TELEKONTRAN, VOL. 8, NO. 1, APRIL 2020
DOI : 10.34010/telekontran.v8i1.3078
p-ISSN : 2303 – 2901
e-ISSN : 2654 – 7384

Perancangan Sistem *Home Automation* Dengan Kendali Perintah Suara Menggunakan *Deep Learning Convolutional Neural Network (DL-CNN)*

Design Of A Voice Controlled Home Automation System Using Deep Learning Convolutional Neural Network (DL-CNN)

Lery Sakti Ramba, Muhammad Aria Rajasa Pohan
Program Studi Teknik Elektro, Fakultas Teknik dan Ilmu Komputer
Universitas Komputer Indonesia Jl. Dipati ukur No 112, Bandung
Email : lerysaktiramba@mahasiswa.unikom.ac.id

Abstrak - Penelitian ini bertujuan untuk merancang sebuah sistem *home automation* yang dapat dikendalikan dengan menggunakan perintah suara. Penelitian ini dilakukan dengan mempelajari penelitian lain yang bekaitan dengan topik pada penelitian ini, berdiskusi dengan pihak-pihak yang kompeten, merancang sistem, menguji sistem, dan melakukan analisis berdasarkan pengujian yang telah dilakukan. Pada penelitian ini dirancang sebuah sistem pengenalan suara menggunakan *Deep Learning Convolutional Neural Network (DL-CNN).* Model CNN yang telah dirancang kemudian akan dilatih untuk mengenali beberapa macam perintah suara. Hasil dari penelitian ini adalah sebuah sistem pengenalan suara yang dapat digunakan untuk mengendalikan beberapa perangkat elektronik yang terhubung ke sistem. Sistem pengenalan suara pada penelitian ini memiliki persentase keberhasilan sebesar 100% pada kondisi ruangan dengan intensitas *background noise* 24dB (senyap), sebesar 67,67% pada kondisi ruangan dengan intensitas *background noise* 42dB, dan hanya 51,67 % pada kondisi ruangan dengan intensitas *background noise* 52dB (bising). Persentase keberhasilan sistem pengenalan suara pada penelitian ini sangat dipengaruhi oleh besarnya intensitas *background noise* pada suatu ruangan. Oleh karena itu, untuk memperoleh hasil yang optimal, sistem pengenalan suara pada penelitian ini lebih cocok digunakan pada ruangan dengan intensitas *background noise* yang rendah.

Kata kunci *:* *Home Automation*, Perintah Suara, *Deep Learning, Convolutional Neural Network.*

Abstract *- The purpose of this research is to design home automation system that can be controlled using voice commands. This research was conducted by studying other research related to the topics in this research, discussing with competent parties, designing systems, testing systems, and conducting analyzes based on tests that have been done. In this research voice recognition system was designed using Deep Learning Convolutional Neural Networks (DL-CNN). The CNN model that has been designed will then be trained to recognize several kinds of voice commands. The result of this research is a speech recognition system that can be used to control several electronic devices connected to the system. The speech recognition system in this research has a 100% success rate in room conditions with background intensity of 24dB (silent), 67.67% in room conditions with 42dB background noise intensity, and only 51.67% in room conditions with background intensity noise 52dB (noisy). The percentage of the success of the speech recognition system in this research is strongly influenced by the intensity of background noise in a room. Therefore, to obtain optimal results, the speech recognition system in this research is more suitable for use in rooms with low intensity background noise.*

Keyword *: Home Automation, Voice Command, Deep Learning, Convolutional Neural Network*

I. PENDAHULUAN

A. Latar Belakang

Saat ini, bidang ilmu yang mempelajari tentang *Artificial Intelligence (AI)* sedang mengalami perkembangan yang begitu pesat. *Artificial intelligence* merupakan bidang ilmu yang menekankan penciptaan sistem/mesin cerdas yang bekerja dan bereaksi sama seperti manusia [1]. Seiring perkembangannya, *artificial intelligence* telah dikembangkan kedalam beberapa *sub*-bidang, salah satunya adalah *machine learning*.

Machine learning merupakan salah satu cabang pengembangan ilmu *artificial intelligence* yang memungkinkan sistem komputer untuk belajar dari pengalaman sebelumnya dan meningkatkan perilaku/respon sistem untuk setiap tugas yang diberikan [2]. *Machine learning* terbagi lagi ke dalam beberapa *sub*-bagian, salah satu adalah *deep learning*, yang merupakan pengembangan lebih lanjut dari *machine learning* [3]. Saat ini, *deep learning* dianggap sebagai salah satu algoritma yang sangat baik dan akurat dalam memecahkan suatu permasalahan.

Salah satu bukti nyata keakuratan *deep learning* dalam memproses data dapat kita lihat pada sistem *Automatic Speech Recognition (ASR)*, seperti *google assistant*, cortana, siri, dan alexa. Model *deep learning* yang digunakan saat ini pada *automatic speech recognition* sehingga dapat menghasilkan output yang akurat adalah menggunakan algoritma *Long Short-term Memory* dan *Recurrent Neural Networks (LSTM RNNs)* [4]. Namun salah satu masalah yang dihadapi adalah untuk membuat sebuah sistem *automatic speech recognition* yang akurat menggunakan LSTM RNNs, maka kita memerlukan sebuah komputer dengan spesifikasi sangat tinggi. Hal ini dikarenakan algoritma LSTM RNNs pada *automatic speech recognition* menggunakan *neural network* yang sangat kompleks. Sehingga mengakibatkan sistem dengan algoritma ini tidak memungkinkan untuk dijalankan pada *hardware* atau komputer komersial dengan spesifikasi umum seperti yang tersedia di pasaran.

Solusi dari permasalah tersebut dapat diatasi dengan menggunakan *Deep Learning Convolutional Neural Networks (DL-CNN).* CNN merupakan pengembangan dari *multi layer perceptron* pada *machine learning* yang didesain untuk mengolah data dua dimensi (gambar) menggunakan beberapa lapisan konvolusi [5]. Secara fungsi, CNN juga memiliki performa yang sangat mumpuni untuk komputasi pengenalan suara dengan cara memodelkan korelasi spektral dari sebuah sinyal akustik [6]. Terlebih karena CNN dapat berjalan di komputer dengan spesifikasi rendah, tidak seperti LSTM RNNs yang membutuhkan komputer dengan spesifikasi sangat tinggi.

Perkembangan teknologi digital terutama dalam bidang telekomunikasi sudah seperti tidak mengenal ruang dan waktu dengan adanya berbagai layanan komunikasi yang menunjang kehidupan manusia sehari-hari [7]. Hal ini telah membuka ruang yang lebar untuk inovasi-inovasi baru di berbagai bidang, salah satunya adalah *smart home* atau *home automation*.

Terdapat beberapa penelitian di bidang *home automation* yang memanfaatkan teknologi *automatic speech recognition* untuk mengendalikan perangat elektronik yang diinginkan. Namun, aplikasi *speech recognition* tersebut mengharuskan kita untuk selalu terkoneksi internet. Hal ini dikarenakan proses mengkonversi sinyal audio ke dalam bentuk teks *(speech to text)* tidak dilakukan pada perangkat yang kita miliki, melainkan dilakukan pada komputer server penyedia layanan *speech recognition* tersebut. Sehingga dengan demikian, metode ini tidak memungkinkan kita untuk memasang sistem pengenalan suara yang data bekerja secara *stand-alone* pada komputer atau perangkat yang kita miliki.

B. Tinjauan State of Art

Algoritma LSTM RNNs seperti yang terdapat pada *automatic speech recognition* dirancang untuk digunakan memproses *sequential data*. RNN LSTMs menggunakan memori *internal* untuk memproses setiap *input sequential data* [8]. Menggunakan algoritma LSTM RNNs untuk menjalankan sistem pengenalan suara, membutuhkan sebuah komputer dengan spesifikasi khusus, dikarenakan kompleksitas jaringan dari algoritma ini sangat tinggi.

Terdapat penelitian sebelumnya yang juga memanfaatkan teknologi *speech recognition* untuk diaplikasikan pada *home automation* [9]. Namun sistem tersebut masih menggunakan aplikasi *speech recognition* bawaan dari sistem operasi perangkat yang digunakan seperti siri, *google assistant*, atau cortana. Dari segi akurasi penggunaan aplikasi tersebut tentu sangat baik, dikarenakan ASR menggunakan algoritma LSTM RNNs yang dikenal dengan kompleksitas jaringannya yang sangat tinggi sehingga mampu menghadirkan *output* dengan akurasi yang tinggi. Namun aplikasi *speech recognition* tersebut mengharuskan kita untuk selalu terhubung ke komputer/server aplikasi *speech recognition* dijalankan melalui *internet*. Seperti yang telah dijelaskan sebelumnya, hal ini dikarenakan proses mengkonversi sinyal audio kedalam bentuk *text* tidak dilakukan pada perangkat yang kita miliki, melainkan dilakukan pada sebuah komputer *server* dengan spesifikasi yang tinggi tempat sistem *speech recognition* tersebut dijalankan

Sedangkan pada penelitian ini, sistem pengenalan suara dibuat menggunakan algoritma CNN. Arsitektur CNN yang digunakan terdiri dari 24 *layer* yang terintegrasi satu sama lain untuk melakukan proses *feature learning* dan klasifikasi terhadap data *input*. Algoritma ini menggunakan jaringan *feed forward* dengan variasi *multi layer perceptron* yang dirancang untuk melakukan *preprocessing* dalam jumlah yang minimal. Algoritma ini tidak diracang untuk memproses *sequential data* seperti pada LSTM RNNs, melainkan dirancang untuk memproses *spatial data* (data dua dimensi) [10]. Hal inilah yang memungkinkan sistem pengenalan suara pada penelitian ini dapat dijalankan pada sebuah komputer komersial tanpa harus membutuhkan spesifikasi khusus.

C. Tujuan

Tujuan dari penelitian ini adalah untuk merancang sebuah sistem pengenalan suara yang akurat dan mampu berjalan secara *stand-alone* pada sebuah komputer komersial. Sistem pengenalan suara dibuat menggunakan *Deep Learning Convolutional Neural Network (DL-CNN)* yang terdiri dari 24 *layer*. *Layer-layer* tersebut akan melakukan proses *feature learning* dan *classification* terhadap *input data*.

