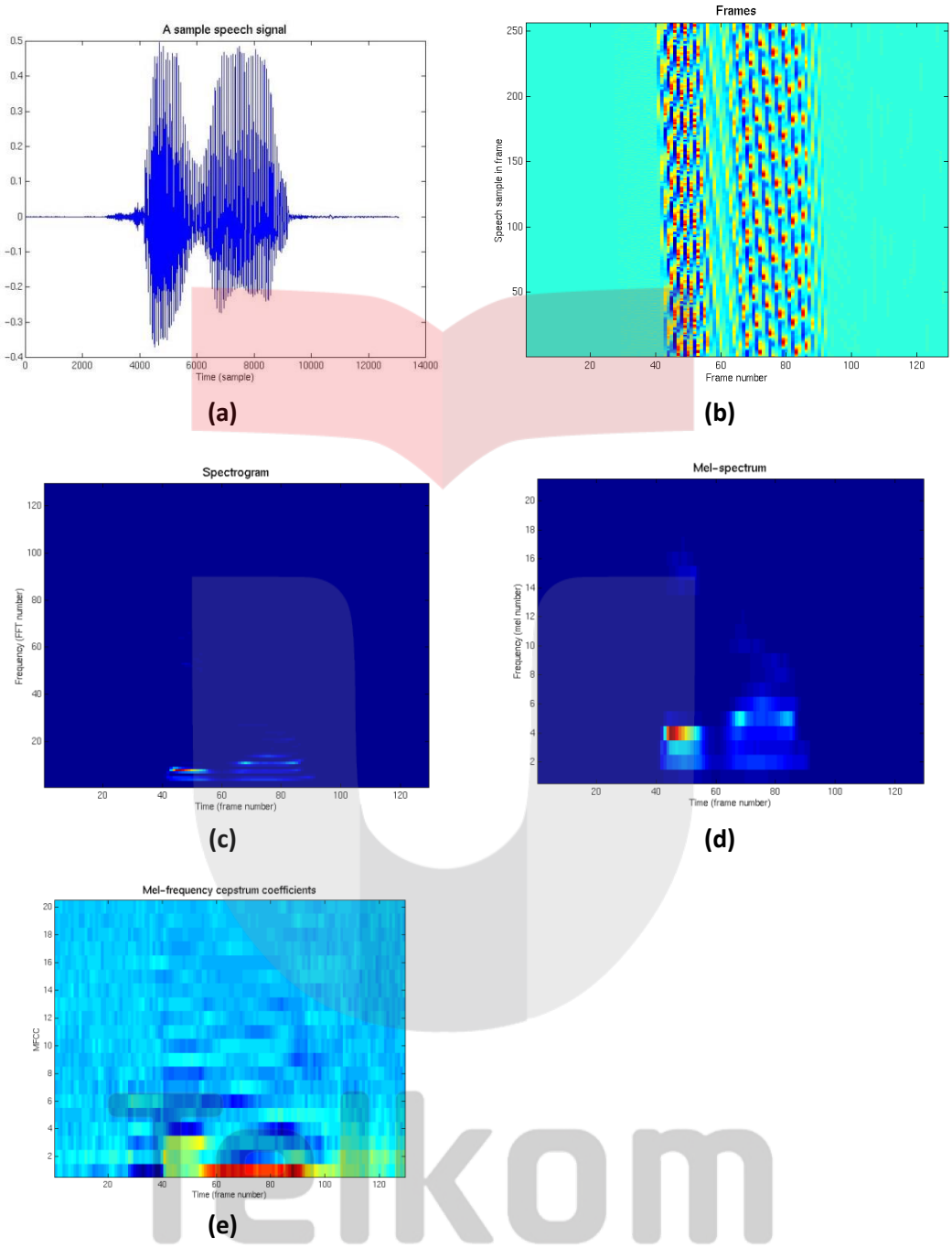


Gambar 4.2 Pengaruh nilai *sample rate* terhadap akurasi

Berdasarkan grafik pada Gambar 4.2 dapat disimpulkan bahwa akurasi terbaik dicapai sistem dengan menggunakan *sample rate* 16000Hz. Hal ini disebabkan karena sinyal dengan *sample rate* 8000Hz belum dapat menyimpan semua karakteristik suara yang diperlukan sehingga ciri yang di ekstrak belum sesuai dengan karakteristik yang diperlukan sistem. Namun ketika *sample rate* diubah menjadi lebih tinggi (44100kHz) ternyata sistem tidak memberikan hasil yang lebih baik. Ini menunjukkan bahwa *sample rate* yang tinggi belum tentu dapat meningkatkan akurasi sistem, karena dengan nilai *sample rate* yang tinggi jumlah representasi *bit*-nya pun akan semakin banyak dan ini akan membuat sistem semakin kesulitan dalam memodelkan sebuah ucapan, sehingga sistem cenderung menghasilkan akurasi yang lebih buruk serta waktu komputasi yang diperlukan pun akan semakin lama. Selain itu, penelitian [7] menyebutkan bahwa *sample rate* terbaik untuk proses pengenalan suara manusia berada pada kisaran 8kHz-20kHz. *Sample rate* 16000Hz adalah nilai yang terbaik untuk kasus ini.

4.3.2 Analisis Hasil *Feature Extraction*

Feature Extraction merupakan proses melakukan ekstraksi ciri dari sebuah sinyal suara. Tujuan dari proses ini ialah untuk mengambil karakteristik unik pada setiap sinyal suara, sehingga pemrosesan pada setiap sinyal suara akan lebih mudah. Pada metode *feature extraction* dengan *Mel Frequency Cepstral Coefficients* (MFCC), sinyal suara akan diubah menjadi cepstral MFCC. Representasi perubahan bentuk sinyal suara menjadi *cepstral* MFCC dapat dilihat pada Gambar 4.3 [3].



