Konsentrasi : Sistem Cerdas

PENERAPAN SPEECH RECOGNITION MENGGUNAKAN METODE LONG SHORT-TERM MEMORY (LSTM) UNTUK PRESENTASI DINAMIS

LAPORAN TUGAS AKHIR

SATRIYA ADHITAMA

5200411545

PROGRAM STUDI INFORMATIKA
FAKULTAS SAINS & TEKNOLOGI
UNIVERSITAS TEKNOLOGI YOGYAKARTA
YOGYAKARTA

2023

PENERAPAN SPEECH RECOGNITION MENGGUNAKAN METODE LONG SHORT-TERM MEMORY (LSTM) UNTUK PRESENTASI DINAMIS

Disusun oleh

SATRIYA ADHITAMA

5200411545

Telah dipertahankan di depan Dewan Penguji

pada tanggal .....

DEWAN PENGUJI

Nama & Gelar	Jabatan	Tanda tangan	Tanggal
Nama NIK	Ketua Penguji	.....	.....
Nama NIK.....	Penguji I	.....	.....
Nama NIK	Penguji II (Dosen Pembimbing)	.....	.....

Yogyakarta, ...
Ketua Program Studi Informatika

Nama
NIK 110909046

LEMBAR PERNYATAAN

Yang bertanda tangan di bawah ini, saya

Nama : Satriya Adhitama

NPM : 5200411545

Program Studi : Informatika

Program : Sarjana

Fakultas : Sains & Teknologi

menyatakan bahwa tugas akhir dengan judul ***"Penerapan Speech Recognition Menggunakan Metode Long Short-Term Memory (LSTM) untuk Presentasi Dinamis"*** ini adalah karya ilmiah asli saya dan belum pernah dipublikasikan oleh orang lain, kecuali yang tertulis sebagai acuan dalam naskah ini dan disebutkan dalam daftar pustaka. Apabila di kemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa yang diberikan Program Studi Informatika Fakultas Teknologi Informasi dan Elektro Universitas Teknologi Yogyakarta.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta
Pada tanggal : 12 Juni 2023

Yang menyatakan

Satriya Adhitama

KATA PENGANTAR

Puji syukur dipanjatkan atas kehadirat Allah SWT, karena dengan limpahan karunia-Nya penulis dapat menyelesaikan Tugas Akhir dengan judul ***"Penerapan Speech Recognition Menggunakan Metode Long Short-Term Memory (LSTM) untuk Presentasi Dinamis".***

Penyusunan Tugas Akhir diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana pada Program Studi Informatika Fakultas Sains & Teknologi Universitas Teknologi Yogyakarta.

Tugas Akhir ini dapat diselesaikan tidak lepas dari segala bantuan, bimbingan, dorongan dan doa dari berbagai pihak, yang pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

a. Bapak Bambang Moertono S., Dr., MM.,Akt.,CA. selaku Rektor Universitas Teknologi Yogyakarta
b. Bapak Sutarman, M. Kom., Ph.D. selaku Dekan Fakultas Sains & Teknologi
c. Ibu Enny Itje Sela, Dr.,S.Si.,M.Kom selaku Ketua Program Studi Informatika
d. Bapak Donny Avianto, S.T.,M.T. selaku Dosen Pembimbing Proyek Informatika

Akhir kata, penulis menyadari bahwa sepenuhnya akan terbatasnya pengetahuan penyusun, sehingga tidak menutup kemungkinan jika ada kesalahan serta kekurangan dalam penyusunan Tugas Akhir, untuk itu sumbang saran dari pembaca sangat diharapkan sebagai bahan pelajaran berharga dimasa yang akan datang.

Yogyakarta, 12 Juni 2023

Satriya Adhitama

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang semakin maju menjadikan berbagai aktivitas untuk lebih mudah dan efektif. Banyak pekerjaan ringan hingga berat dapat disederhanakan dari segi proses persiapan sampai penggunaan. Teknologi ini menjawab berbagai kebutuhan untuk segala aspek dan bidang kehidupan. Salah satu bidang yang mendapatkan manfaat besar dari kehadiran teknologi adalah komunikasi.

Presentasi merupakan metode komunikasi dengan menyampaikan informasi, ide, dan gagasan oleh seseorang kepada sekelompok orang/audiens yang banyak. Power point menjadi salah satu media yang dapat mendukung pembicara untuk dapat menyampaikan informasi, ide, atau gagasan yang diinginkan. Power point dapat berupa informasi visual dan tulisan yang berupa inti dari ide atau gagasan yang akan disampaikan. Tulisan-tulisan dan gambar ini akan ditampilkan pada layar untuk dapat dilihat oleh audiens secara jelas. Kemudian, pembicara akan memaparkan apa yang menjadi penjelasan dari tulisan dan/atau gambar yang ditampilkan.

Power point dapat disusun secara dinamis. Dinamis yang dimaksud adalah pembicara dapat melakukan interaksi dengan leluasa terhadap bagian-bagian tertentu yang terdapat dalam *power point* yang disusun. Pembicara dapat memajukan atau memundurkan slide presentasi, menuju ke bagian tertentu, menjalankan audio atau video, menampilkan bagian yang disembunyikan, dan lain sebagainya. Pembicara akan berinteraksi secara dinamis terhadap *power point* agar informasi yang ingin disampaikan dapat tepat sasaran dan audiens lebih tertarik untuk mendengar pembicara dan melihat layar presentasi.

Namun, terkadang materi yang disusun secara dinamis pada *software* pendukung presentasi akan mengakibatkan pembicara kesusahan untuk berinteraksi. Pembicara harus selalu mengarahkan pointer ke bagian yang dinginkan atau membutuhkan asisten untuk dapat membantu melakukan interaksi

dengan *software* presentasi. Pembicara memerlukan alat tambahan untuk dapat mengoperasikan *software* presentasi seperti *keyboard, mouse,* atau *remote control.*

Hal ini menimbulkan distraksi bagi pembicara karena harus berurusan dengan hal teknis presentasi daripada fokus dengan informasi apa yang harus disampaikan. Teknis presentasi untuk memilih konten yang ingin ditampilkan selama presentasi berlangsung akan menjadi gangguan tersendiri bagi *presenter*. Perlu adanya teknologi yang bisa membantu pembicara untuk melakukan presentasi secara dinamis dengan mudah dan minim distraksi teknis. Maka dari itu, disusunlah penelitian ini untuk merancang sistem yang mampu merekognisi suara kemudian mengubahnya menjadi perintah (*command*) guna mengoperasikan perangkat lunak pembantu presentasi.

Implementasi pengenalan ucapan (*speech recognition*) dilakuakan menggunakan model *machine learning sequential Long Short-Term Memory* (LSTM). LSTM yang merupakan turunan dari RNN telah mencapai peningkatan performa untuk *Automatic Speech Recognition* (Oruh dkk., 2022). LSTM dapat menangkap informasi *time series* yang cocok digunakan untuk pengenalan ucapan (*speech recognition*) (Xiang dkk., 2019). LSTM akan dapat mengenali kata tertentu yang muncul pada sinyal suara (*keyword spotting*). Dengan demikian, LSTM menjadi pilihan yang tepat untuk diimplementasikan pada sistem pengenalan ucapan *software* presentasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, penelitian ini memiliki rumusan masalah yaitu distraksi pengoperasian *software* presentasi bagi pembicara. *Software* presentasi hanya dapat dioperasikan menggunakan perangkat pendukung lain seperti *keyboard, mouse,* atau *remote control*. Hal ini tentu akan mengganggu jalannya presentasi pembicara karena diperlukannya aktivitas tambahan selain harus fokus terhadap materi yang akan dikomunikasikan. Dengan demikian, diperlukan sistem *speech-to-command* guna membantu memudahkan teknis presentasi.

1.3 Batasan Masalah

Penelitian “Penerapan Speech Recognition Menggunakan Metode *Long Short-Term Memory* (LSTM) untuk Presentasi Dinamis”, yang mencakup berbagai hal, sebagai berikut:

a. Perancangan sistem untuk dapat melakukan kontrol presentasi *power point* desktop dengan perintah suara;
b. Perancangan dilakukan dalam bahasa pemrograman python;
c. Speech recognition diterapkan menggunakan metode *long short-term memory* (LSTM)

1.4 Tujuan penelitian

Penelitian ini bertujuan untuk mengimplementasi dan menganalisis sistem speech recognition menggunakan metode *Long Short-Term Memory* (LSTM) yang dapat digunakan sebagai penunjang bagi pembicara untuk dapat melakukan presentasi secara dinamis hanya dengan perintah suara.

1.5 Manfaat Penelitian

Implementasi *speech recognition* untuk presentasi dapat memudahkan *presenter*/pembicara untuk dapat mengoperasikan *software* presentasi tanpa menggunakan perangkat tambahan pembantu lain seperti *mouse*, *keyboard*, atau *remote control*. Dengan demikian, distraksi yang muncul dari penggunaan perangkat tambahan pembantu ini dapat dikurangi, sehingga pembicara dapat lebih fokus untuk menyampaikan materi presentasi dengan baik ke audiens.