Pada proses *feature learning*, terdapat lima *layer* konvolusi yang digunakan. Semakin banyak *layer* konvolusi yang digunakan maka akurasi sistem juga akan semakin baik namun penggunaan layer konvolusi yang banyak akan membuat proses komputasi menjadi semakin kompleks dan dalam. Arsitektur DL-CNN pada sistem ini di-*set* hanya menggunakan lima layer konvolusi dengan tujuan supaya sistem pengenalan suara mampu berjalan secara *stand-alone* pada sebuah komputer tanpa harus membutuhkan spesifikasi khusus, namun disatu sisi, tetap memiliki akurasi yang tinggi dalam mengklasifikasikan data masukan.

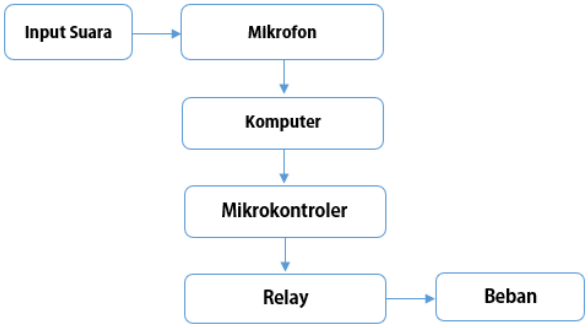
Sistem pengenalan suara pada penelitian ini dirancang untung pengaplikasian pada sistem *home automation*. Sistem akan dilatih untuk mampu mengenali beberapa jenis perintah yang dapat digunakan utnuk mengendalikan beberapa perangkat elektronik menggunakan perintah suara.

.

II. METODOLOGI

A. Diagram Blok Sistem

Hardware utama pada sistem ini terdiri mikrofon, komputer, mikrokontroler, dan relay. Diagram blok dari sistem yang dirancang ditunjukkan pada **Gambar 1**.



Gambar 1 Diagram Blok Sistem

Berikut adalah penjelasan dari setiap bagian pada diagram blok tersebut.

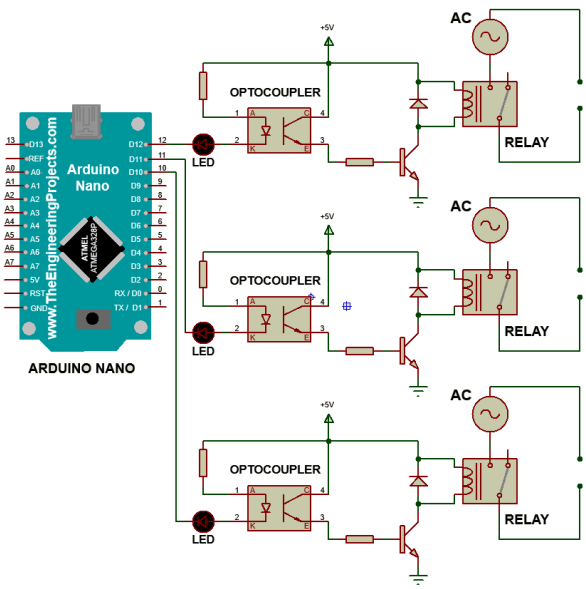
1. *Input* Suara
   Pada dasarnya, suara yang dihasilkan oleh manusia adalah gelombang akustik/bunyi yang saling berpadu sedemikian rupa sehingga membentuk suara tertentu dapat dikenali oleh manusia. Gelombang akustik inilah yang akan dijadikan sebagai *input* untuk mikrofon.
2. Mikrofon
   Pada sistem ini, mikrofon berfungsi sebagai komponen untuk mendeteksi *input* gelombang akustik/bunyi. Mikrofon akan mengkonversi gelombang *input* akustik kedalam energi listrik atau yang sering disebut sinyal audio. Pada penelitian ini jenis mikrofon yang digunakan adalah mikrofon kondensor.
3. Komputer
   Pada sistem ini, komputer merupakan *hardware* utamanya. Komputer bertugas untuk memproses setiap *input* yang diterima mikrofon dan kemudian mengolah sinyal audio yang diterima dari mikrofon. Data sinyal audio yang diterima dari mikrofon akan diproses pada komputer menggunakan algoritma yang akan dirancang untuk mampu mengenali perintah suara. Komputer pada sistem ini juga bertugas untuk berkomunikasi dengan mikrokontroler.
4. Mikrokontroler
   Mikrokontroler adalah bagian yang akan mengendalikan perangkat elektronik berdasarkan *output* dari sistem pengenalan suara. Mikrokontroler akan berkomunikasi

secara *real-time* dengan komputer melalui komunikasi *serial*. *Output* dari sistem pengenalan suara yang dijalankan pada komputer merupakan acuan utama bagi mikrokontroler untuk menentukan perangkat elektronik mana yang hendak dikendalikan.

5. Relay
   Pada sistem ini, relay berperan sebagai *switch* antara beban dengan sumber tegangan. Relay sepenuhnya dikendalikan oleh *input* sinyal (*low* atau *high*) yang diterima dari mikrokontroler. Rangkaian relay yang digunakan pada sistem ini adalah rangkaian relay dengan *low level trigger*.
6. Beban
   Beban adalah perangkat elektronik yang hendak dikendalikan oleh sistem. Beban yang digunakan bisa beragam. Namun pada sistem ini perangkat elektronik yang hendak dikendalikan adalah lampu, kipas angin, dan *lock door solenoid*.

B. Diagram Skematik Sistem

Diagram skematik *hardware* sistem pada penelitian ini dapat dilihat pada **Gambar 2**. Komponen utama skematik sistem ini adalah Arduino Nano dan relay.

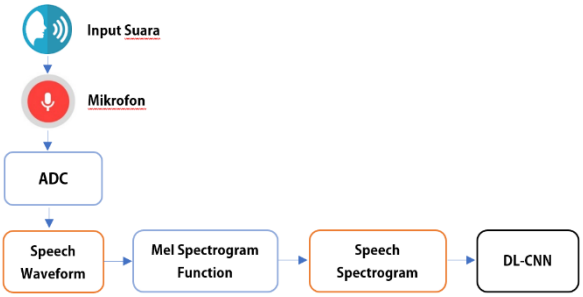


Gambar 2 Diagram skematik sistem

Arduino Nano secara *real-time* akan berkomunikasi dengan komputer dimana program pengenalan suara dijalankan. *Output* dari sistem pengenalan suara akan dijadikan sebagai acuan utama oleh mikrokontroler untuk mengirim *trigger* (*low* atau *high*) ke rangkaian relay. Seperti pada **Gambar 2** diatas, rangkaian yang digunakan pada sistem ini adalah rangkaian relay dengan *low level trigger*.

C. Konversi Speech Waveform ke Speech Spectrogram

Proses konversi sinyal audio dari *speech waveform* ke dalam bentuk *speech spectrogram* merupakan proses yang sangat penting pada sistem ini. Hal ini dikarenakan representasi sinyal audio dalam bentuk *spectrogram* merupakan *input* untuk CNN pada sistem ini. Berikut pada **Gambar 3** adalah diagram blok proses konversi *speech waveform* ke *speech spectrogram*.



Gambar 3 Diagram blok proses konversi speech waveform ke speech spectrogram

Berikut ini adalah penjelasan setiap bagian pada diagram blok diatas.

1. *Analog to Digital Converter (ADC)*
   Sinyal yang dideteksi oleh mikrofon pada dasarnya adalah sinyal analog. Oleh karena itu, sebelum sinyal suara tersebut dapat diproses oleh MATLAB, pertama-tama perlu mengubah sinyal analog tersebut menjadi sinyal digital. Frekuensi sampling yang digunakan pada proses ini adalah 16 KHz.
2. *Speech Waveform*
   Speech waveform merupakan sinyal hasil *sampling* dari sinyal analog yang dideteksi oleh mikrofon. Pada tahap ini, *speech waveform* yang ada merupakan sinyal digital yang memiliki frekuensi sampling 16 KHz, sesuai dengan inisialisasi yang ditentukan sebelumnya pada program MATLAB. *Speech waveform* direpresentasikan dalam bentuk grafik waktu terhadap amplitudo.
3. *Mel Spectrogram*
   Mel Spectrogram merupakan algoritma yang digunakan untuk menghitung *spectrogram* dari suatu sinyal audio. Pada algoritma ini, terdapat beberapa parameter yang harus diinisialisasi untuk keperluan komputasi tersebut, yakni:
   - *Window Length*
   - *Overlap Length*

- *FFT Length*
- *Numbands*
- *Frequency Range*

Pada algoritma ini sinyal audio digital pertama-tama di-*buffer* kedalam *frame* berdasarkan jumlah *samples Window Length*. Setiap *frame* kemudian mengalami proses *overlapped* bedasarkan jumlah *samples* pada *Overlap Length* yang diinisialisasi pada *script* matlab. Periodik *hamming window* kemudian akan diterapkan ke setiap *frame*. Fungsi matematika pada *hamming window* menggunakan persamaan (1) berikut.

$$w(n) = 0.54 - 0.46\cos\left(2\pi\frac{n}{N}\right), 0 \leq n \leq N \quad (1)$$

Dimana:
w = Hamming window
n = Time index
N = Number of samples

Frame tersebut kemudian akan ditransformasi dari domain waktu kedalam domain frekuensi menggunakan *Discreate Fourier Transform (DFT).* Pada perhitungan ini, diterapkan juga algoritma *Fast Fourier Transfor (FFT)* dengan tujuan untuk mempercepat proses komputasi DFT. Transformasi DFT menggunakan persamaan (2) berikut.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j2\pi\left(\frac{k}{N}\right)n} \quad (2)$$

Dimana:
x(k) = *Sample in frequency*
x(n) = *Sample in time*
n = *Time index*
k = *Frequency index*
N = *Number of samples*

Nilai *N* dari persamaan diatas ditentukan dari nilai *FFT Length* yang telah diinisialisasi sebelumnya pada *script* MATLAB. Setiap *frame* dari sinyal dalam domain frekuensi kemudian akan melalui *mel filter bank. Output* spektral dari *mel filter bank* dijumlahkan, dan setiap *channel*-nya digabungkan sehingga setiap *frame* ditransformasikan ke dalam vektor kolom *Numbands-element.*

D. Arsitektur CNN

Arsitektur CNN pada sistem ini terdiri dari 24 layer. **Gambar 4** merupakan diagram blok dari arsitektur CNN pada sistem ini. Arsitektur CNN pada sistem ini memiliki beberapa *layer* berulang sebanyak 5 kali. Setiap perulangan tersebut terdiri dari *convolutional layer, batch normalization layer,* dan *ReLU layer*. Arsitektur CNN ini akan melakukan proses komputasi terhadap *input* data sampai menghasilkan klasifikasi terhadap suatu kelas tertentu.