Gambar 4.3 [3] Proses MFCC pada sebuah sinyal suara
(a) Sinyal suara awal; (b) Frames, setelah frame blocking;
(c) Spektrum, setelah FFT; (d) Mel-spectrum, setelah mel-frequency wrapping;
(e) Cepstral MFCC

Untuk dapat menganalisis performansi dari *feature extraction* MFCC ini, dapat dilihat dari hasil pengenalan suara. *Feature extraction* yang baik ialah yang *feature extraction* yang dapat mengambil ciri yang sesuai dengan kebutuhan sistem, sehingga proses pengenalan suara dapat memperoleh akurasi maksimal. Pada kasus Tugas Akhir ini diperlukan *feature extraction* yang dapat

membedakan suara-suara dengan kata yang mirip. Misalnya kata غِيْبٌ (*ghiibun*) dengan kata غَيْبٌ (*ghoibun*). Pada kasus pelafalan huruf hijaiyah ini sangat banyak kata dengan lafal yang sangat mirip. Beberapa contoh kata-kata yang mirip secara pendengaran manusia dan akurasi deteksinya dapat dilihat pada tabel 4.1.

Tabel 4.1 Akurasi dari kata-kata yang mirip

No	Kata	Dibaca	Akurasi
1	تُوْنَ	Tuuna	70%
2	تَوْنَ	Tauna	30%
3	سَوْفَ	Saufa	80%
4	سُوْفَ	Suufa	60%
5	مَوْتَ	Mauta	70%
6	مَاتَ	Maata	90%

Berdasarkan Tabel 4.1 dapat dilihat bahwa semua kata-kata yang mirip dapat dikenali dengan baik oleh sistem, kecuali kata تَوْنَ yang akurasi deteksinya hanya 30%. Dari hasil pengujian ini dapat disimpulkan bahwa fitur ekstraksi yang digunakan sudah dapat menjauhkan karakter serupa dari suara yang hampir mirip secara pendengaran manusia. Dari hasil pengujian ini juga dapat dilihat bahwa meskipun kata-kata pada Tabel 4.1 mirip secara pendengaran manusia, tetapi secara cepstral MFCC kata-kata ini jauh berbeda sehingga tidak terjadi kesalahan pada saat pemodelan dan pelabelan suara.