1.6 Sistematika Penulisan

Tugas akhir ini disusun dengan sistematika yang merupakan kerangka dan pedoman penulisan untuk memudahkan mengetahui dan memahami pembahasan yang terkandung secara menyeluruh. Berikut adalah sistematika penulisan yang digunakan:

Tabel 1. 1 Sistematika Penulisan

BAB I	**PENDAHULUAN** Bab I berisi penjelasan mengenai masalah yang melatarbalakangi mengapa penelitian dilakukan. Pada bab ini disajikan rincian tentang masalah dari umum ke khusus. Adapun susunan dari bab ini terdiri atas latar belakang, rumusan masalah, manfaat penelitian, dan sistematika penulisan
BAB II	**KAJIAN HASIL PENELITIAN DAN LANDASAN TEORI** Bab II meliputi penelitian-penelitian terdahulu yang relevan terhadap pembahasan yang akan disajikan pada penelitian sekarang, yaitu berkaitan tentang *speech recognition* dan metode LSTM. Kemudian, dasar-dasar teori yang dibutuhkan pada penelitian ini akan dijelaskan secara rinci, seperti *automatic speech recognition* (ASR), *Long Short-Term Memory* (LSTM).
BAB III	**METODE PENELITIAN** Bab III menjelaskan tentang metode penelitian yang dilakukan penulis dalam pengembangan rancangan implementasi sistem yang meliputi desain penelitian, penentuan objek penelitian, fokus penelitian, sumber data, teknik pengumpulan data, dan teknik analisis data.
BAB IV	**ANALISIS DAN PERANCANGAN SISTEM** Bab IV terdiri atas analisis sistem tanpa implementasi yang dibandingkan dengan sistem yang diusulkan dengan termuat analisisnya secara fungsional dan non-fungsional. Dari sistem yang telah dianalisis, perancangan dilakukan untuk mendapatkan desain *logic* dan fisikdari sistem yang akan dibangun.
BAB V	**IMPLEMENTASI DAN HASIL SERTA PEMBAHASAN** Bab V terdiri dari implementasi sistem dan analisis hasil penelitian yan tela dilakukan. Implementasi dari sistem pengenalan ucapan (*speech* recognition) menggunakan metode LSTM akan diujikan berdasarkan parameter yang telah ditentukan sehingga dapat diketahui bangaimana tingkat keberhasilan penelitian untuk dapat menjawab masalah yang ada.

BAB VI	**PENUTUP** Bab VI berisi kesimpulan dan saran dari penelitian yang telah dilakukan. Kesimpulan menjelaskan penyelesaian dari rumusan masalah yang dilakukan secara objektif. Saran berisi jalan keluar untuk mengatasi segala masalah yang ada pada penelitian.

BAB II
KAJIAN HASIL PENELITIAN DAN LANDASAN TEORI

2.1 Kajian Hasil Penelitian

Beberapa penelitian berikut merupakan hasil dari peneliti terdahulu yang memiliki bidang dan tema yang sama dengan penelitian yang akan dilakukan. Kajian penelitian ini diperlukan untuk dijadikan sebagai dasar alasan mengapa metode *long short-term memory* (LSTM) dapat diterapkan dengan baik untuk masalah pengenalan ucapan (*speech recognition*).

Pertama, penelitian oleh Cahuantzi, dkk (2021) membandingkan antara 2 metode untuk *learning symbolic sequences*, yaitu *long short-term memory* (LSTM) dan *gated recurrent unit* (GRU). Kompleksitas *seed strings* digunakan sebagai pembanding kedua metode tersebut untuk dapat mengetahui metode mana yang memiliki performa atau waktu *learning* yang terbaik. Kedua metode akan diuji pada kompleksitas *seed strings* yang berbeda. Pada kompleksitas rendah dilakukan pelatihan pada 3.600 RNN dengan total 37 *seed strings* berbeda, minimal 1.100 panjang karakter, kompleksitas LZW antara 2 hingga 12, jumlah unit pada *hidden layer* mencapai 25 hingga 250, serta 0,001 dan 0,01 untuk learning rate. Proses *training* menghasilkan median waktu latih 37,19 detik untuk LSTM dan 19,72 detik untuk GRU. Pada kompleksitas tinggi, digunakan 300 *seed strings* dengan 10, 33, atau 52 simbol, komleksitas LZW di antara 1.000 hingga 1.850, panjang karakter mencapai 2.400, dan total 4.500 RNN digunakan untuk *training*. Proses *training* untuk kompleksitas tinggi menghasilkan median waktu latih 12,53 untuk LSTM dan 22,84 detik untuk GTU. Hasil pengujian menunjukkan bahwa LSTM jauh mengungguli GRU dalam waktu latih untuk kompleksitas *sequences* yang tinggi. Peneliti juga menunjukkan bahwa LSTM dan GRU mencapai akurasi yang hampir sama berdasarkan perbandingan kompleksitas LZW dengan metrik jarak Damerau-Lavenshtein (DL) dan Jaro-Winkler (JW), namun LSTM dapat melatih lebih cepat dibanding GRU. Hal ini membuktikan bahwa LSTM dapat digunakan dengan baik untuk data yang

memiliki kompleksitas *sequence* tinggi dengan mencapai perform metrics a latih yang cepat dan akurasi prediksi yang tidak jauh beda dengan metode pembandingnya (GRU).

Kedua, penelitian oleh Oruh dkk (2022) adalah tentang pengembangan model *hybrid long short-term memory-recurrent neural network* (LSTM-RNN) untuk *automatic speech recognition* (ASR). LSTM-RNN merupakan model gabungan dengan mengintegrasikan RNN sebagai *forget gate* pada arsitektur LSTM. Modifikasi ini menyebabkan perubahan pada peningkatan sumber komputasi. Penulis membandingkan performa LSTM-RNN terhadap model *deep learning* dan *sequential* lain. Pengujian dilakukan pada 2.400 file suara berbeda dengan format wav. File suara ini terdiri atas 15 penutur (pria & wanita) dengan masing-masing mengucapkan angka 0-9 sebanyak 16 kali. Penulis menetapkan sebanyak 90% (2.160 pengucapan) untuk data latih dan 10% (240 pengucapan) untuk data uji. Perbandingan LSTM-RNN dengan model *deep learning* lain menunjukkan bahwa model ini memiliki akurasi yang tinggi. LSTM-RNN menghasilkan 99,36%, 72,83% untuk ResNet-18, 74,17% untuk ResNet-34, 89,67% untuk DenseNet-121, 87,17% untuk DenseNet-169, dan 77,17% untuk VGG-16. Perbandingan LSTM-RNN dengan *sequential model* lain juga menunjukkan bahwa LSTM-RNN memiliki tingkat akurasi yang lebih tinggi. LSTM-RNN menghasilkan 99,36%, 87,11% untuk *simple LSTM*, 80,86% untuk *bidirectional LSTM*, 83,59% untuk *simple RNN*, dan 90,23% untuk GRU. Hasil penelitian tersebut membuktikan bahwa LSTM dengan model *hybrid* maupun *non-hybrid* memiliki performa yang baik untuk penerapan *automatic speech recognition* (ASR). Pada model *sequential* lain yang dibandingkan, variasi dari LSTM juga menunjukkan akurasi yang cukup baik, yaitu lebih dari 80%.

Ketiga, penelitian oleh Atcheson dkk (2019) menggunakan proses Gaussian pada LSTM dan MLP untuk memprediksi *continuous-time* dan dimensi emosi pada ucapan ambigu. Penulis mengimplementasikan metode LSTM dan MLP dengan atau tanpa proses Gaussian untuk kasus *contiuous emotion recognition* (CER), sehingga dapat mengetahui nada emosi pada ucapan. Dengan demikian, dapat melengkapi pekerjaan *automatic speech recognition* (ASR),

dengan mencoba melihat konten linguistik pada ucapan. Penelitian dilakukan dengan mengukur *condordance correlation coefficient (*CCC) antara rangkaian waktu yang terprediksi dan mean dari anotasi target. Selain itu juga dihitung *som of the log probabilities* (SLP), dengan semakin tinggi nilai maka semakin akurat prediksinya. Eksperimen yang telah dilakukan menujukkan bahwa *long short-term memory-gaussian process* (LSTM-GP) memiliki hasil prediksi terbaik dengan nilai CCC 0,78 (network 4-4-4) dan SLP -119,2 (network 2-2). Pelatihan dengan LSTM menghasilkan nilai CCC 0,71 (network 8-8) dan SLP –344,5 (network 4-4-4). Kedua variasi LSTM ini dibandingkan dengan *multi-layer perceptron* (MLP) yang menghasilkan nilai 0,63 CCC dan -356,4 SLP dengan proses gaussian, serta nilai 0,43 CCC dan -530,9 dengan tanpa proses gaussian. Penelitian yang telah dilakukan menunjukkan bahwa LSTM (tanpa atau dengan gaussian) memiliki performa yang mengungguli MLP dalam kasus CER. Hasil ini membuktikan bahwa LSTM dapat dengan baik digunakan dalam mendukung pengembangan *learning model* untuk ASR.

Keempat, penelitian oleh Arief dkk (2021) pada klasifikasi audio ucapan untuk mengetahui kondisi emosional seseorang berdasarkan intonasi bicara. Pembuatan model *machine learning* untuk klasifikasi ini dilakukan menggunakan algoritma *Long Short-Term Memory* (LSTM) untuk merekognisi 7 macam emosi, yaitu *angry* (marah), *disgust* (jijik), *fear* (takut), *happpy* (senang), *neutral* (netral), *sad* (sedih), dan *surprise* (terkejut) yang masing-masing terdiri atas 200 audio file. Dari penelitian tersebut, model LSTM yang dilatih dengan jumlah 100 *epoch* dan ukuran *batch* 64, sehingga dapat menghasilkan hasil akurasi data latih sebesar 98,47% dan data uji 97,02%, serta nilai ROC curve sebesar 99,9%. Hasil prediksi menunjukkan dari total 840 audio file, hanya 25 yang tidak diprediksi dengan benar. Penelitian tersebut menunjukkan bahwa LSTM dapat digunakan dengan baik untuk kasus klasifikasi ucapan yang mendalkan pola intonasi dan pembentukan pada kata ucapan.

Beberapa hasil penelitian di atas digunakan sebagai dasar untuk pemilhan metode LSTM yang akan diimplementasikan pada kasus *speech recognition*. Hasil penelitian yang telah dipaparkan menunjukkan bahwa LSTM dengan

berbagai variasi modelnya memiliki performa dan tingkat akurasi yang baik untuk digunakan pada data *sequential* yang kompleks. Dengan demikian, metode ini dapat dengan tepat digunakan untuk diterapkan untuk pengembangan sistem pengenalan ucapan (*speech recognition*).