Pada **Gambar 5** adalah *flowchart* proses klasifikasi pada arsitektur CNN yang telah dirancang. Penjelasannya adalah sebagai berikut.

1. *Input* pada arsitektur CNN ini, merupakan gambar yang berukuran [40 x 98]. Gambar *input* tersebut adalah gambar yang diambil dari *speech spectrogram* dengan durasi satu detik. (3.1)
2. Pada *convolution layer* yang pertama, gambar *input* akan dikonvolusikan oleh filter yang berukuran [3 x 3], *stride* berukuran [1 x 1], dan *padding "same".* Dikarenakan *PaddingMode* yang digunakan adalah "same", maka sistem akan memberlakukan aturan zero padding terhadap *input* matriks. Nilai dari *zero padding* dapat ditentukan menggunakan persamaan (3) berikut ini.

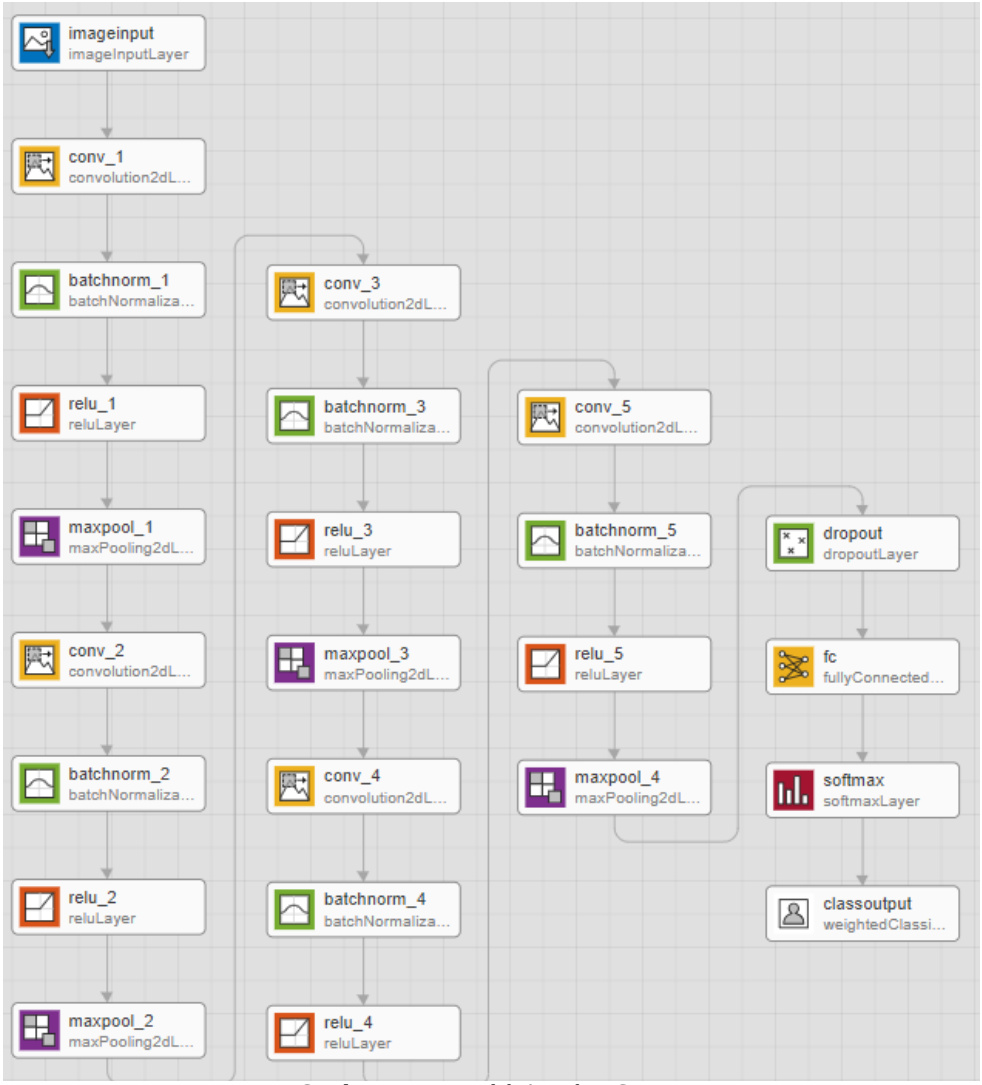
$$Zero\ Padding\ (P) = \frac{K-1}{2} \quad (3)$$

Dimana:
K = Filter Size
Sedangkan *output size* dari c*onvolution layer* dapat dihitung dengan menggunakan persamaan (4) berikut ini. (3.2)

$$Output\ Size\ (O) = \frac{W-K+2P}{S} + 1 \quad (4)$$

Dimana :
W = *Input Size*
K = *Filter Size*
P = *Padding*
S = *Stride*

Dikarenakan *PaddingMode* yang digunakan dalam *mode "same"* dan ukuran *stride* yang digunakan adalah [1 x 1], maka dapat dipastikan bahwa *output size* dari *convolution layer* ini adalah sama dengan ukuran *input size* dari gambar *input* yakni [40 x 98]. Setelah itu, *output* dari *convolution layer* akan melewati *batch normalization layer* yang fungsinya adalah untuk mempercepat proses *training* dari CNN. Selain itu, *input* gambar juga akan melalui *ReLU layer* yang bertujuan untuk mengubah nilai minus pada *output* proses konvolusi pada *convolution layer* menjadi nol.



Gambar 4 Diagram Blok Arsitektur CNN

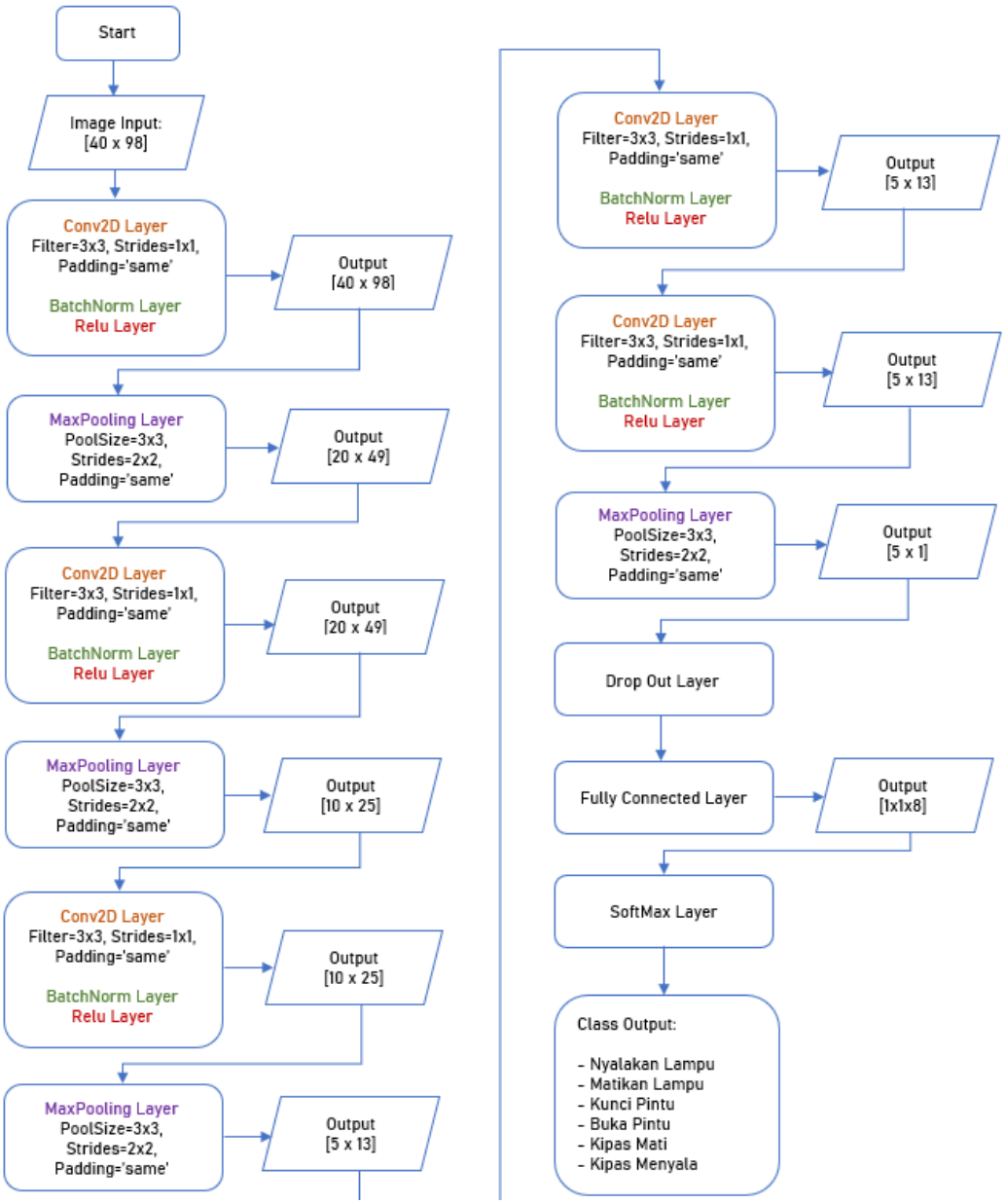
3. Setelah melewati proses konvolusi pada *convolution layer*, maka *output* dari proses konvolusi pertama tersebut selanjutnya akan menjadi *input* untuk *max pooling layer*. Pada *layer* ini, gambar akan di *downsampling* menggunakan *pool size* berukuran [3 x 3], *stride* [2 x 2], dan *PaddingMode 'same'*. Nilai dari padding pada layer ini dapat digitung menggunakan persamaan (1) seperti yang terdapat pada convolution layer . Sedangkan *output size* pada *max pooling layer* menggunakan persamaan (5) berikut.

$$Output\ [h\ w] = \frac{W - Ps + (2 \times P)}{S} + 1 \qquad (5)$$

Dimana:
W = *Input Size*
Ps = *Pool Size*
P = *Padding*
S = *Stride*

Output dari *max pooling layer* berdasarkan persamaan (3) dan (5) adalah [20 x 49].