Selanjutnya, analisis fitur hasil MFCC dapat dilihat dari hasil pengenalan kata yang memiliki akurasi paling buruk. Berikut ini merupakan lima kata yang menghasilkan akurasi deteksi paling rendah (dari hasil pengujian dengan jumlah state 5 dan codebook 64).

Tabel 4.2 Kata dengan akurasi terendah

No	Kata	Dibaca	Akurasi Deteksi
1	رَتَّلَ	Rottala	0%
2	سَخَّرَ	Sakhkhoro	30%
3	جُوْعًا	Juu ‘an	33%
4	تُوْنَ	Tuuna	40%
5	يُكَبَّرُ	Yukabbaru	40%

Berdasarkan Tabel 4.2 dapat dilihat bahwa kata yang memiliki akurasi terendah ialah kata رَتَّلَ dengan akurasi 0%. Hal ini menunjukkan bahwa kata رَتَّلَ tidak dapat diambil cirinya dengan baik, dan mungkin bahwa ciri yang diambil pada kata ini mirip dengan ciri yang diambil dari kata lainnya. Untuk

dapat menganalisis cepstral kata رَتَّلَ mirip dengan cepstral kata yang mana, dapat dilihat dari hasil pengujian kata tersebut. Berikut ini merupakan hasil pengujian pada kata رَتَّلَ.

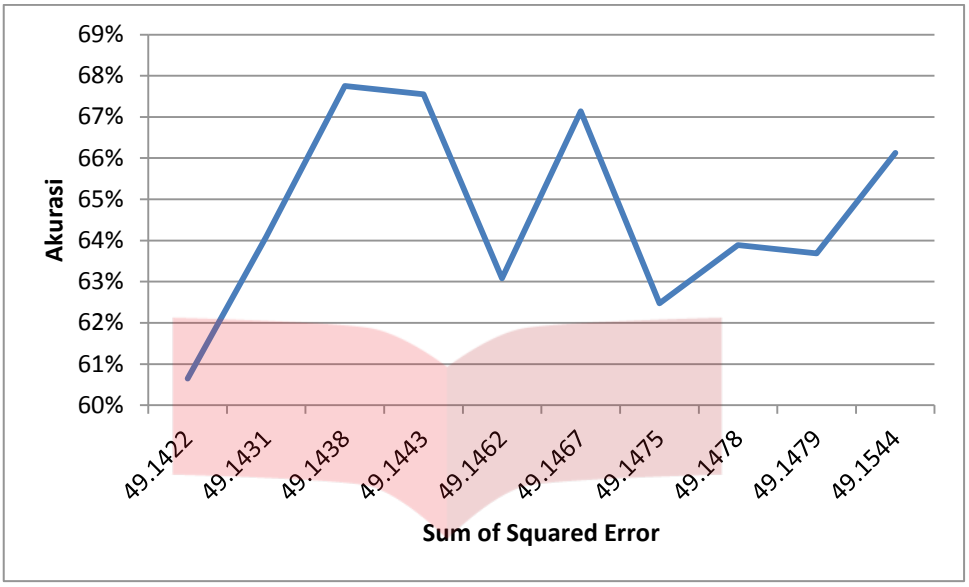
Tabel 4.3 Hasil deteksi pada kata '*rottala*'

No	Kata	Hasil Deteksi
1	رَتَّلَ (rottala)	يُكَبَّرُ (yukabbaru)
2	رَتَّلَ (rottala)	كَبَّرَ (kabbaro)
3	رَتَّلَ (rottala)	كَذَلِكَ (kadzalika)
4	رَتَّلَ (rottala)	حَفَّفَ (haffafa)
5	رَتَّلَ (rottala)	حَفَّفَ (haffafa)
6	رَتَّلَ (rottala)	كَبَّرَ (kabbaro)
7	رَتَّلَ (rottala)	حَفَفَّ (haffafa)
8	رَتَّلَ (rottala)	حَفَفَّ (haffafa)
9	رَتَّلَ (rottala)	يَتَلَاوَمُوْنَ (yatalaawamuuna)
10	رَتَّلَ (rottala)	يُكَبَّرُ (yukabbaru)