Tabel 2. 1 Penelitian Terdahulu

No	Judul	Penulis	Metode	Hasil/ Kesimpulan
1	*A Comparison of LSTM and GRU networks for learning symbolic sequences*	Roberto Cahuantzi, Xinye Chen, dan Steffan Güttel	LSTM dan GRU	LSTM memiliki performa yang jauh mengungguli GRU untuk kompleksitas data yang tinggi dengan perbandingan hasil waktu latih 12,53 detik dan 22,84 detik. Hasil juga menunjukkan prediksi akurasi dengan metrik jarak DL dan JW yang dihasilkan LSTM tidak jauh beda dengan GRU untuk semua kasus.
2	*Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition*	Jane Oruh, Serestina Viriri, dan Adekanmi Adegun	LSTM-RNN	Perbandingan hasil dilakukan untuk beberapa model *deep learning* dan *sequential model* terhadap LSTM-RNN. Akurasi LSTM-RNN jauh unggul dibanding model *deep learning* dan *sequential model* lain mencapai 99,36%.
3	*Using Gaussian Processes with LSTM neural Network to predict continuous-time,dimensional emotion in ambiguous speech*	Mia Actheson, Julien Epps, dan Vidhayasaharan	LSTM, gaussian LSTM, MLP, gaussian MLP	Penelitian ini membandingkan 2 metode neural network, yaitu MLP dan LSTM dengan perbedaan penggunaan proses gaussian (GP) terhadap masing-masing metode. Performa model tersebut menghasilkan *condordance correlation coefficient* (CCC) mencapai 0,78 untuk LSTM-GP, 0,71 untuk LSTM, 0,63 untuk MLP-GP, dan 0,43 untuk MLP
4	Klasifikasi Audio Ucapan Emosional Menggunakan Model LSTM	Raynaldy Arief, Nur Aviva Iriawan, dan Armin Lawi	LSTM	Model LSTM dilatih dengan jumlah *epoch* 100 dan ukuran *batch* 64 yang menghasikan akurasi 98,47% untuk data latih dan 97,02% untuk data uji, serta 99,9% nilai kurva ROC

5	Penerapan Speech Recognition Menggunakan Metode *Long Short-Term Memory* (LSTM) untuk Presentasi Dinamis	Satriya Adhitama	LSTM	

2.2 Landasan Teori

2.2.1 Presentasi

Presentasi merupakan adalah aktivitas mengungkapkan pikiran, gagasan, ide, pendapat, argumen, dan yang lainnya menggunakan bahasa lisan(Lisnawati & Ertinawati, 2019). Presentasi dilakukan dengan menyampaikan gagasan yang disusun secara terstruktur pada media presentasi seperti *powerpoint*, *google slide*, *canva*, dan sebagainya. Media pendukung tersebut mampu menarik audiens untuk dapat memperhatikan dan mendengarkan isi presentasi.

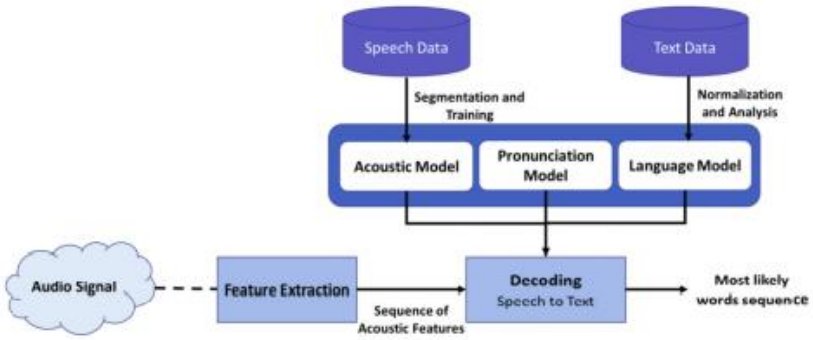
2.2.2 Automatic Speech Recognition

Automatic speech recognition (ASR) merupakan salah satu tugasn dari *natural language processing* (NLP) yang melakukan transkripsi komputasi *real-time* dari bahasa yang diucapkan. ASR menjadi studi studi *human-computer interface* sejak 1950 dan berkembang pesat seiring waktu. Perkembangan dari ASR memberikan kemampuan untuk sistem dapat mengenali ucapan dengan kondisi yang beragam seperti adanya *noise* (kebisingan), aksen, diksi, dan nada (Kamath dkk., 2019).

ASR adalah metode untuk dapat mengubah ucapan kata menjadi teks dengan menggunakan komputer yang berbasis perangkat lunak. Sistem dirancang dengan teknik tertentu yang bertujuan untuk mengenali dan memproses suara manusia (Sen dkk., 2019). Prinsip dasar dari ASR yaitu seseorang berbicara mengeluarkan variasi tekanan suara pada *larynx* (pangkal tenggorokan), kemudian suara yang dihasilkan akan digitalisasi menggunakan *microphone* dan dikirimkan melalui sebuah perantara atau jaringan (Fendji dkk., 2022).

Mesin pengenal suara menerima *input* suara dalam bentuk yang telah didigitalisasi. Kemudian, *input* ini akan diubah menjadi *acoustic units* (*acoustic*

vectors) melalui *acoustic model* (AM). Mesin pengenal menganalisis urutan (*sequence*) dari *acoustic vectors* dengan membandingkan kesesuaiannya pada data dan *acoustic model* untuk dapat dicari kandidat urutan yang paling mendekati. Sistem ASR mempu mengubah *input* suara dalam bentuk model matematis yang kemudian dapat dibuat menjadi teks untuk dapat mengenali ucapan.



Gambar 2. 1 Arsitektur ASR

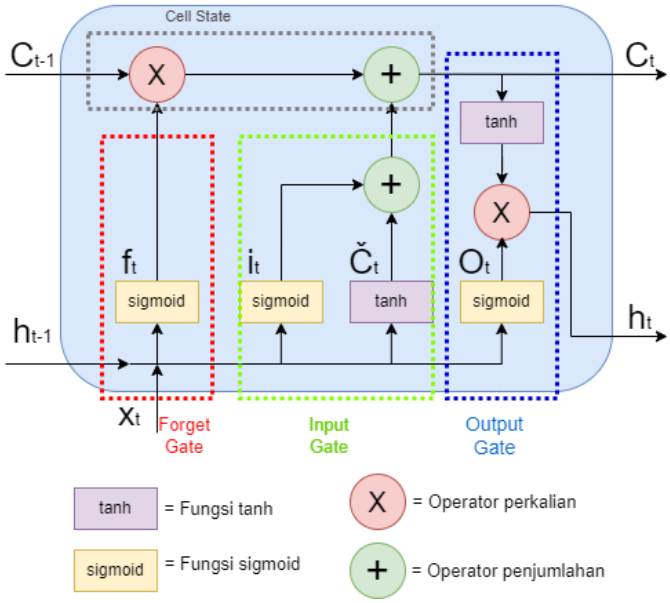
Gambar di atas merupakan arsitektur dari *automatic speech recognition* (ASR). Secara umum, sistem ASR memiliki arsitektur yang sama, meskipun kosakata yang digunakan terbatas, menengah, banyak, atau sangat banyak (Fendji dkk., 2022). ASR memiliki 5 (lima) komponen utama, yaitu *feature extraction*, *acoustic model* (model suara), *language model* (model bahasa), *pronounciation model* (model pengucapan), dan *decoder*.

2.2.3 Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) merupakan pengembangan metode dari *recurrent neural network* (RNN) yang dirancang dengan *memory cell* yang mampu merepresentasikan dependensi jangka panjang (*long-term dependency*) terhadap urutan waktu yang terjadi pada data (Sagheer & Kotb, 2019). Penanganan *vanishing gradient problem* pada RNN dapat diatasi menggunakan LSTM. LSTM secara khusus dirancang untuk menghindari masalah dependensi jangka panjang.

LSTM memiliki tiga gates atau gerbang yang masing-masing memiliki peran untuk melindungi dan mengontrol cell state. Cell state merupakan garis horizontal yang melewati bagian atas diagram sel LSTM yang memiliki kemampuan untuk

menghapus atau menambahkan informasi baru yang masuk dalam waktu t dengan memanfaatkan struktur cermat yang disebut gerbang. Gates atau gerbang sendiri adalah sebuah cara yang digunakan oleh LSTM untuk melakukan seleksi terhadap informasi yang 10 masuk ke dalam sel



Gambar 2. 2 Arsitektur LSTM

Forget gate merupakan gerbang pertama yang dioperasikan dalam sel LSTM. Forget gate ini menentukan informasi mana yang harus dipertahankan dan yang harus dibuang dari cell state. Gerbang ini menerima dua input, masing-masing dari ht-1 dan xt yang dimana ht-1 merupakan keluaran dari proses LSTM pada time step t-1 dan xt merupakan input pada time step t. Output dari gerbang ini adalah angka dengan rentang 0 hingga 1.

$$f_t = \sigma(w_f \,.\, [h_{t-1}, x_t] + b_f)$$

Setelah didapatkan nilai dengan rentang 0 hingga 1 pada forget gate, maka operasi selanjutnya adalah pada input gate. Input gate terdiri dari dua bagian, bagian pertama menggunakan fungsi sigmoid yang menentukan informasi mana yang ingin di-update. Sedangkan bagian kedua menggunakan fungsi tanh yang berfungsi untuk menentukan vektor yang akan ditambahkan pada nilai cell state (C't). Kedua bagian tersebut berfungsi untuk menentukan informasi baru apa saja yang ingin disimpan pada cell state.