4. *Output* dari *max pooling layer* yang berukuran [20 x 49] pada proses sebelumnya, kemudian akan menjadi *input* pada proses kovolusi kedua dengan filter berukuran [3 x 3], *stride* berukuran [1 x 1], dan *PaddingMode 'same'*. Dengan memasukkan nilai-nilai tersebut kedalam persamaan (3) dan (4), maka ukuran *output* dari proses konvolusi yang kedua adalah sama dengan ukuran *input*-nya yakni [20 x 49].
5. Setelah itu, *output* dari *convolution layer* yang kedua akan di *downsampling* lagi pada *max pooling layer* dengan *pool size* berukuran [3 x 3], *stride* [2 x 2], dan *PaddingMode 'same'*. Dengan memasukkan nilai-nilai tersebut kedalam persamaan (3) dan (5) maka akan didapat *output* matriks dengan ukuran [10 x 25].



Gambar 5 Flowchart proses klasifikasi

6. *Output* matriks dari *max pooling layer* pada proses sebelumnya yang berukuran [10 x 25], kemudian akan menjadi *input* pada proses konvolusi yang ketiga dengan filter berukuran [3 x 3], *stride* [2 x 2], dan *Padding Mode 'same'*. Dengan memasukkan nilai-nilai tersebut kedalam persamaan (3) dan (4), maka ukuran *output* dari proses konvolusi yang ketiga adalah sama dengan ukuran *input*-nya yakni [10 x 25].
7. Setelah itu, *output* dari *layer* konvolusi yang ketiga yang berukuran [10 x 25] akan di *downsampling* lagi pada *max pooling layer* dengan *pool size* berukuran [3 x 3], *stride* [2 x 2], dan *PaddingMode 'same'*. Dengan memasukkan nilai-nilai tersebut kedalam persamaan (3) dan (5) maka akan didapat *output* matriks dengan ukuran [5 x 13].
8. Setelah itu, *output* matriks yang berukuran [5 x 13] dari *max pooling layer*, akan dikonvolusi berturut-turut menggunakan dua *convolution layer* sekaligus dengan filter berukuran [3 x 3], *strides* [1 x1], dan *PaddingMode 'same'*. Dari kedua proses konvolusi ini, *output* yang dihasilkannya adalah matriks yang berukuran sama dengan *input*-nya yakni [5 x 13].
9. Dari hasil konvolusi pada *layer* sebelumnya yang menghasilkan matriks [5 x 13], maka data tersebut selanjutnya akan melewati

proses *downsampling* terakhir pada *max pooling layer* dengan *pool size* berukuran [1 x 13], *stride* [1 x 1], dan *padding* [0,0,0,0]. Dengan memasukkan nilai-nilai tersebut kedalam persamaan (3) dan (5), maka akan didapatkan *output* dengan ukuran [5 x 1].

10. Data pada *layer* sebelumnya akan diproses pada *dropout layer*. *Dropout layer* bertujuan untuk mencegah *overfitting* dan juga mempercepat proses *training* dengan cara beberapa membuang beberapa neuron secara acak berdasarkan probabilitas yang diberikan.
11. *Fully connected layer* pada arsitektur ini akan mengolah data dengan melakukan transformasi pada dimensi data agar dapat diklasifikasikan. Dalam proses klasifikasinya, *fully connected layer* akan menggabungkan setiap *feature* yang ada untuk mengklasifikasikan gambar *input*. Banyaknya data yang dihasilkan pada proses ini akan sama dengan jumlah kelas data yang ada.
12. *Layer* terakhir pada proses klasifikasi adalah *softmax layer*. *Layer* ini akan menghitung probabilitas gambar *input* terhadap semua kelas target yang memungkinkan dan kemudian akan menentukan kelas target berdasarkan *input* yang diberikan.

E. Training Deep Learning Convolutional Neural Network (DL-CNN)

Pada sistem ini terdapat enam macam perintah suara yang akan dijadikan sebagai perintah untuk mengendalikan perangkat elektronik. *Data set* disimpan pada suatu direktori yang bernama 'Data-Training'. Dalam direktori 'Data-Training' terdapat beberapa *sub*-direktori, yang mana setiap *sub*-direktori tersebut memuat kumpulan data yang saling berkaitan. Data-data tersebutlah yang akan dijadikan sebagai *data training* pada sistem ini. Kelas-kelas data training adalah "Nyalakan Lampu", "Matikan Lampu", "Kunci Pintu", Buka Pintu", "Kipas Menyala", "Kipas Mati", "*Unknown"*, dan "*Background"*

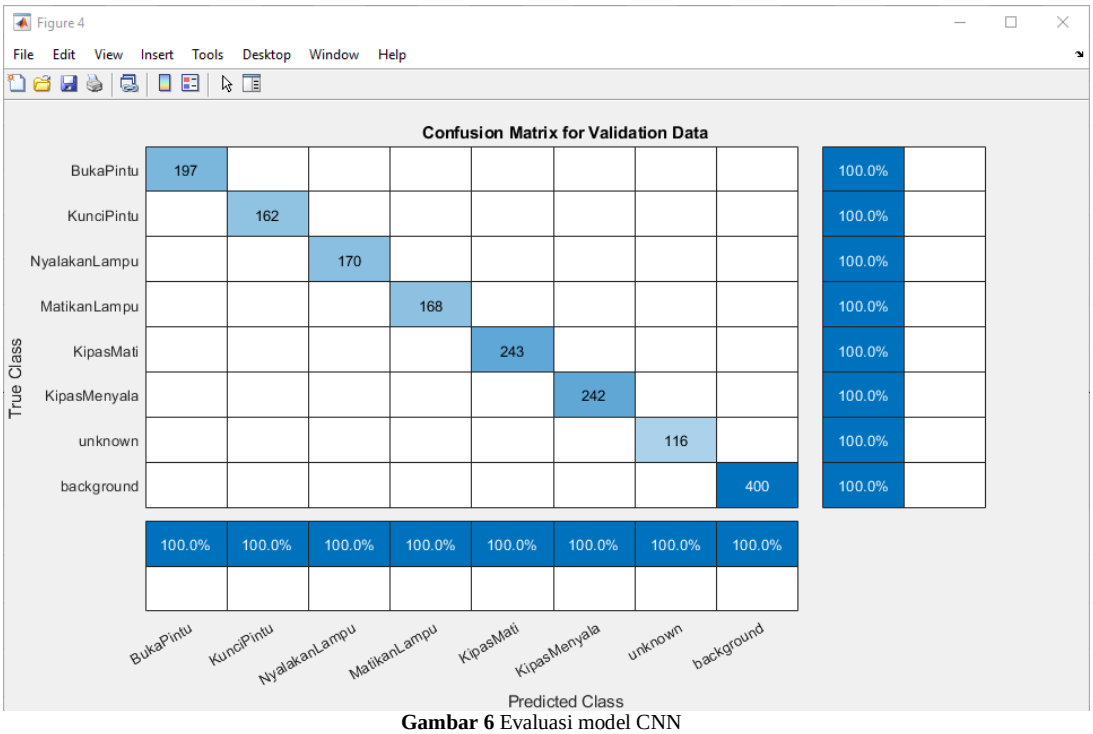
Kelas data utama yang hendak dijadikan sebagai acuan untuk mengendalikan perangkat elektronik adalah Nyalakan Lampu, Matikan Lampu, Kunci Pintu, Buka Pintu, Kipas Menyala, dan Kipas Mati. Ke-enam kelas data utama tersebut akan dilatih menggunakan *data training* yang berjumlah sekitar 2000 data untuk setiap kelas. *Training* arsitektur CNN pada sistem ini menggunakan algoritma *Adam Optimizer*. Algoritma ini bekerja dengan menghitung *learning rates* untuk setiap parameter yang berbeda.

Pada sistem ini, nilai *epoch* yang digunakan adalah 25, yang berarti bahwa arsitektur CNN akan mengulang siklus belajar dari *data training* sebanyak 25 kali. Setelah proses *training* selesai, sistem akan mengevaluasi model CNN yang telah dilatih dengan menggunakan data validasi yang sebelumnya telah dipisahkan dari *data training*. Proses ini bertujuan untuk mengukur persentase keakuratan klasifikasi dari model CNN yang telah dilatih. **Gambar 6** adalah hasil evaluasi dari model CNN terhadap setiap kelas data.