Berdasarkan Tabel 4.3 dapat dilihat bahwa kata رَتَّلَ cenderung dikenali sebagai kata حَفَفَّ, sehingga dimungkinkan bahwa ada kemiripan ciri pada kedua kata tersebut sehingga proses pengenalan pada kata رَتَّلَ menghasilkan hasil deteksi yang salah.

4.3.3 Analisis Nilai SSE

Pemilihan *codebook* yang paling optimal dapat dilihat dari kerapatan *feature vector* suara terhadap nilai centroidnya. Kerapatan ini dapat diobervasi dengan menggunakan perhitungan *sum of squared error* (SSE) dari sebuah *codebook* yang sudah terbentuk. Berikut ini merupakan hasil pengujian keterkaitan nilai SSE terhadap akurasi sistem (pengujian dilakukan dengan ukuran *codebook* 64 dan 5 *state* HMM).

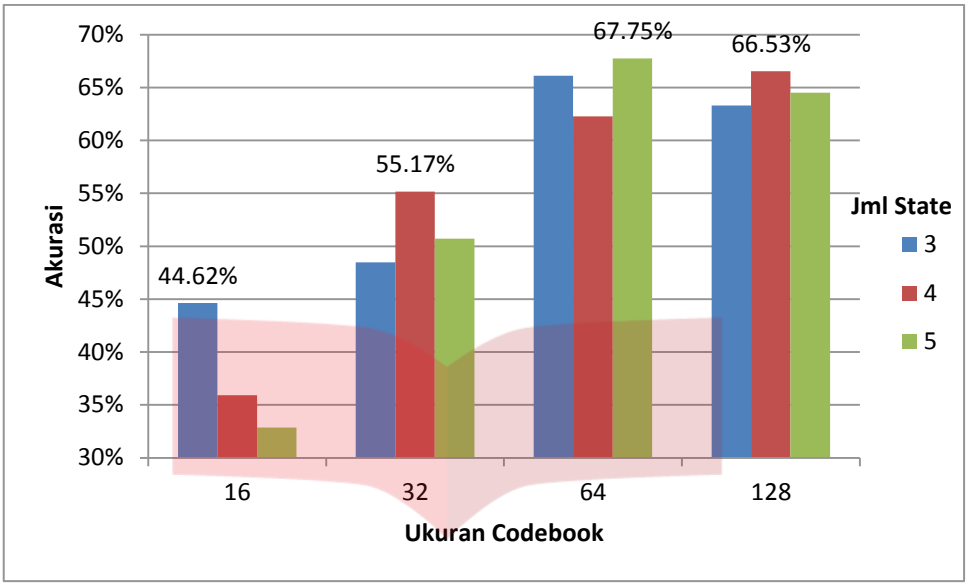


Gambar 4.4 Pengaruh SSE terhadap akurasi

Berdasarkan grafik pada Gambar 4.4 dapat dilihat bahwa ukuran SSE yang kecil belum tentu dapat menghasilkan akurasi yang baik. Tentu saja hal ini kontras dengan konsep keterkaitan SSE dengan *codebook* yang menyebutkan bahwa semakin kecil nilai SSE maka *codebook* yang dihasilkan akan semakin baik. Jika dilihat dari konsep tersebut, seharusnya akurasi yang dihasilkan juga semakin baik. Tapi perlu diingat pula bahwa akurasi tidak hanya dipengaruhi oleh *codebook*, tetapi juga dipengaruhi oleh proses *training* HMM. Jika codebook yang dihasilkan pada proses *vector quantization* sudah baik tetapi pada proses *training* pemodelan yang dilakukan tidak optimal, maka akurasi yang dihasilkan pun tidak akan maksimal. Jadi, nilai SSE mungkin dapat digunakan untuk pemilihan *codebook* terbaik, namun nilai ini belum dapat digunakan untuk memperkirakan akurasi sistem.