$$i_t = \sigma(w_i \,.\, [h_{t-1}, x_t] + b_i)$$

$$C'_t = tanh(w_C \,.\, [h_{t-1}, x_t] + b_C)$$

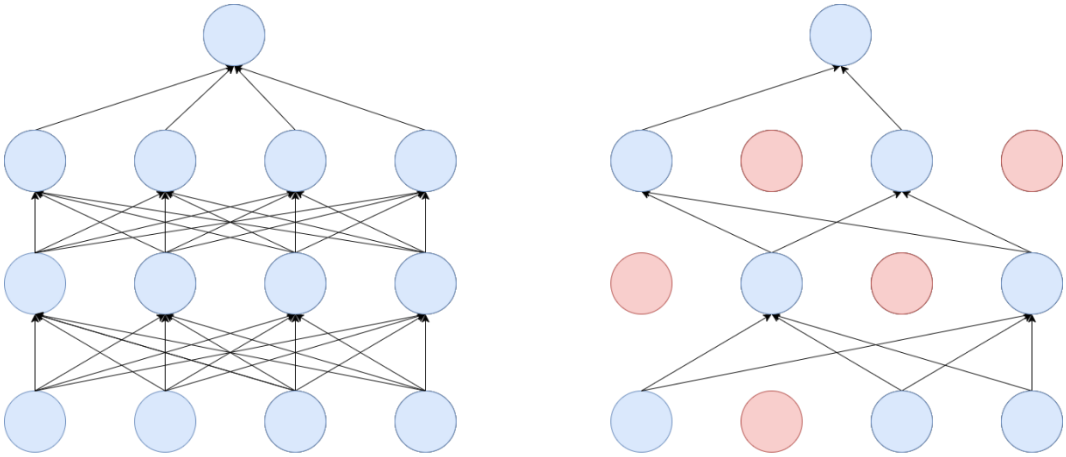
Hasil yang didapatkan dari operasi pada forget gate dan input gate selanjutnya dioperasikan agar hasil dari operasi tersebut dapat digunakan untuk meng-update cell state Ct-1 .Pada operasi ini dilakukan perkalian pada state sel Ct-1 dengan hasil dari layer forget gate (ft). Lalu tambahkan hasil perkalian tersebut dengan it * Ct. Setelah dua operasi tersebut (operasi perkalian Ct-1 dan ft lalu penambahan dengan it * C't), langkah selanjutnya adalah dengan mengoperasikan output gate. Pada output gate, layer sigmoid akan dioperasikan terlebih dahulu untuk menentukan bagian dari cell state apa saja yang akan digunkan sebagai hasil keluaran. Setelah operasi sigmoid dilakukan, selanjutnya operasi tanh dijalankan pada cell state lalu dikalikan dengan keluaran dari layer sigmoid yang telah dihitung sebelumnya.

$$i_t = f_t \ * \ C_{t-1} + \ C'_t$$
$$o_t = \sigma(w_o \,.\, [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \ * \ tanh(C_t)$$

2.2.4 *Dropout Regularization*

Regularisasi adalah metode sebuah proses untuk mengatasi permasalahan *overfitting* pada model yang dapat memengaruhi performa model. Kosep dari *fully connected layer* menimbulkan masalah eksponensial memori yang disebut *overfitting* sebagai akibat dari koneksi neuron yang terlalu banyak dan berlebih sehingga pembelajaran yang dilakukan terlalu berlebihan (Jabir & Falih, 2021).

Regularisasi dropout merupakan teknik untuk secara acak menjadikan bagian input unit menjadi 0 pada setiap langkah training. Beberapa neuron akan dinonaktifkan untuk mengurangi interdependensi neuron dan terlalu bergantung kepada fitur spesifik.



(a) Sebelum regulasisasi *dropout* (b) Sesudah regulasisasi *dropout*

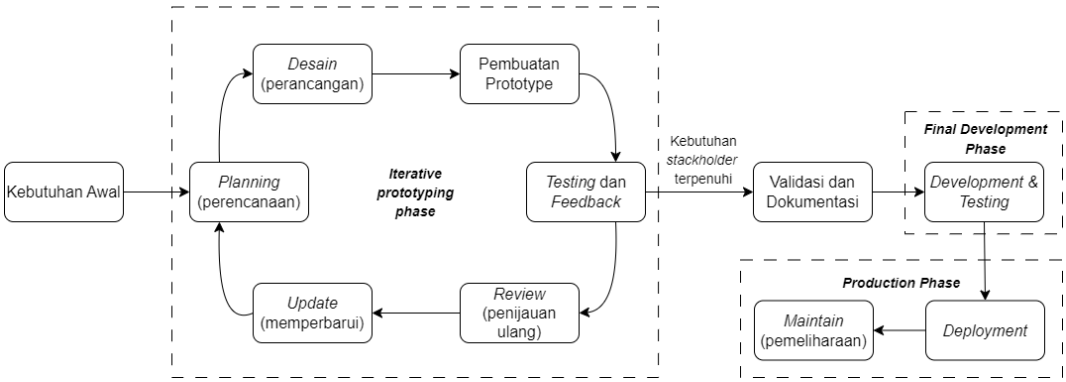
Gambar 2. 3 Regularisasi dropout

2.2.5 Metode Pengembangan Sistem dengan *Prototype*

Model prototyping merupakan metode pengembangan sistem dengan pembuatan *prototype* (purwarupa) secara iteratif hingga kebutuhan *stackholder* terpenuhi.. Metode ini mengumpulkan kebutuhan informasi *stackholder* secara cepat dan berfokus pada penyajian aspek perangkat lunak yang akan nampak, sehingga *stackholder* dapat mengevaluasi model *prototype* untuk menyaring kebutuhan pengembangan sistem (Pricillia & Zulfachmi, 2021).

Prototype merupakan alat yang memberikan ide bagi pengembang dan pengguna tentang bagaimana sistem berfungsi dalam bentuk lengkapnya. *Prototype* menjadi sebuah model atau representasi dari sistem yang dikembangkan guna memperoleh pemahaman mengenai fitur, desain, dan fungsionalitas sebelum dilakukan pengembangan penuh. Proses untuk menghasilkan sebuah *prototype* (purwarupa) disebut dengan *prototyping*.

Gambar 2. 4 di bawah merupakan bagan metode pengembangan sistem menggunakan *prototype*:



Gambar 2. 4 Model *prototype*

Gambar 2. 4 menunjukkan tahapan-tahapan pengembangan sistem menggunakan iterasi *prototyping*. Informasi kebutuhan awal sistem ditentukan dengan melibatkan *stackholder* sehingga pengembangan dapat dilakukan secara tepat dan jelas, sehingga dapat meminimalkan proses iterasi pembuatan *prototype*. Fase pembuatan *prototype* dilakukan secara iteratif dengan pengembang akan melakukan tahapan desain dan pembuatan purwarupa secara cepat sehingga dapat memperoleh *feedback stackholder* segera mungkin. Hasil pengujian dan *feedback* dari *stackholder* digunakan oleh pengembang sebagai bahan untuk keterbaharuan dan evaluasi model *prototype* yang telah dikembangkan pada iterasi sebelumnya.

Setelah kebutuhan *stackholder* terpenuhi,model akhir *prototype* akan divalidasi dengan kebutuhan awal yan telah ditentukan oleh *stackholder*. Dokumentasi digunakan untuk mencatat segala revisi kebutuhan dan perancangan selama proses pengembangan berlangsung. Fase akhir pengembangan dilakukan untuk menyempurnakan sistem dan menguji kelayakannya sebelum di-*deploy*. *Deployment* merupakan peralihan dari *environment* (lingkungan) dari *developmment* (pengembangan) menuju produksi yang bertujuan untuk mengimplementasikan sistem sehingga dapat digunakan oleh pengguna akhir.

2.2.6 *Sparse Categorical Crossentropy*

Sparse categorical crossentropy adalah fungsi *loss* yang umum digunakan untuk kasus klasifikasi multi kelas ketika label target merupakan bilangan bulat (*integer*). Fungsi *loss* ini berasal dari *categorical crossentropy* namun label direpresentasikan sebagai matriks dengan format *sparse*. Matriks *sparse*

merepresentasikan label sebagai sebuah nilai index tunggal daripada vektor *one-hot encoding*.

Sebagai perbandingan, Tabel 2. 2 dan Tabel 2. 3 menunjukkan contoh data untuk *sparse categorical crossentropy* dan *categorical crossentropy*:

Tabel 2. 2 Data dengan label integer

Nama	X1	X2	X3	Y (label)
A	0,5	0,5	0,7	0
B	0,7	0,2	0,2	1
C	0,8	0,9	0,2	2
D	0,1	1	0,9	3

Tabel 2. 3 Data dengan label *one-hot encoding*

Nama	X1	X2	X3	Y=0	Y=1	Y=2	Y=3
A	0,5	0,5	0,7	1	0	0	0
B	0,7	0,2	0,2	0	1	0	0
C	0,8	0,9	0,2	0	0	1	0
D	0,1	1	0,9	0	0	0	1

Tabel 2. 2 menunjukkan contoh data dengan label target dengan bentuk bilangan bulat. Nilai target yang direpresentasikan sebagai index sesuai dengan jumlah kelas, sehingga perhitungan *loss* dapat menggunakan *sparse categorical crossentropy*. Tabel 2. 3 menunjukkan contoh data dengan label target dibuat terpisah dengan *one-hot encoding*. Nilai label dipisahkan ke kolom-kolom yang berbeda dengan nilai 0 sebagai *false* dan 1 sebagai *true* terhadap kolom target yang sesuai.