F. Pendeteksian Perintah Suara Menggunakan Model CNN Untuk Mengendalikan Perangkat Elektronik

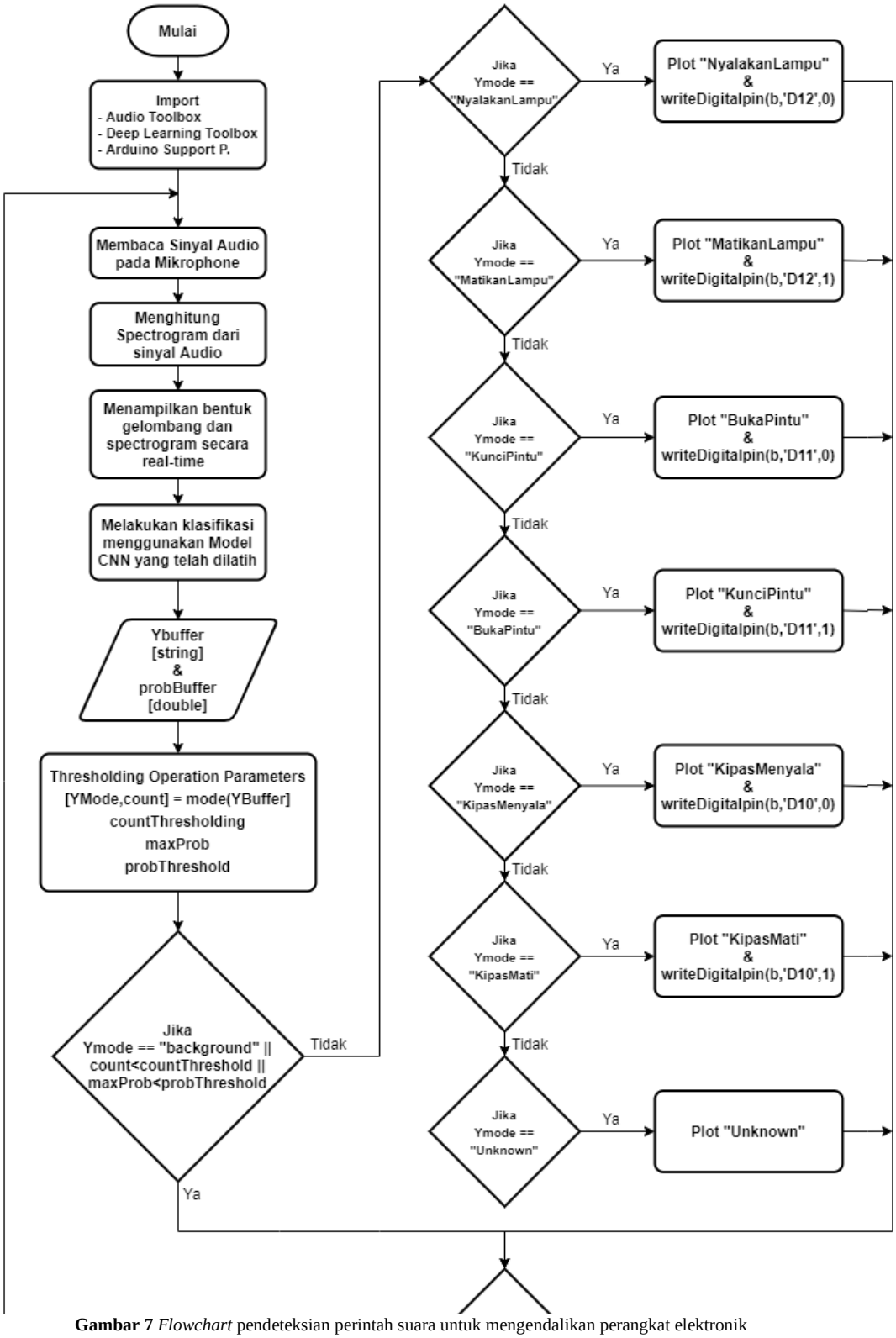
Hasil dari proses *training* model CNN yang telah dilakukan sebelumnya akan menghasilkan sebuah model CNN dengan nilai bobot dan bias tertentu. Model CNN yang telah di-*training* inilah yang akan digunakan untuk mengklasifikasikan perintah suara yang dideteksi oleh mikrofon secara *real time* untuk mengendalikan perangkat elektronik tertentu. Untuk lebih lengkapnya, *flowchart* dari sistem kendali ini dapat dilihat pada **Gambar 7.**
Berikut ini adalah penjelasan dari *flowchart* seperti yang terdapat pada **Gambar 7**.

1. Proses pertama dimulai dengan meng-*import* beberapa *library/toolbox* yang hendak digunakan pada sistem ini. Terdapat tiga *library* yang digunakan, yakni *audio toolbox*, *deep learning toolbox,* dan *arduino support package*. *Audio toolbox* merupakan *library* yang digunakan untuk membaca sinyal audio yang dideteksi mikrofon. *Deep learning toolbox* merupakan *library* yang digunakan untuk membuat arsitektur *deep learning convolutional neural network*. Selain itu, *deep learning toolbox* juga digunakan untuk melatih arsitektur CNN yang telah dibuat. *Library* yang terakhir adalah *arduino support package*, yang berfungsi untuk mengatur komunikasi *serial* antara MATLAB dengan *board* mikrokontroler arduino. *Library arduino support package* ini juga memungkinkan *board* mikrokontroler arduino untuk dapat diprogram menggunakan bahasa pemrograman MATLAB.



Gambar 6 Evaluasi model CNN

2. Lalu sistem akan membaca sinyal audio yang dideteksi oleh mikrofon. Pada proses ini, *soundcard* yang tertanam pada komputer merupakan perangkat utama yang akan digunakan untuk membaca sinyal audio pada mikrofon. Sinyal audio yang akan dibaca oleh MATLAB merupakan sinyal digital yang sebelumnya telah di-*sampling* oleh *soundcard* menggunakan ADC dengan frekuensi *sampling* 16 KHz.
3. Sinyal audio yang terbaca kemudian akan dikonversi ke dalam bentuk *spectrogram*. Proses konversi ini menggunaan *mel spectrogram function* pada MATLAB.
4. *Spectrogram* dari sinyal audio kemudian akan digunakan sebagai *input* untuk model CNN yang telah dilatih sebelumnya. Model CNN yang telah dilatih pada sistem ini diprogram untuk dapat melakukan klasifikasi terhadap delapan kelas data yakni 'MatikanLampu', 'NyalakanLampu', 'KunciPintu', 'BukaPintu', 'KipasMenyala', 'KipasMati', *'Unknown'*, dan *'background'*.
5. *Output* proses klasifikasi oleh model CNN terbagi atas dua jenis data yakni label dan probabilitas. Label merupakan data *string* yang berkaitan dengan kelas data yang ada pada model CNN. Data *string* tersebut kemudian dideklarasikan sebagai Ybuffer. Sedangkan probabilitas adalah data *double* atau data desimal yang menunjukkan nilai probabilitas dari setiap kelas data terhadap *speech spectrogram* yang diproses oleh model CNN. Data probabilitas dideklarasikan sebagai profBuffer.
6. *Output* dari proses klasifikasi yakni Ybuffer dan probBuffer kemudian digunakan untuk menghitung parameter-parameter yang hendak digunakan dalam proses selanjutnya.
7. *Thresholding operation* merupakan perhitungan yang bertujuan untuk mendeklarasikan hasil deteksi model CNN [11]. Deklarasi hasil deteksi tersebut kemudian akan dijadikan sebagai acuan untuk menentukan *output* dari pin mikrokontroler arduino yang terhubung ke rangkaian relay.
8. Sistem ini dirancang untuk bekerja secara *real time* memproses sinyal audio yang dideteksi oleh mikrofon dan kemudian melakukan klasifikasi menggunakan model CNN untuk mengendalikan perangkat elektronik yang terhubung ke *board* mikrokontroler arduino. Hasil deteksi akan ditampilkan pada *window detection* bersamaan dengan bentuk *speech waveform* dan *speech spectrogram* dari sinyal audio yang dideteksi mikrofon. Untuk menyelesaikan atau menghentikan proses deteksi, dapat dilakukan dengan menutup *window detection*, sehingga dengan demikian sistem akan berhenti beroperasi.



Gambar 7 *Flowchart* pendeteksian perintah suara untuk mengendalikan perangkat elektronik

III. HASIL DAN PEMBAHASAN

Pengujian terhadap sistem yang telah dirancang dilakukan melalui dua jenis pengujian, yakni pengujian sistem pengenalan suara dan pengujian sistem pengendalian perangkat elektronik menggunakan perintah suara. Berikut ini adalah hasil dari kedua pengujian tersebut.

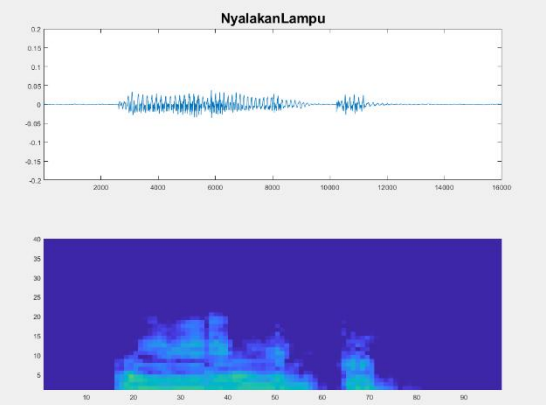
A. Pengujian sistem pengenalan suara

Pengujian sistem pengenalan suara dilakukan secara langsung pada beberapa kondisi yang berbeda. Berikut adalah hasil pengujian yang telah dilakukan.

1. Pengujian pada intensitas *background noise* 24dB (senyap)

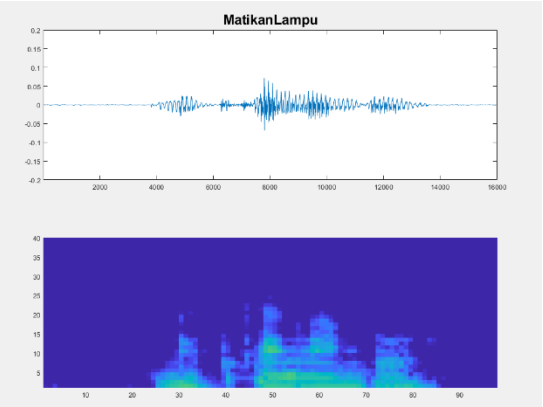
   Pengujian ini dilakukan pada ruangan dengan intensitas *background noise* rata-rata 24dB. Masing-masing perintah suara yang telah dilatih pada model CNN diuji untuk mengetahui *output* klasifikasi dari model CNN tersebut. Berikut ini pada **Gambar 8** sampai **Gambar 13** adalah hasil pengujiannya.

   a. Perintah "Nyalakan Lampu"



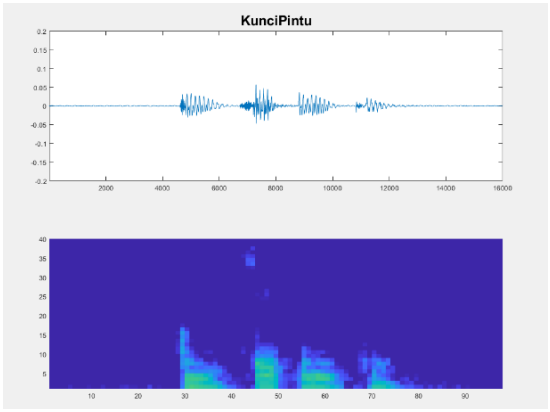
Gambar 8 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Nyalakan Lampu' pada intensitas *background noise* 24dB

   b. Perintah "Matikan Lampu"