4.3.4 Analisis Ukuran *Codebook* dan Jumlah *State*

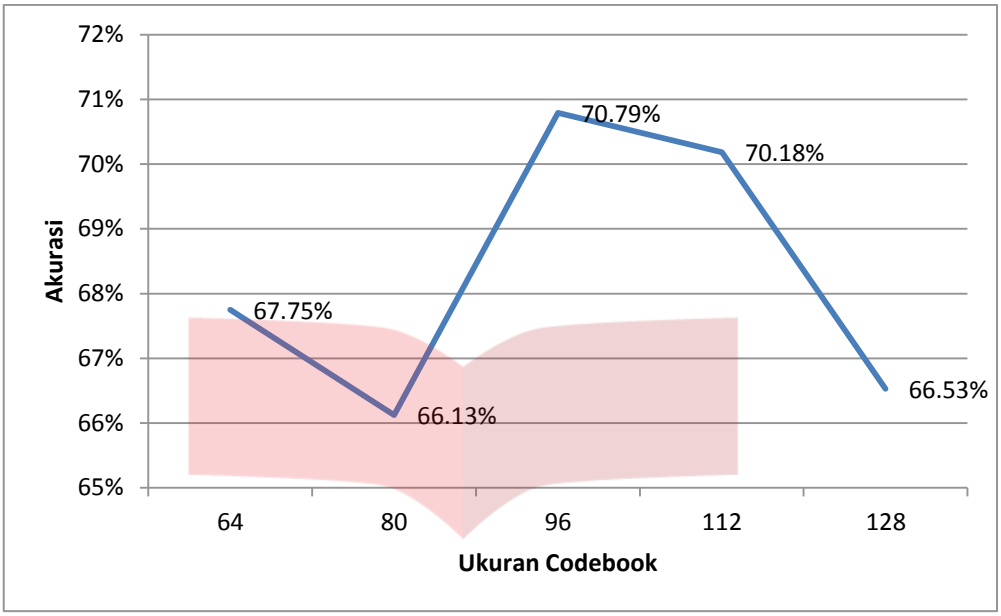
Codebook merupakan representasi dari *feature vector* semua sinyal suara yang telah di *clustering*. Indeks dari codebook ini dapat dikatakan sebagai simbol observasi Hidden Markov Model dan akan digunakan pada proses *training*. Ukuran *codebook* dapat disesuaikan dengan kebutuhan sistem, namun pada Tugas Akhir ini, ukuran *codebook* yang akan diobservasi ialah 16, 32, 64, dan 128. Selain ukuran *codebook*, jumlah *state* HMM pun dapat mempengaruhi akurasi sistem. Pada Tugas Akhir ini jumlah *state* yang akan di observasi ialah 3, 4 dan 5. Berikut ini merupakan hasil pengujian akurasi sistem terhadap jumlah *state* dan ukuran *codebook*.



Gambar 4.5 Pengaruh jumlah *state* dan ukuran *codebook* terhadap akurasi

Berdasarkan grafik pada Gambar 4.5 dapat dilihat bahwa akurasi tertinggi (67.75%) dicapai pada codebook 64 dengan jumlah state 5, sementara akurasi terendah (44.62%) didapat dari codebook 16 dengan jumlah state 5. Dapat diamati pula, bahwa dengan meningkatnya ukuran *codebook* maka akurasi sistem cenderung meningkat. Namun, perlu diperhatikan juga bahwa ukuran *codebook* yang terlalu besar akan membuat *codebook* menjadi tidak baik, dalam artian bahwa sangat memungkinkan jika beberapa titik yang seharusnya berada dalam satu *cluster*, namun karena batasan yang terlalu banyak, titik-titik tersebut menjadi terpisah. Hal ini tentu akan sangat mempengaruhi akurasi sistem, karena *codebook* ini akan akan digunakan pada proses *vector quantization* dalam menentukan deretan observasi HMM.