Sparse categorical crossentropy ini mengukur perbedaan antara distribusi probabilitas yang terprediksi dan distribusi sesungguhnya dari label target. Memori yang digunakan menjadi lebih efisien dengan pendekatan ini, khususnya untuk data dengan jumlah kelas yang besar, sehingga dapat menghindari untuk melakukan *one-hot encoding*. Dengan secara langsung menggunakan label bilangan bulat, ini akan mengurangi memori yang digunakan untuk menyimpan dan memproses label target.

$$SparseCategoricalCrossentropy = -\log(softmax(y)[t] = -\log(\frac{exp(y)}{\sum_{j=1}^{n} exp(y)}[t]$$

BAB III
METODE PENELITIAN

3.1 Bahan/Data

3.1.1 Data yang diperoleh

Sistem pengenalan ucapan yang akan dirancang membutuhkan data yang dapat digunakan sebagai perintah suara (*voice command*) untuk mengoperasikan *software* presentasi. Perintah dilakukan dengan mengucapkan kata-kata tertentu yang kemudian akan dikonversikan dalam bentuk teks, sehingga dapat dieksekusi sesuai dengan apa yang diucapkan pembicara. Ucapan dari pembicara akan dideteksi dengan model *machine learning* yang telah dibangun, kemudian hasilnya akan dicocokkan dengan daftar kata yang ada di dalam basis data.

Data yang diperoleh merupakan *dataset* yang berisi kumpulan *file* suara dengan format WAV. Masing-masing *file* memuat suara manusia untuk ucapan kata tertentu.. Kosa kata yang digunakan terbatas untuk memastikan bahwa proses menjadi ringan, namun tetap cukup beragam untuk pelatihan model pada data sehingga dapat berguna untuk berbagai pengaplikasian (Warden, 2018). Tujuan utama digunakannya dataset ini untuk menghasilkan model pelatihan mesin yang terbaik. Dataset memiliki 34 variasi kata berbahasa Inggris dengan jumlah 103.807 ucapan. Berikut adalah datanya:

Tabel 3. 1 Dataset *speech_commands_v0.02*

No.	Kata	Jumlah Ucapan
1	*Backward*	1.664
2	*Bed*	2.014
3	*Bird*	2.064
4	*Dog*	2.128
5	*Down*	3.917
6	*Eight*	3.787
7	*Five*	4.052
8	*Follow*	1.579
9	*Forward*	1.557

10	*Four*	3.728
11	*Go*	3.880
12	*Happy*	2.054
13	*House*	2.113
14	*Learn*	1.575
15	*Left*	3.801
16	*Marvin*	2.100
17	*Nine*	3.934
18	*No*	3.941
19	*Off*	3.754
20	*On*	3.845
21	*One*	3.890
22	*Right*	3.778
23	*Seven*	3.998
24	*Sheila*	2.022
25	*Six*	3.860
26	*Stop*	3.872
27	*Three*	3.727
28	*Tree*	1.759
29	*Two*	3.880
30	*Up*	3.723
31	*Visual*	1.592
32	*Wow*	2.123
33	*Yes*	4.044
34	*Zero*	4.052

Pada unduhan dataset termuat *file* teks yang dinamakan *validation_list.txt*, yang berisi kumpulan *file* untuk memvalidasi hasil selama pelatihan berlangsung, dan membantu menyesuaikan hyperparameters. *File testing_list.txt* memuat nama dari klip suara yang digunakan untuk mengukur model yang dilatih. Data suara dan teks tersbut akan digunakan lebih lanjut untuk pelatihan model *machine learning* menggunakan LSTM.

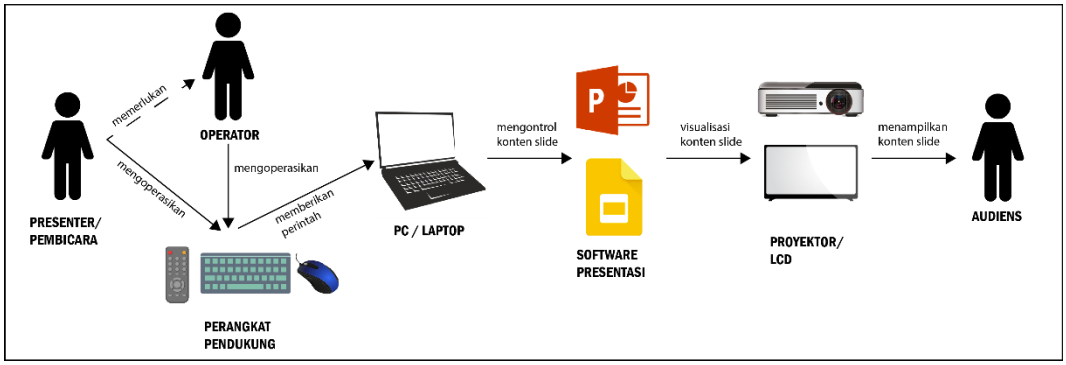
3.1.2 Prosedur pengumpulan data

Data yang dikumpulkan oleh Pete Warden ini dapat diunduh secara *online* karena sifatnya yang *open source*. Penulis mengunduh *speech_command_v0.02* pada https://www.kaggle.com/datasets/mok0na/speech-commands-v002. Data ucapan ini diambil melalui *microphone* telepon atau laptop yag dimiliki pengguna (Warden, 2018). Data yang dikumpulkan berupa ucapan berbahasa Inggris untuk membatasi cakupan pengumupulan data. Para subjek diminta untuk merekam suara pada ruangan tertutup sendirian dengan pintu tertutup untuk menghindari percakapan lain yang bersifat privasi. Subjek ini terdiri dari sejumlah orang yang berbeda-beda sehingga model dapat melakukan pelatihan dengan baik.

Pete juga mengumpulkan data dengan menghindari informasi yang dapat diidentifikasi dari kontributor dengan alasan privasi. Hal tersebut berarti *dataset* tidak termuat fitur seperti jenis kelamin, etnis, maupun ID pengguna. Data juga dibatasi untuk panjang durasi ucapan sekitar 1 (satu) detik. Data ucapan disimpan dalam satu *file* berbeda-beda, sesuai dengan kata yang telah ditentukan. Dengan demikian, tidak tersedia data yang berupa kalimat utuh agar lebih mudah untuk melakukan klasifikasi.

3.2 Aturan Bisnis

Penggunaan *software* presentasi cenderung dapat membantu kita dalam menyampaikan berbagai ide dan gagasan yang tersusun dengan visual yang menarik kepada audiens. Perangkat tambahan masih diperlukan sebagai perantara bagi *presenter* untuk mengubah konten yang ingin ditampilkan pada layar. Pilihan lain agar presenter tidak perlu menggunakan perangkat perantara ini yaitu dengan menentukan orang lain sebagai operator selama presentasi berlangsung.

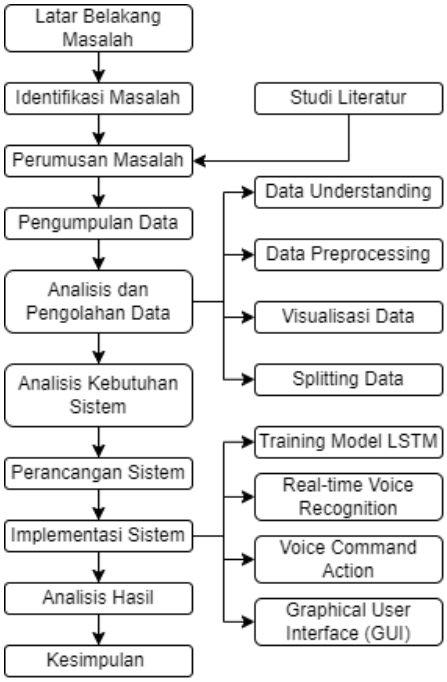


Gambar 2. 5 Aturan bisnis saat ini

Dengan adanya perantara operasional untuk *software* presentasi ini, tentu akan memberikan ketidakmudahan dan ketidakleluasaan *presenter* dalam menyampaikan materi yang telah disusun. Setiap beberapa saat, *presenter* diharuskan untuk secara langsung menggunakan perangkat pendukung (seperti *mouse, keyboard,* atau *pointer*). Tidak jarang juga perangkat seperti *keyboard* dan *mouse* diletakkan pada posisi yang berjauhan dengan *presenter*. *Pointer* yang harus terus dibawa menyebabkan *presenter* tidak dapat leluasa menggunakan gestur tangannya. Ketidaksinkronan seringkali terjadi antara *presenter* dengan operator yang mengubah halaman presentasi tidak sesuai dengan yang diinginkan.

3.3 Tahap Penelitian

Penelitian ini dilakukan dengan beberapa tahapan secara urut, yaitu menentukan latar belakang, identifikasi masalah, studi literatur, identifikasi dan perumusan masalah, analisis dan pengolahan data, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, analisis hasil, dan kesimpulan. Berikut merupakan bagan dari tahapan penelitian yang dilakukan:



Gambar 3. 1 Bagan Tahapan Penelitian

Setiap bagian tahapan penelitian di atas memiliki rincian sebagai berikut:

1. Latar Belakang Masalah

   Tahap pertama yang dilakukan adalah dengan mencari permasalahan beserta cakupannya untuk dapat ditentukan jenis penyelesaian yang dapat diterapkan. Latar belakang masalah menjadi landasan untuk memperkuat penelitian sehingga memiliki dampak dan kontribusi untuk masalah spesifik yang ditentukan.

2. Identifikasi Masalah

   Hasil penentuan masalah kemudian diidentifikasi sehingga memberikan kejelasan hal apa saja yang dapat berkaitan dengan permasalahan. Idenfikasi dapat dilakukan sehingga dapat ditentukan pendekatan penyelesaian masalah dengan metode yang sesuai.

3. Studi Literatur

   Studi literatur merupakan tahapan untuk mencari referensi berbentuk literatur tertulis hasil penelitian terdahulu. Literatur yang ditentukan memiliki relevansi untuk permasalahan yang ditentukan dalam hal metode penyelesaian dan

pengembangan, rancangan atau tahapan. Dengan adanya literatur ini, dapat menjadi perbandingan sehingga penelitian yang akan dilakukan dapat memberikan keterbaharuan maupun hasil yang lebih baik daripada hasil penelitian sebelumnya.