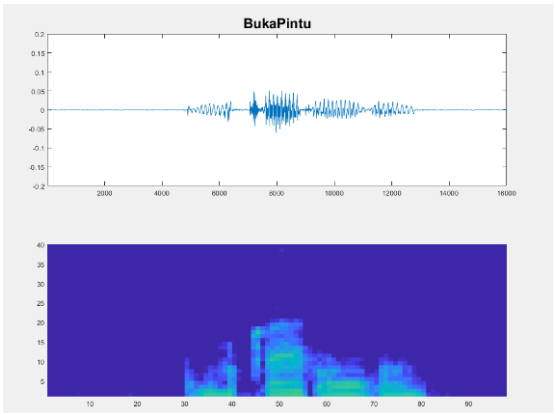
Gambar 9 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Matikan Lampu' pada intensitas *background noise* 24dB

   c. Perintah "Kunci Pintu"



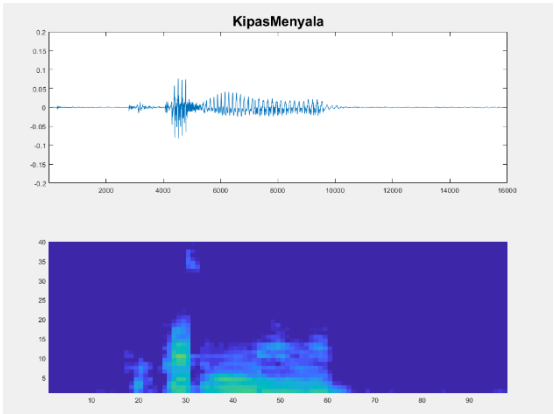
Gambar 10 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Kunci Pintu' pada intensitas *background noise* 24dB

   d. Perintah "Buka Pintu"



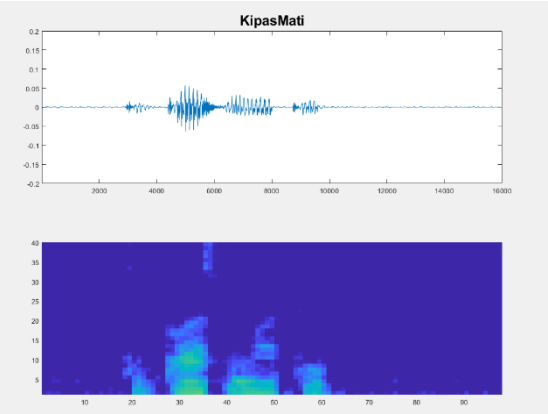
Gambar 11 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Buka Pintu' pada intensitas *background noise* 24dB

   e. Perintah "Kipas Menyala"



Gambar 12 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Kipas Menyala' pada intensitas *background noise* 24dB

f. Perintah Kipas Mati



Gambar 13 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Kipas Mati' pada intensitas *background noise* 24dB

Gambar 8 sampai **Gambar 13** adalah bentuk *speech waveform* dan *speech spectrogram* dari hasil pengujian yang telah dilakukan pada masing-masing perintah. Untuk mengetahui akurasi sistem dalam mendeteksi perintah suara, maka pengujian ini dilakukan sebanyak 10 kali pada setiap perintah suara. Berikut pada **Tabel I** adalah hasil pengujian tersebut.

Tabel I. Hasil pengujian pada intensitas *background noise* 24dB (senyap)

No	Perintah Suara	Jumlah Uji	Jumlah Deteksi Benar
1	Nyalakan Lampu	10	10
2	Matikan Lampu	10	10
3	Kunci Pintu	10	10
4	Buka Pintu	10	10
5	Kipas Mati	10	10
6	Kipas Menyala	10	10

Dari hasil pengujian yang telah dilakukan, persentase keberhasilan deteksi dapat dihitung menggunakan persamaan (6).

$$\%\ Berhasil = \frac{Jumlah\ Deteksi\ Benar}{Jumlah\ Uji} \times 100\ \% \quad (6)$$

a. Persentase keberhasilan mengenali perintah "Nyalakan Lampu"
$\frac{10}{10} \times\ 100\% = 100\ \%$

b. Persentase keberhasilan mengenali perintah "Matikan Lampu"
$\frac{10}{10} \times\ 100\% = 100\ \%$

c. Persentase keberhasilan mengenali perintah "Kunci Pintu"
$\frac{10}{10} \times\ 100\% = 100\ \%$

d. Persentase keberhasilan mengenali perintah "Buka Pintu"
$\frac{10}{10} \times\ 100\% = 100\ \%$

e. Persentase keberhasilan mengenali perintah "Kipas Menyala"
$\frac{10}{10} \times\ 100\% = 100\ \%$

f. Persentase keberhasilan mengenali perintah "Kipas Mati"
$\frac{10}{10} \times\ 100\% = 100\ \%$

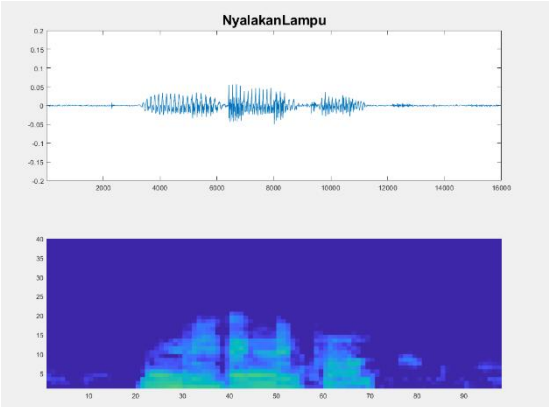
Dengan menggunakan persamaan (6), dapat dihitung persentase keberhasilan total sistem mengenali perintah suara pada intensitas *background noise* 24db (senyap) sebagai berikut:

$$\frac{60}{60} \times\ 100\% = 100\ \%$$

2. Pengujian pada intensitas *background noise* 42dB (senyap)

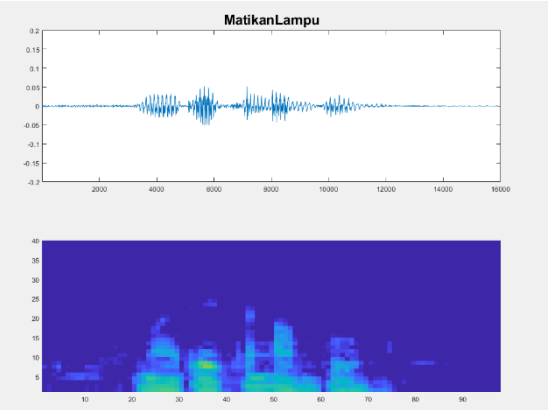
Pengujian ini dilakukan pada ruangan dengan intensitas *background noise* rata-rata 42dB. Masing-masing perintah suara yang telah dilatih pada model CNN diuji untuk mengetahui *output* klasifikasi dari model CNN tersebut. Berikut ini pada **Gambar 14** sampai **Gambar 19** adalah hasil pengujiannya.

a. Perintah "Nyalakan Lampu"



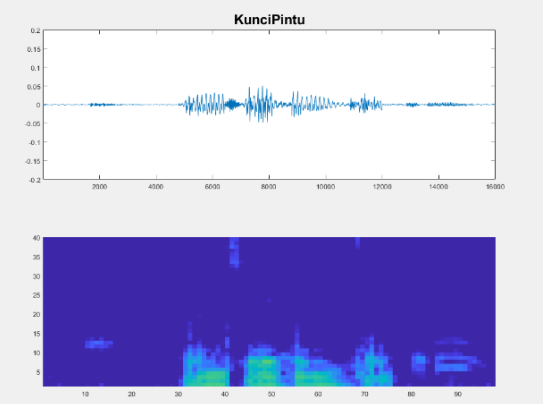
Gambar 14 Bentuk *speech waveform* dan *speech spectrogram* perintah 'Nyalakan Lampu' pada intensitas *background noise* 42dB

b. Perintah “Matikan Lampu”



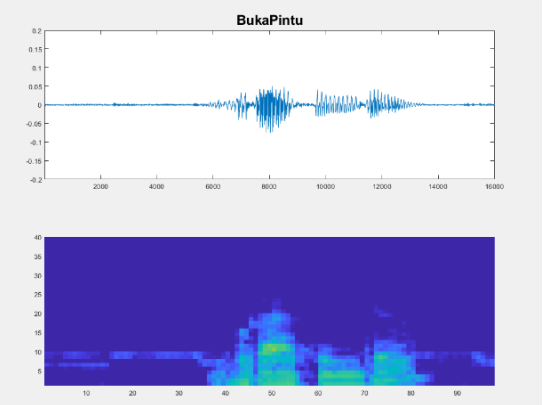
Gambar 15 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Matikan Lampu’ pada intensitas *background noise* 42dB

c. Perintah “Kunci Pintu”



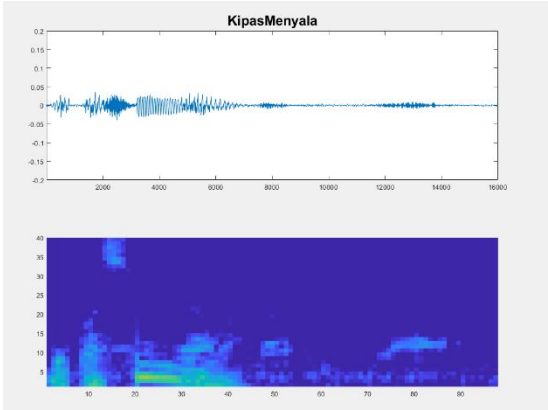
Gambar 16 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Kunci Pintu’ pada intensitas *background noise* 42dB

d. Perintah “Buka Pintu”



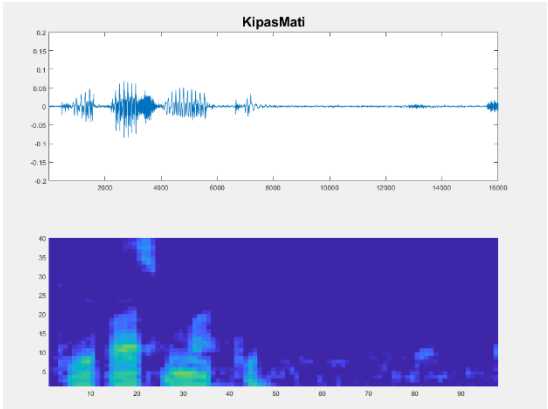
Gambar 17 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Buka Pintu’ pada intensitas *background noise* 42dB

e. Perintah “Kipas Menyala”



Gambar 18 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Kipas Menyala’ pada intensitas *background noise* 42dB

f. Perintah Kipas Mati



Gambar 19 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Kipas Mati’ pada intensitas *background noise* 42dB

Gambar 14 sampai **Gambar 19** adalah bentuk *speech waveform* dan *speech spectrogram* dari hasil pengujian yang telah dilakukan pada masing-masing perintah. Untuk mengetahui akurasi sistem dalam mendeteksi perintah suara, maka pengujian ini dilakukan sebanyak 10 kali pada setiap perintah suara. Berikut pada **Tabel II** adalah hasil pengujian tersebut.