Berdasarkan Gambar 4.4 juga dapat dilihat bahwa akurasi menurun ketika ukuran codebook 128. Dari hasil pengamatan ini dapat disimpulkan bahwa ukuran *codebook* paling optimal berada rentang 64 sampai 128. Selanjutnya dilakukan pengujian terhadap ukuran *codebook* 64-128 untuk mencari ukuran *codebook* paling optimal pada sistem. Hasil pengujian tersebut dapat dilihat pada Gambar 4.6.



Gambar 4.6 Pengaruh ukuran *codebook* terhadap akurasi

Berdasarkan grafik pada Gambar 4.6 didapatkan bahwa ukuran *codebook* paling optimal ialah 96 dengan akurasi paling tinggi (70,79%). Tetapi perlu diperhatikan juga bahwa ukuran *codebook* standar ialah 2^n, sehingga akurasi terbaik yang akan diambil pada Tugas Akhir ini adalah 67.75% yang didapatkan dari ukuran *codebook* 64 dan 5 *state* HMM.

Lain halnya dengan ukuran *codebook*, jumlah *state* pada HMM tidak dapat dijadikan sebagai patokan untuk memperkirakan akurasi sistem karena semakin banyak jumlah *state* belum tentu akan meningkatkan akurasi sistem bahkan sebaliknya, jumlah *state* yang semakin banyak dapat menurunkan akurasi sistem. Jumlah *state* terbaik didapatkan dari hasil observasi[9]. Jumlah *state* terbaik pada kasus ini ialah 5 dengan ukuran *codebook* 64.

Hal lain yang menarik pada sistem ini ialah perubahan akurasi sistem setiap kali *training*. Pada beberapa pengujian yang telah dilakukan menunjukkan bahwa akurasi yang dihasilkan oleh sistem cenderung berubah-ubah (tidak stabil) pada setiap kali *training*. Hal ini disebabkan karena proses inisialisasi yang menggunakan teknik random pada proses *vector quantization* (pembentukan *codebook*) dan *training* HMM.

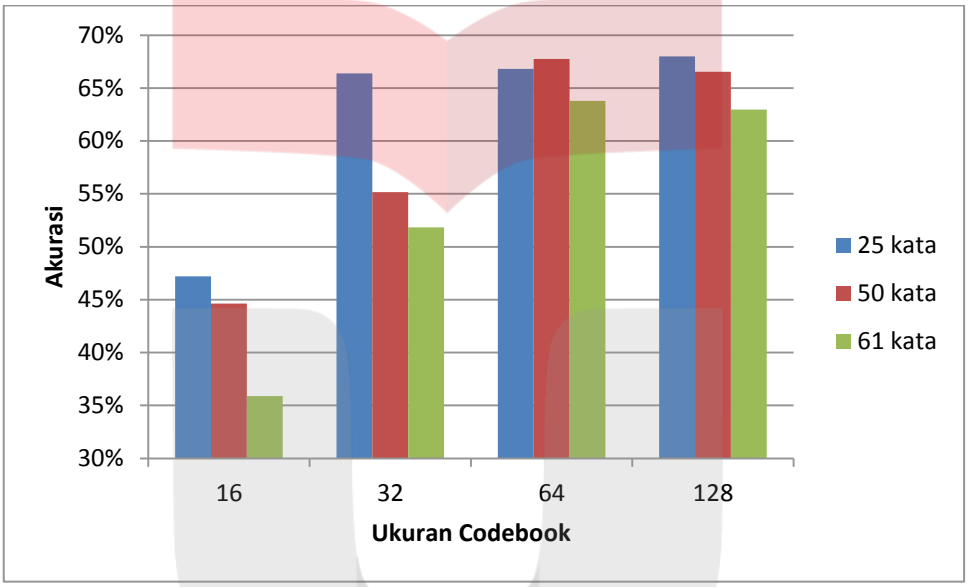
Tabel 4.4 Perbandingan akurasi pada *training* pertama dan kedua

Parameter		Akurasi	
Codebook	State	Training ke-1	Training ke-2
64	3	66.13%	66.13%
	4	62.27%	62.88%
	5	67.75%	64.91%
128	3	63.29%	66.13%
	4	66.53%	65.72%
	5	64.50%	65.31%

Berdasarkan Tabel 4.4 dapat dilihat perubahan akurasi sistem saat training pertama dan kedua. Perubahan akurasi yang dihasilkan setiap kali *training* berkisar antara 0%-5%.