4. Perumusan Masalah

   Masalah yang telah ditentukan kemudian dirumuskan menjadi lebih rinci sehingga dapat ditentukan objek penelitian beserta subjek yang terkait di dalamnya. Rumusan masalah didukung dengan adanya studi literatur sebagai referensi. Tujuan perumusan masalah adalah untuk menggambarkan masalah secara objektif, menetapkan batasan-batasan masalah berikut parameter yang relevan, dan menentukan alur penyelesaian masalah.

5. Pengumpulan data

   Data-data yang mendukung untuk penyelesaian masalah dikumpulkan secara lengkap untuk kebutuhan lebih lanjut. Adapun data yang dikumpulkan adalah data suara yang berisi berbagai ucapan sebagai kata kunci untuk memberikan perintah pada *software* presentasi.

6. Analisis dan Pengolahan Data

   Data yang telah diperoleh kemudian dianalisis sehingga dapat diketahui nilai informasi yang ada di dalamnya sehingga mampu digunakan secara tepat untuk permasalahan yang ada. Data kemudian dikelola dengan bantuan bahasa pemrograman Python dengan tahapan yaitu *data understanding, data preprocessing, data visualization,* dan *data splitting* (membagi data).

7. Analisis Kebutuhan sistem

   Menentukan kebutuhan fungsional dan non-fungsional sistem.

8. Perancangan Sistem

   Pernacangan sistem merupakan proses untuk merencanakan dan mengembangkan struktur, komponen, dan interaksi antar elemen.

9. Implementasi sistem

   Rancangan sistem yang telah dibuat sebelumnya diimplementasikan menjadi suatu sistem yang dapat digunakan oleh pengguna. Adapun beberapa hal yang diimplementasikan dalam sistem, yaitu pelatihan model *machine learning*

LSTM, pembuatan sistem pengenalan ucapan secara *real-time*, perintah ucapan pada *software* presentasi, dan *Graphical User Interface* (GUI).

10. Analsis Hasil

    Model LSTM yang telah dieksperimenkan akan dianalisis untuk menemukan bagaimana agar dapat menghasilkan model yang terbaik untuk sistem pengenalan ucapan.

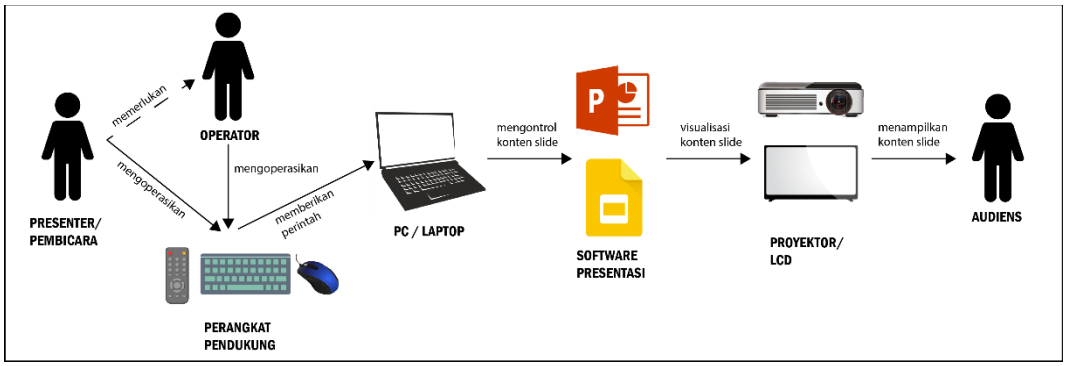
11. Kesimpulan

    Dari hasil pengujian ditentukan kesimpulan dari model yang dihasilkan dan sistem yang berjalan.

BAB IV
ANALISIS DAN PERANCANGAN SISTEM

4.1 Analisis Sistem

4.1.1 Analisis Sistem yang Berjalan

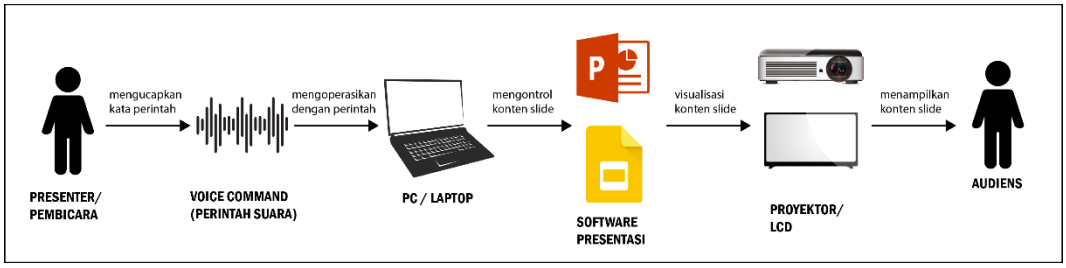


Gambar 4. 1 Arsitektur sistem saat ini

Gambar 4. 1 menunjukkan kondisi sistem semula. Alur tersebut menggambarkan proses untuk pembicara dapat melakukan presentasi menggunakan *software* pendukung hingga dapat ditampilkan ke audiens. Pada sistem ini, pembicara memerlukan bantuan operator atau perangkat keras pendukung seperti *remote control*, *mouse*, atau *keyboard* untuk dapat memberikan perintah pada mesin (komputer). Perintah ini kemudian akan menjadi *input* bagi komputer untuk dapat mengontrol konten *slide* presentasi yang ingin ditampilkan.

Kelemahan dari sistem ini adalah pembicara perlu secara langsung mengatur teknis pengubahan *slide* presentasi dengan bantuan operator atau perangkat pendukung. Hal ini akan menjadi distraksi bagi pembicara yang seharusnya dapat fokus kepada audiens untuk menyampaikan materi presentasi. Adanya operator sebagai asisten tentu menyebabkan pembicara perlu untuk melakukan koordinasi dan sinkronisasi dengannya. Penggunaan perangkat pendukung juga menyebabkan pembicara diharuskan secara terus menerus selama presentasi berlangsung untuk dapat mengoperasikan perangkat, sehingga akan berpengaruh terhadap penguasaan panggung dan adanya gestur tubuh yang tidak diperlukan.

4.1.2 Analisis Sistem yang Diusulkan



Gambar 4. 2 Arsitektur sistem usulan

Gambar 4. 2 merupakan usulan rancangan sistem untuk dapat menanggulangi permasalahan yang timbul dari sistem sebelumnya. Sistem usulan ini memiliki perbedaan pada cara memberikan perintah pada mesin (komputer) untuk dapat mengontrol konten *slide* presentasi. Pembicara dapat secara langsung mengoperasikan komputer hanya dengan perintah suara (*voice command*).

Tabel 4. 1 Kata kunci dan perintah eksekusi

No	Kata	Kegunaan
1	Down	Menuju ke slide berikutnya ketika tidak sedang *slide show*
2	Go	Memainkan media player
3	Left	Menuju ke slide sebelumnya ketika sedang *slide show*
4	Off	Mengubah state app menajadi OFF (tidak menerima perintah) hingga dihidupkan kembali
5	On	Mengubah state app menjadi ON (mampu menerima perintah)
6	Right	Menuju ke slide berikutnya ketika sedanga *slide show*
7	Stop	Mematikan streaming audio dan terminate program
8	Up	Menuju ke slide sebelumnya ketika tidak sedang *slide show*

Keuntungan dari penggunaan *voice command* ini adalah pembicara tidak memerlukan bantuan tambahan seperti operator sebagai asisten dan perangkat pendukung lainnya. Pembicara dapat mengucapkan kata perintah yang telah disimpan di basis data dan dilatih dengan model *machine learning* yang ada. Hal ini tentu akan memudahkan pembicara dalam menyampaikan materi sekaligus mengontrol penampilan *slide* presentasi yang diinginkan secara bersamaan dengan baik. Dengan demikian, distraksi presentasi untuk pembicara yang diakibatkan oleh pengoperasian presentasi secara teknis dapat berkurang.

a. Analisis Fungsional

Kebutuhan fungsional merupakan kebutuhan yang mencakup segala proses yang terdapat pada sistem. Adapun kebutuhan fungsional pada penelitian ini dapat dapat dideskripsikan sebagai berikut:

1. Kebutuhan Masukan

Kebutuhan masukan (*input*) merupakan kebutuhan yang diperoleh dari pengguna berupa perintah pengoperasian sistem dan/atau data yang diperlukan. Kebutuhan masukan dalam penelitian "Penerapan *Speech Recognition* Menggunakan Metode *Long Short-Term Memory* (LSTM) untuk Presentasi Dinamis" adalah sebagai berikut:

1) *Input* suara pengguna ketika melakukan presentasi.

2. Kebutuhan Proses

Kebutuhan proses merupakan kebutuhan sistem untuk dapat melakukan pemrosesan data masukan dengan perhitungan dan pengolahan data untuk mendapatkan hasil sesuai yang diharapkan. Kebutuhan proses dalam penelitian "Penerapan *Speech Recognition* Menggunakan Metode *Long Short-Term Memory* (LSTM) untuk Presentasi Dinamis" adalah sebagai berikut:

1) Pengguna membuka program/sistem pengenalan suara;
2) Pengguna memberikan *input* berupa sinyal suara pada alat perekam suara;
3) Sistem menangkap sinyal suara yang masuk;
4) Sinyal suara didigitalisasi agar sistem mampu mendapatkan data berupa kumpulan angka;
5) Sistem memuat model LSTM yang telah dilatih untuk dapat melakukan *fitting* pada data *input*;
6) Sistem memecah data suara untuk mendapatkan masing-masing bagian suara selama 1 detik;
7) Bagian-bagian suara diklasifikasikan oleh sistem sehingga dapat diketahui kata apa yang tepat (*keyword spotting*)
8) Apabila ditemukan kata yang sesuai dengan target, maka lakukan

ubah menjadi perintah untuk mengoperasikan keyboard secara atomatis;

9) Keyboard dieksekusi untuk dapat mengoperasikan satu atau beberapa fungsionalitas *software* presentasi sesuai dengan waktu ucapan pengguna.