Tabel II. Hasil pengujian pada intensitas *background noise* 42dB

No	Perintah Suara	Jumlah Uji	Jumlah Deteksi Benar
1	Nyalakan Lampu	10	7
2	Matikan Lampu	10	6
3	Kunci Pintu	10	6
4	Buka Pintu	10	8
5	Kipas Mati	10	6
6	Kipas Menyala	10	7

Dari hasil pengujian yang telah dilakukan, persentase keberhasilan deteksi dapat dihitung menggunakan persamaan (6).

a. Persentase keberhasilan mengenali perintah “Nyalakan Lampu”

$$\frac{7}{10} \times 100\% = 70\ \%$$

b. Persentase keberhasilan mengenali perintah “Matikan Lampu”

$$\frac{6}{10} \times 100\% = 60\ \%$$

c. Persentase keberhasilan mengenali perintah “Kunci Pintu”

$$\frac{6}{10} \times 100\% = 60\ \%$$

d. Persentase keberhasilan mengenali perintah “Buka Pintu”

$$\frac{8}{10} \times 100\% = 80\ \%$$

e. Persentase keberhasilan mengenali perintah “Kipas Menyala”

$$\frac{6}{10} \times 100\% = 60\ \%$$

f. Persentase keberhasilan mengenali perintah “Kipas Mati”

$$\frac{7}{10} \times 100\% = 70\ \%$$

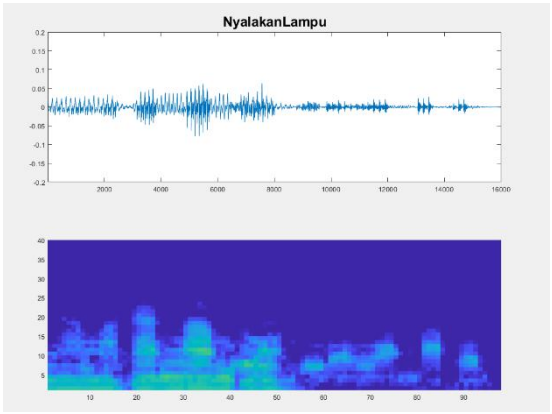
Dengan menggunakan persamaan (6), dapat dihitung persentase keberhasilan total sistem mengenali perintah suara pada intensitas *background noise* 42db (senyap) sebagai berikut:

$$\frac{40}{60} \times 100\% = 66.67\ \%$$

3. Pengujian pada intensitas *background noise* 52dB (senyap)

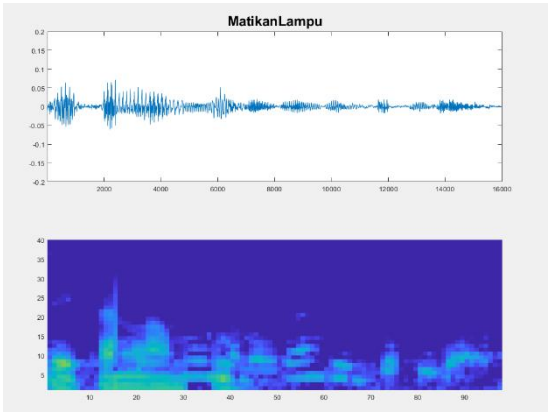
Pengujian ini dilakukan pada ruangan dengan intensitas *background noise* rata-rata 52dB. Masing-masing perintah suara yang telah dilatih pada model CNN diuji untuk mengetahui *output* klasifikasi dari model CNN tersebut. Berikut ini pada **Gambar 20** sampai **Gambar 25** adalah hasil pengujiannya.

a. Perintah “Nyalakan Lampu”



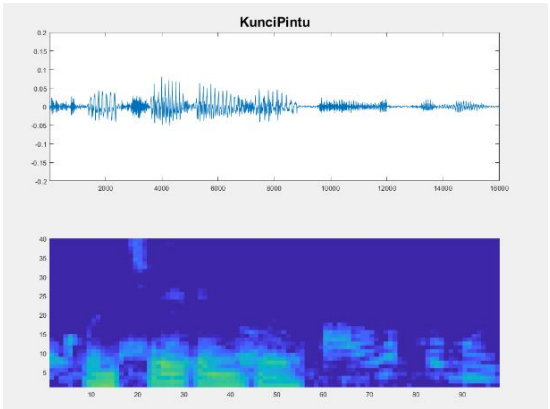
Gambar 20 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Nyalakan Lampu’ pada intensitas *background noise* 52dB

b. Perintah “Matikan Lampu”



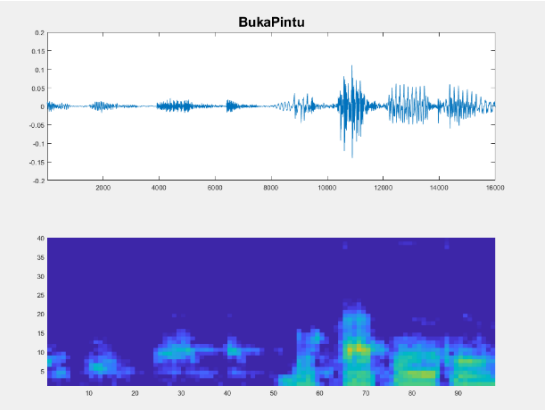
Gambar 21 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Matikan Lampu’ pada intensitas *background noise* 52dB

c. Perintah “Kunci Pintu”



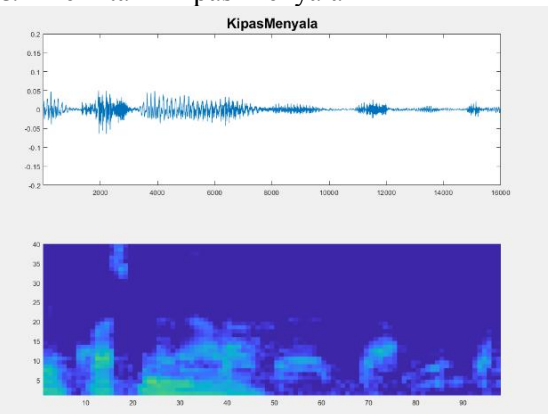
Gambar 22 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Kunci Pintu’ pada intensitas *background noise* 52dB

d. Perintah “Buka Pintu”



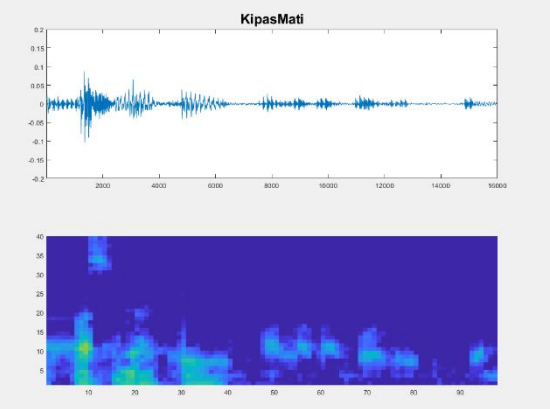
Gambar 23 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Buka Pintu’ pada intensitas *background noise* 52dB

e. Perintah “Kipas Menyala”



Gambar 24 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Kipas Menyala’ pada intensitas *background noise* 52dB

f. Perintah Kipas Mati



Gambar 25 Bentuk *speech waveform* dan *speech spectrogram* perintah ‘Kipas Mati’ pada intensitas *background noise* 52dB

Gambar 20 sampai **Gambar 25** adalah bentuk *speech waveform* dan *speech spectrogram* dari hasil pengujian yang telah dilakukan pada masing-masing perintah.

Untuk mengetahui akurasi sistem dalam mendeteksi perintah suara, maka pengujian ini dilakukan sebanyak 10 kali pada setiap perintah suara. Berikut pada **Tabel III** adalah hasil pengujian tersebut.