4.3.5 Analisis Jumlah Model

Dalam HMM, setiap kata akan dimodelkan menjadi sebuah model HMM. Semakin banyak jumlah kata yang di-*train*, maka akan semakin banyak model yang dibentuk. Pengaruh jumlah model terhadap akurasi sistem dapat dilihat pada Gambar 4.7

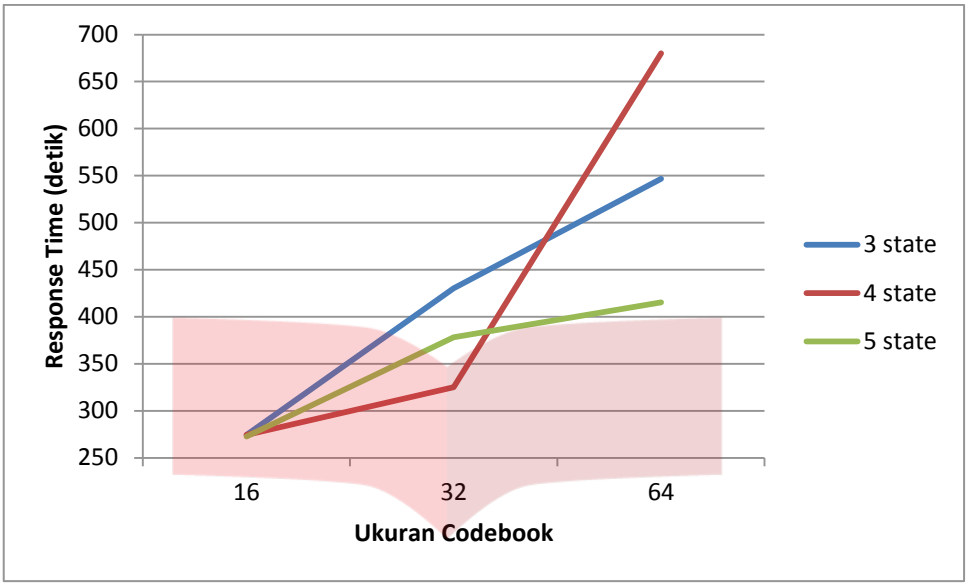


Gambar 4.7 Pengaruh jumlah model terhadap akurasi

Berdasarkan grafik pada Gambar 4.7 dapat dilihat bahwa semakin banyak model yang di-*train*, maka akurasinya cenderung semakin kecil. Hal ini dimungkinkan karena adanya model-model yang mirip secara cepstral sehingga sistem salah dalam melakukan klasifikasi. Solusi agar HMM tetap memiliki akurasi yang baik dengan jumlah model yang besar ialah dengan cara menambah *audio file* di setiap modelnya pada saat *training*. Semakin banyak *audio file* pada setiap model, maka deretan observasi pada saat *training* akan semakin banyak sehingga proses re-estimasi parameter HMM akan semakin baik dan akan menghasilkan model yang paling optimal. Dengan model HMM yang optimal, akurasi sistem akan semakin baik.

4.3.6 Analisis *Response Time*

Response time merupakan waktu yang diperlukan sistem untuk mengolah sebuah data dari mulai *user* menekan tombol untuk men-*trigger* sebuah *action* hingga sistem mengeluarkan *output* yang diminta user. *Response time* yang akan dianalisis pada sistem yang telah dibangun ini adalah *response time* saat *training* dan *response time* saat *testing*. Berikut ini merupakan *response time* yang diberikan oleh sistem berdasarkan hasil pengujian pada saat training dilihat dari parameter ukuran *codebook* dan jumlah *state*.