3. Kebutuhan Luaran

Kebutuhan luaran merupakan kebutuhan sistem untuk dapat menampilkan dan/atau mengeksekusi perintah sesuai dengan yang diinginkan pengguna. Kebutuhan luaran dalam penelitian "Penerapan *Speech Recognition* Menggunakan Metode *Long Short-Term Memory* (LSTM) untuk Presentasi Dinamis" adalah sebagai berikut:

1) Hasil *keyword spotting* akan dieksekusi sesuai dengan prediksi dan waktu pengucapan, sehingga dapat dijadikan sebagai perintah suara untuk mengoperasikan *software* presentasi.

b. Analisis non Fungsional

Kebutuhan non fungsional merupakan kebutuhan tambahan sistem yang mampu mendukung agar mampu menerima inputan, melakukan pemrosesan, dan mengeksekusi hasil luaran. Penelitian ini memiliki kebutuhan non fungsional sebagi berikut:

1. Kebutuhan Perangkat Lunak

Berikut merupakan perangkal lunak pendukung yang digunakan sistem:

1) *Operating system*

Windows 10 sebagaai sistem operasi perangkat yang digunakan pada penelitian, pengujian, dan penerapan sistem.

2) Code Editor

Visual Studio Code digunakan untuk merancang sistem dengan bahasa pemrograman Python. File python dengan ekstensi ipynb digunakan untuk visualisasi dan proses pelatihan model, sedangkan file dengan ekstensi py digunakan untuk menjalankan sistem

pengenalan suara.

3) *Diagram Maker*

Draw.io digunakan untuk membuat diagram dan gambar rancangan arsitektur sistem lainnya.

4) *Interface Design*

2. Kebutuhan Perangkat Keras

Berikut merupakan perangkat keras yang digunkan:

1) Processor : Intel® Core™ i5-4300U CPU@1.90GHz 2.49Ghz
2) Grafik : Intel® HD Graphics Family
3) RAM : 8GB
4) Penyimpanan : 170GB

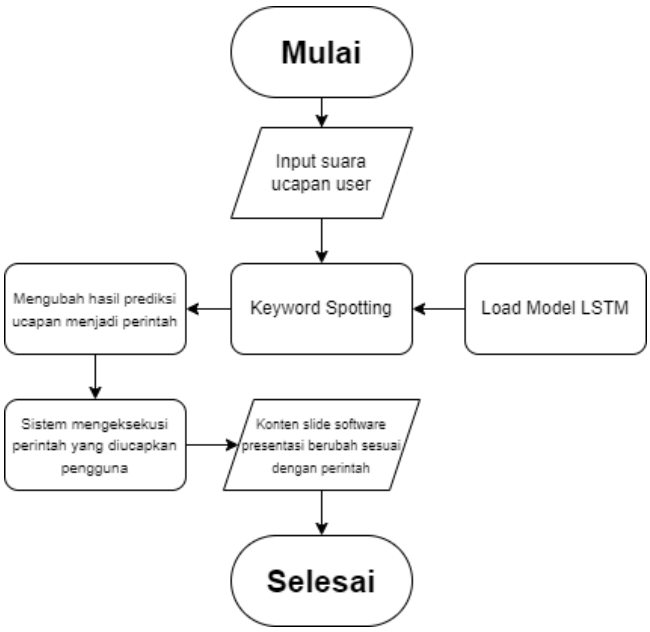
4.2 Desain Sistem

4.2.1 Perancangan Logic

Perancangan *logic* adalah perencanaan keseluruhan sistem yang dibangun untuk menjelaskan konsep dasar kerja sistem, sehingga mampu memberikan solusi secara prinsip. Model perancangan konseptual terdiri atas identifikasi entitas, relasi, kardinalitas, dan konstrain dari kasus yang ada. Model ini dapat dimodifikasi sesuai dengan perubahan kebutuhan data atau alur sistem yang dikembangkan. Berikut merupakan model perancangan konseptual yang disusun dalam penelitian ini.

1. Flowchart

a) Flowchart Sistem

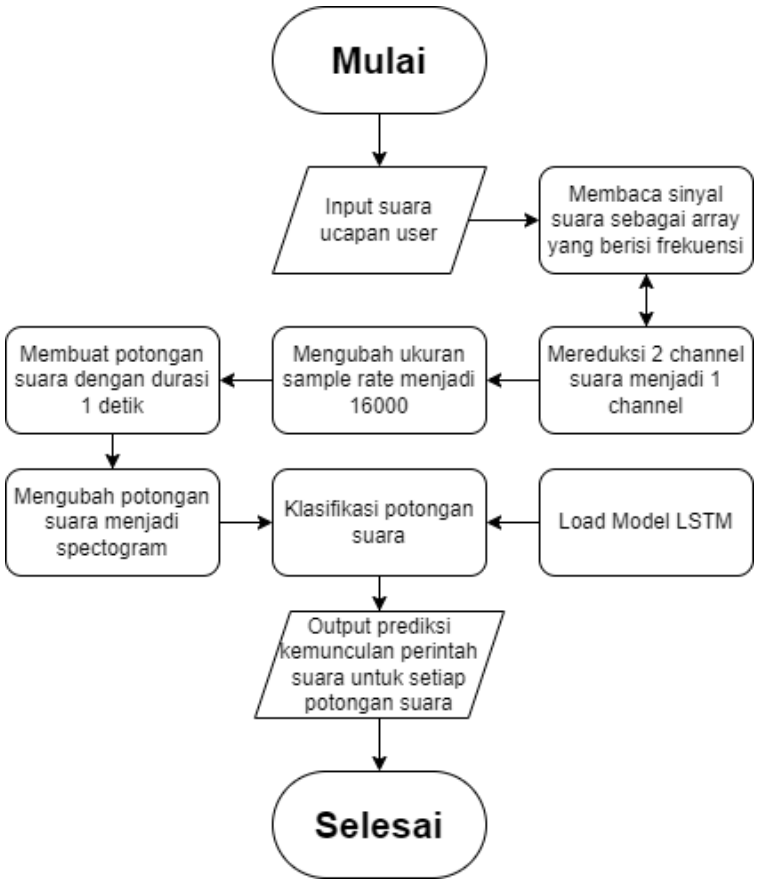


Gambar 4. 3 Flowchart sistem

Gambar 4. 3 menunjukkan alur kerja sistem sehingga mampu mengenali ucapan pada suatu sinyal suara. Pencarian kata (*keyword spotting*) dapat dilakukan dengan algoritma LSTM. Pemilihan algoritma LSTM didasarkan pada data yang diolah merupakan *time series*. Dengan memperhatikan urutan waktu dan intensitas sara pada *audio spectogram*, LSTM mampu mengenali kata apa yang muncul pada sinyal suara yang masuk.

Proses *keyword spotting* digunakan untuk mencari kata-kata yang muncul pada input sinyal suara. Sistem hanya mampu mengenali kosa kata tertentu (*limited vocabulary*) sesuai dengan data yang dilatih untuk membangun model *machine learning*. Sistem akan memprediksi kata yang muncul, sehingga komputer dapat mengeksekusi suatu perintah berdasarkan hasil klasifikasi potongan suara pada waktu tertentu. Dengan demikian, pengguna dapat memberikan perintah suara untuk mengubah penampilan konten *software presentasi* sesuai dengan kata asli dan waktu pengucapannya.