Tabel III. Hasil pengujian pada intensitas *background noise* 52dB (bising)

No	Perintah Suara	Jumlah Uji	Jumlah Deteksi Benar
1	Nyalakan Lampu	10	5
2	Matikan Lampu	10	4
3	Kunci Pintu	10	5
4	Buka Pintu	10	6
5	Kipas Mati	10	5
6	Kipas Menyala	10	6

Dari hasil pengujian yang telah dilakukan, persentase keberhasilan deteksi dapat dihitung menggunakan persamaan (6).

a. Persentase keberhasilan mengenali perintah “Nyalakan Lampu”

$$\frac{5}{10} \times 100\% = 50\ \%$$

b. Persentase keberhasilan mengenali perintah “Matikan Lampu”

$$\frac{4}{10} \times 100\% = 40\ \%$$

c. Persentase keberhasilan mengenali perintah “Kunci Pintu”

$$\frac{5}{10} \times 100\% = 50\ \%$$

d. Persentase keberhasilan mengenali perintah “Buka Pintu”

$$\frac{6}{10} \times 100\% = 60\ \%$$

e. Persentase keberhasilan mengenali perintah “Kipas Menyala”

$$\frac{5}{10} \times 100\% = 50\ \%$$

f. Persentase keberhasilan mengenali perintah “Kipas Mati”

$$\frac{6}{10} \times 100\% = 60\ \%$$

Dengan menggunakan persamaan (6), dapat dihitung persentase keberhasilan total sistem mengenali perintah suara pada intensitas *background noise* 52db (bising) sebagai berikut:

$$\frac{31}{60} \times 100\% = 51.67\ \%$$

B. Pengujian sistem pengendalian perangkat elektronik menggunakan perintah suara

Pengujian sistem pengendalian perangkat elektronik pada tahap ini dilakukan menggunakan sebuah komputer yang di dalamnya telah terpasang *software* MATLAB R2019a. Pengujian dilakukan dengan menjalankan sistem pengenalan suara menggunakan MATLAB R2019a dan kemudian menyebutkan secara langung perintah-perintah yang telah dilatih pada model CNN. Tujuan dari pengujian itu adalah melihat apakah sistem dapat mengendalikan perangkat elektronik dengan benar berdasarkan perintah suara yang diberikan. Hasil pengujian pada tahap ini dapat dilihat pada **Tabel IV** berikut

Tabel IV. Hasil pengujian sistem pengendali perangkat elektronik menggunakan perintah suara

No	**Perintah Suara**	**Jumlah Uji**	**Jumlah Deteksi Benar**
1	Nyalakan Lampu	10	10
2	Matikan Lampu	10	10
3	Kunci Pintu	10	10
4	Buka Pintu	10	10
5	Kipas Mati	10	10
6	Kipas Menyala	10	10

Dengan menggunakan persamaan (6), maka dapat diperoleh tingkat keberhasilan sistem untuk mengendalikan perangkat elektronik melalui komunikasi *serial* antara MATLAB dan Arduino adalah sebagai berikut.

$$\frac{60}{60} \times 100\% = 100\ \%$$

C. Analisis Hasil Pengujian

Berdasarkan persentase keberhasilan pengenalan perintah suara yang telah dilakukan, terdapat beberapa hal yang sangat berpengaruh terhadap keakurasian sistem dalam mengenali perintah suara. Berikut ini adalah hal-hal yang mempengaruhi keakurasian sistem.

1. Intensitas *background noise*
   Berdasarkan hasil pengujian yang telah dilakukan sebelumnya, dapat diketahui bahwa akurasi pengenalan perintah suara pada kondisi senyap dengan kondisi bising sangat berbeda. Pada kondisi senyap (intensitas *background noise* 24dB) akurasi pendeteksian perintah suara mencapai 100%. Sedangkan pada kondisi bising (intensitas *background noise* 52dB) akurasi pendeteksian perintah suara hanya 51,67%. Hasil ini memiliki sedikit kemiripan dengan penelitian sebelumnya yang berjudul "*Applying Voice Recognition Technology for Smart Home Network*" [9]. Pada penelitian tersebut, persentase akurasi sistem pada kondisi senyap adalah 80%. Sedangkan pada kondisi bising, persentase keakuratannya turun menjadi 60%. Dari hasil tersebut, dapat diketahui bahwa intensitas *background noise* sangat mempengaruhi akurasi dari sistem pengenalan suara.
2. *Data Training*
   Hal lain yang memiliki pengaruh signifikan pada akurasi dari sistem pengenalan suara pada penelitian ini adalah *data training*. Pada penelitian ini, *data training* yang digunakan pada setiap kelas berada pada kisaran 1700 sampai 2500 data. Algoritma *deep learning convolutional neural network* pada penelitian ini, melakukan ekstraksi ciri pada setiap *data training* yang ada untuk nantinya dijadikan sebagai acuan pada proses klasifikasi. Oleh karena itu, semakin banyak dan beragam *data training* yang ada pada setiap kelas, maka akan semakin akurat pula model CNN dalam melakukan proses klasifikasi secara *real-time.*

Sistem secara *real time* melakukan proses klasifikasi terhadap sinyal audio yang dideteksi mikrofon dan hanya akan berhenti beroperasi ketika jendela *window detection* yang menampilkan *speech waveform* dan *speech spectrogram* ditutup. Keberhasilan proses pengendalian perangkat elektronik pada penelitian ini berhubungan secara langsung terhadap keberhasilan sistem pengenalan suara dalam mendeteksi perintah suara dengan benar. Hal ini dikarenakan, pada program MATLAB yang dibuat, sistem akan secara *real-time* membaca *output* klasifikasi dari model CNN. *Output* dari model CNN kemudian akan dijadikan sebagai acuan untuk mengendalikan perangkat elektronik yang terhubung ke setiap *pin* pada *board* arduino. Kesalahan model CNN dalam mendeteksi perintah suara, secara langsung juga akan mengakibatkan kesalahan pada proses pengendalian perangkat elektronik.

IV. KESIMPULAN

Berdasarkan pengujian yang telah dilakukan, data ditarik beberapa kesimpulan. Sistem pengenalan suara pada penelitian ini memiliki persentase keberhasilan sebesar 100% pada kondisi ruangan dengan intensitas *background noise* 24dB (senyap), sebesar 67,67% pada kondisi ruangan dengan intensitas *background noise* 42dB, dan sebesar 51,67% pada kondisi ruangan dengan intensitas *background noise* 52dB (bising). Dari hasil tersebut dapat disimpulkan bahwa semakin senyap kondisi suatu ruangan, maka akan semakin baik pula akurasi dari sistem pengenalan suara pada penelitian ini. Sebaliknya semakin bising kondisi suatu ruangan, maka akurasi dari sistem pengenalan suara akan semakin menurun. Oleh karena itu, untuk memperoleh hasil yang optimal, sistem ini lebih cocok digunakan pada ruangan dengan intensitas *background noise* yang rendah.

Sistem pengenalan suara menggunakan DL-CNN pada penelitian ini dirancang untuk dapat diaplikasikan pada *home automation*. Perintah-perintah yang dapat dikenali oleh sistem pengenalan suara pada sistem ini adalah sebagai berikut.

- Nyalakan Lampu
- Matikan Lampu
- Kunci Pintu
- Buka Pintu
- Kipas Mati
- Kipas Menyala

Persentase keberhasilan sistem mengendalikan perangkat elektronik berdasarkan output dari sistem pengenalan suara pada penelitian ini adalah sebesar 100%. Hal ini berarti bahwa sistem kendali perangkat elektronik mampu merespon setiap output dari model CNN pada sistem pengenalan suara dengan sempurna.

Sistem pengenalan suara pada penelitian ini mampu melakukan klasifikasi terhadap sinyal audio dengan durasi maksimal satu detik. Sistem akan mengambil representasi dua dimensi dari sinyal audio (*spectrogram*) setiap satu detik, dan kemudian langsung melakukan klasifikasi terhadap data tersebut. Sistem pengenalan suara pada penelitian ini dapat bekerja secara *stand alone* pada sebuah komputer komersial tanpa harus menggunakan komputer dengan spesifikasi khusus. Ini dapat dicapai karena sistem pengenalan suara pada penelitian ini dibuat menggunakan jaringan *feed forward* dengan variasi *multi layer perceptrons* yang dirancang untuk melakukan *preprocessing* dalam jumlah minimal.

Sistem pengenalan suara pada penelitian ini dapat dikembangkan lagi dari segi akurasinya. Sehingga dengan demikian diharapkan bahwa, siste tidak hanya memiliki akurasi yan baik pada kondisi ruangan senyap, namun juga memiliki akurasi yang baik pada kondisi ruangan yang bising. Sistem pengenalan suara pada penelitian ini hanya mampu memproses perintah suara dengan durasi pengucapan maksimal 1 detik. Pengembangan selanjutnya bisa dilakukan dengan menambah kemampuan sistem dalam mendeteksi perintah suara dengan durasi pengucapan yang lebih lama. Sehingga dengan demikian, sistem pengenalan suara dapat di program untuk mengenali perintah suara yang lebih beragam dan variatif.

DAFTAR PUSTAKA

[1] G. Matt, *Essentials of Artificial Intelligence*. 2012.

[2] N. Giang, D. Stefan, B. Martin, T. Viet, L. P. Alvaro, H. Ignacio, M. Peter and H. Ladislav, "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," *Artificial Intelligence Review*, vol. 52, pp. 77-124, Jan. 2019.

[3] S. Jurgen, "Deep learning in neural networks: An overview," *Neural Networks*, vol.61, pp. 85-117, Jan 2015.

[4] B. Francoise, "The neural networks behind Google Voice Transcription," 2015. [Online]. Available: https://ai.googleblog.com/2015/08/the-neural-networks-behind-google-voice.html. [Accessed: 12-Feb-2020].

[5] O. Kalin, R. Olatunji, Y. K. Joo, F. Jeremy, S. Karin, S. C. Eric, "Accelerating Deep Convolutional Neural Networks Using Specialized Hardware," Feb. 2015.

[6] Z. Ying, P. Mohammad, B. Philemon, Z. Saizheng, L. Cesar, B. Yoshua and C. Aaron, "Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks," Jan. 2017.

[7] R. Tri, Wahyudi, "Sistem Keamanan Rumah Dengan Monitoring Menggunakan Jaringan Telepon Selular" Telekontran, vol. 1, pp. 24-32, Jan. 2013.

[8] Tutorialspoint, "TensorFlow – CNN and RNN Difference," [Online]. Available: https://www.tutorialspoint.com/tensorflow/tensorflow_cnn_and_rnn_difference.htm. [Accessed: 23-Feb-2020].

[9] A. A. Aml and S. M. Mohamed, "Applying Voice Recognition Technology for Smart Home Networks," *2016 International Conference on Engineering & MIS (ICEMIS)*, pp. 1-6, Sept. 2016.

[10] L. Yanan, Z. Jie, Y. Chengfei, Z. Xiaosong, L. Tao and B. Gang, "HIS-CNN: A Novel Convolution Neural Network for Hyperspectral Image," *2018 International Converence on Audio, Language and Image Processing (ICALIP)*, pp. 464-469, Jul. 2018.

[11] MatWorks, "Speech Command Recognition Using Deep Learning," [Online]. Available: https://www.mathworks.com/help/deeplearning/examples/deep-learning-speech-recognition.html. [Accessed: 08-July-2019].