Gambar 4.8 Pengaruh *codebook* terhadap *response time*

Berdasarkan Gambar 4.8 dapat disimpulkan bahwa faktor yang paling mempengaruhi waktu *training* ialah ukuran *codebook* pada saat proses *vector quantization.* Semakin besar ukuran *codebook* yang digunakan, maka waktu pada saat pembuatan *codebook* akan semakin lama. Hal ini disebabkan karena ukuran *codebook* yang besar akan menyebabkan proses iterasi (setiap iterasi terdapat perhitungan tertentu yang memerlukan waktu) pada algoritma *k-means* menjadi semakin banyak. Jumlah *cluster* yang banyak mengakibatkan kondisi konvergen lebih lama tercapai. Selain itu jumlah *audio file* untuk proses *training* akan sangat mempengaruhi waktu *training*, karena semakin banyak *audio file* yang dimasukkan, maka pemrosesan setiap *file* akan semakin banyak dan waktu yang diperlukanpun akan semakin lama. Setiap audio yang dimasukkan kedalam sistem akan memiliki proses *preprocessing*, *feature extraction*, *vector quantization*, dan pembentukan model HMM masing-masing yang memerlukan waktu. Selain itu, *feature vector* yang digunakan saat pembentukan *codebook* pun akan semakin besar.

Sementara *response time* untuk proses *testing* berkisar antara 0.04 sampai 0.2 detik (tergantung spesifikasi komputer) untuk setiap pengenalan ucapan. Lain halnya pada proses *training*, *response time* pada proses testing tidak dipengaruhi oleh jumlah *state* dan ukuran *codebook*, karena pada saat testing tidak dilakukan proses pembentukan *codebook*.

Daftar Pustaka

[1] Ardisasmita, M.S., 2003, *Sistem Kendali Peralatan Dengan Perintah Suara Menggunakan Model Hidden Markov dan Jaringan Syaraf Tiruan*, Risalah Lokakarya Komputasi dalam Sains da Tekologi Nuklir XIV, Juli 2003.

[2] Becchetti, C., & Lucio, P. R., 1999, *Speech Recognition Theory and C++ Implementation*, Fondazione Ugo Bordoni: Wiley.

[3] Do MN. (1994). DSP Mini-Project: An Automatic Speaker Recognition Sistem.

[4] Gales, Mark., Young, Steve. 2008. *The Application of Hidden Markov Models in Speech Recognition*. Vol. 1, No. 3 (2007) 195–304.

[5] Kinnunen, Tomi., 2003, *Spectral Fefatures for Authomatic Text-Independent Speaker Recognition*

[6] Muda, L., Mumtaj, B., & Elamvazuthi., I., 2010, *Voice Recognition Algorithm using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques*, Journal of Computing, Volume 2, Issue 3, March 2010.

[7] Rabiner, L., & Biing-Hwang, J., 1993, *Fundamentals of Speech Recognition*, Englewood Cliff: Prentice-Hall International, Inc.

[8] Rabiner, L.R., & B-H, Juang., 2006, *Speech Recognition: Statistical Methods*, Elsevier Ltd.

[9] Rabiner, Lawrence, B. H. Juang. 1991. *Hidden Markov Model for Speech Recognition*. Technometrics, VOL. 33, NO. 3, August 1991

[10] Rabiner, Lawrence. 1989. *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition*. IEEE, Vol. 77, No. 2, pp. 257-285, Februari 1989.

[11] Tychtl, Zbyni k dan Psutka, Josef. *Speech Production Based on the Mel-Frequency Cepstral Coefficients*.

[12] Wildermoth, B.R., 2001. *Text-Independent speaker recognition using source based features*.

[13] Yulita, I.N., Liong T.H., Adiwijaya, 2012, *Fuzzy hidden markov models for indonesian speech classification*, Journal of Advanced Computational Intelligence and Intelligent Informatics 16:3 pp. 381-387.