b) Flowchart Pengenalan Ucapan *Keyword Spotting*

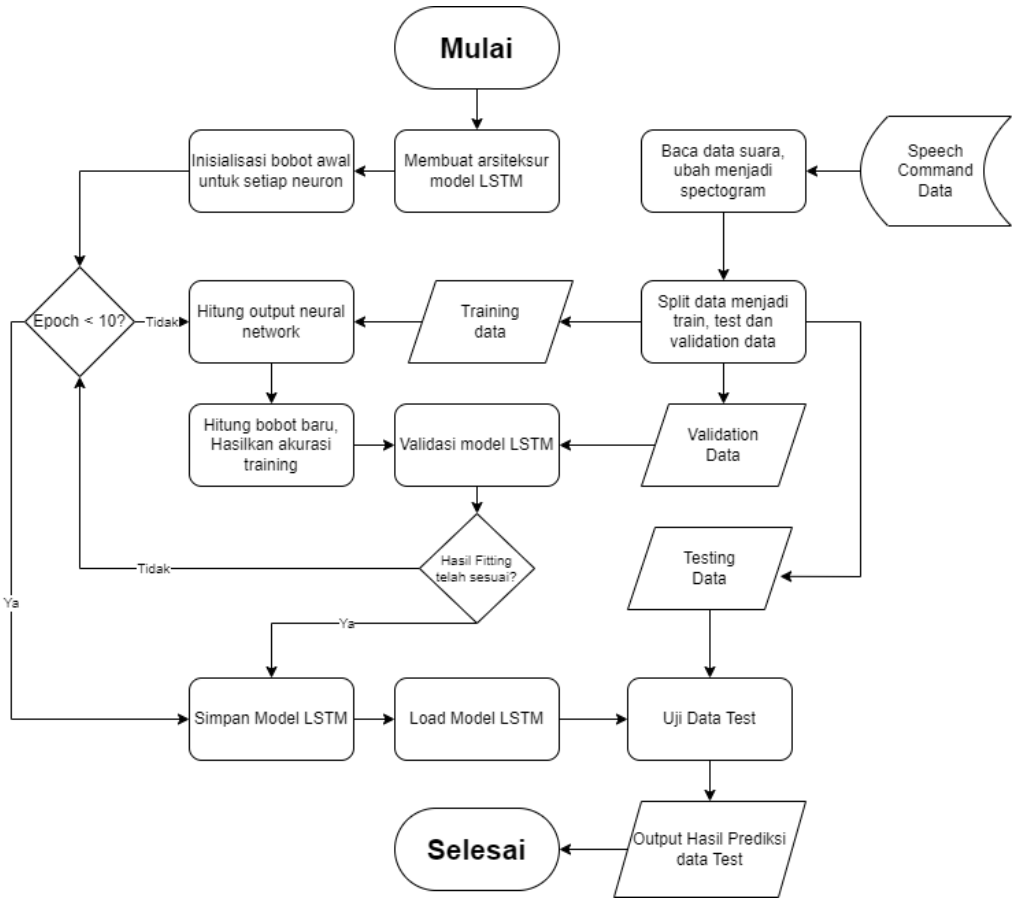


Gambar 4. 4 Flowchart *Keyword Spotting*

Gambar 4. 4 menjelaskan alur dari proses *keyword spotting*. Sinyal suara yang masuk akan dijadikan sebagai data *numerical sequence* yang merupakan frekuensi suara. Umumnya, suara memiliki 2 (dua) saluran (*channel*) yang terbagi menjadi *channel* kanan dan kiri sehingga pendengar mampu mendengar suara sesuai dengan posisi. Untuk proses analisis, kita perlu mereduksi ukuran channel menjadi 1 (satu).

Audio kemudian akan dipotong sehingga menghasilkan bagian-bagian kecil suara yang berdurasi 1 detik dengan sample rate 16000. Setiap potongan suara diubah menjadi spectogram untuk dapat dilakukan analisis. Spectogram diklasifikasikan menggunakan model LSTM untuk memprediksi kemunculan kata pada setiap detiknya.

c) Flowchart Pembuatan Model LSTM



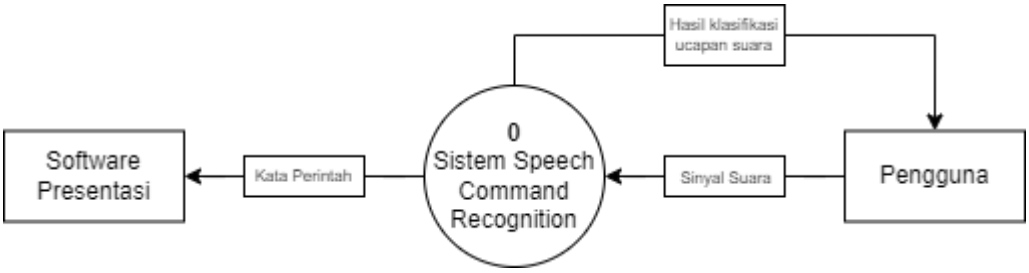
Gambar 4. 5 Flowchart pembuatan model LSTM

Gambar 4. 5 merupakan alur proses pembuatan model LSTM. Dari arsitektur LSTM yang telah dibuat, akan dicari bobot terbaik sehingga mampu memprediksi kemunculan kata dengan tepat. Dataset akan dibagi menjadi 3 (tiga) sesuai dengan penggunaannya, yaitu data uji, latih, dan validasi. Data latih digunakan untuk mencari bobot neuron yang akan berubah setiap iterasi data. Data validasi digunakan untuk melakukan validasi bobot setelah 1 epoch selesai diproses. Data uji digunakan untuk memeriksa seberapa baik model akhir yang dihasilkan dari proses pelatihan model untuk dapat melakukan klasifikasi suara dengan tepat.

2. Diagram Konteks

Diagram konteks adalah diagram untuk menentukan konteks dan batasan model sistem dan berisika gambaran umum dari sistem tersebut. Diagram konteks

menjadi tingkatan paling atas dari *data flow diagram* (DFD). DFD level 0 ini menggambarkan alur data secara global dalam sistem dengan logika.

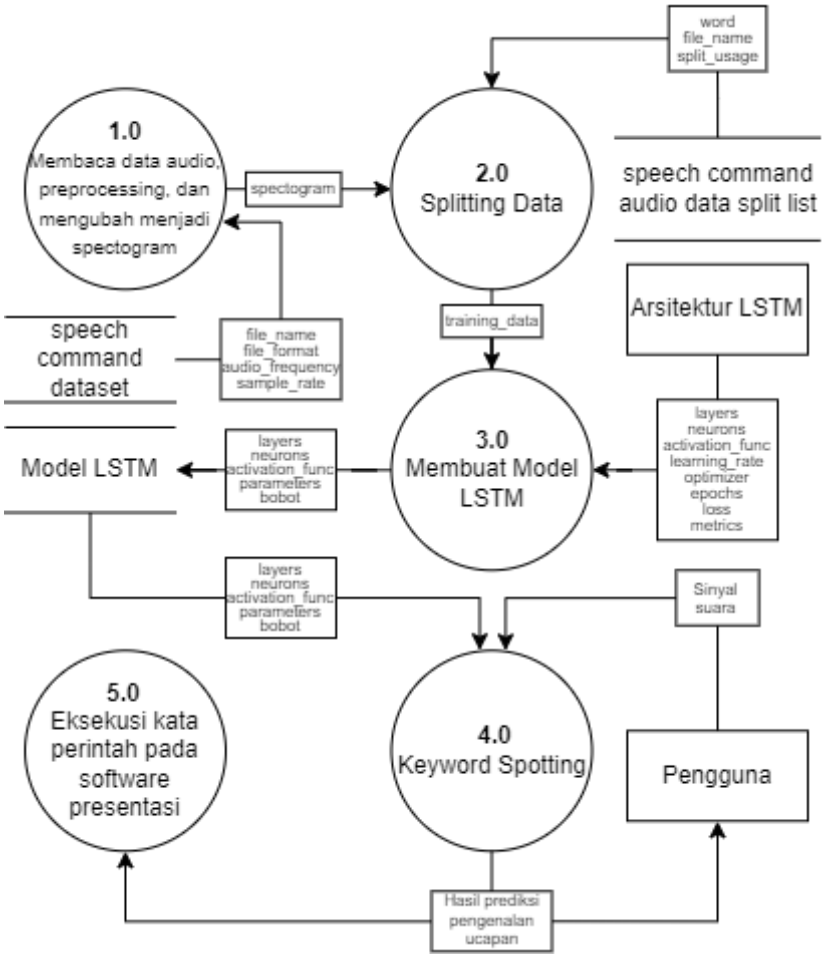


Gambar 4. 6 DFD Level 0

Gambar 4. 6merupakan DFD level 0 dari sistem yang dirancang untuk penelitian "Penerapan *Speech Recognition* Menggunakan Metode *Long Short-Term Memory* (LSTM) untuk Presentasi Dinamis". Secara umum, sistem akan menerima data berupa sinyal suara dari pengguna. Sinyal suara yang masuk ini akan diolah untuk dihasilkan klasifikasi ucapan untuk dapat memberikan perintah pengoperasian software presentasi.

3. Data Flow Diagram

Data flow diagram (DFD) merupakan diagram alir data pada suatu rancangan sistem. Diagram ini menggambarkan pergerakan data di antara tempat penyimpanan, entitas, dan fungsionalitas. DFD memiliki beberapa tingkatan yang menandakan terdapatnya sub proses untuk setiap proses yang ada sehingga penggunaan data dapat diketahui dengan jelas. Semakin banyak tingkatan yang dirancang, maka semakin rinci dan kompleks alur pertukaran data yang terjadi pada sistem.



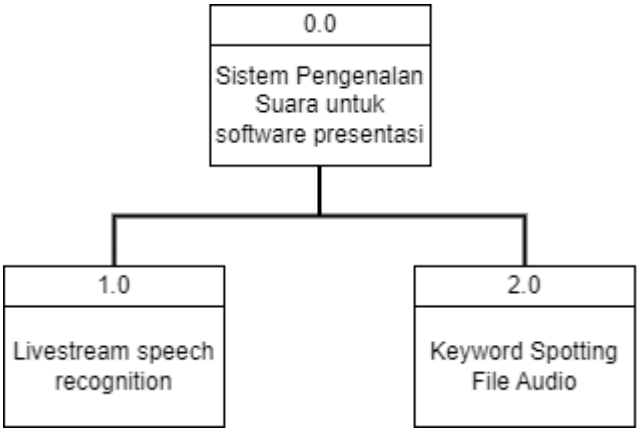
Gambar 4. 7 DFD Level 1

Gambar 4. 7 menunjukkan DFD level 1 dari sistem yang dirancang. Pada tingkatan ini digambarkan pertukaran data untuk pemrosesan data, pembagian data, pelatihan model, dan klasifikasi untuk menemukan kata pada sinyal suara (*keyword spotting*). Data suara diubah menjadi spectogram untuk kemudian dilakukan pembuatan model sesuai dengan pembagian datanya. Arsitektur dan bobot dari model LSTM kemudian akan disimpan dan digunakan untuk proses klasifikasi kata yang muncul pada suara. Kata yang terdeteksi sebagai perintah, kemudian akan dieksekusi untuk mengoperasikan software presentasi.

4. Diagram Jenjang

Diagram jenjang merupakan diagram yang terdiri atas bagian berjenjang sebagai struktur dari sistem yang menunjukkan urutan proses yang terjadi di dalamnya. Diagram ini juga disebut dengan HIPO (*Hierarchy plus Input-Process-*

Oputput). HIPO menggambarkan hierarki dari suatu sistem dari yang tertinggi hingga terendah. Urutan ini menunjukkan alur rancangan sistem yang bermula dari atas ke bawah untuk menjelaskan proses atau modul apa yang termuat pada modul dengan hierarki yang lebih tinggi.

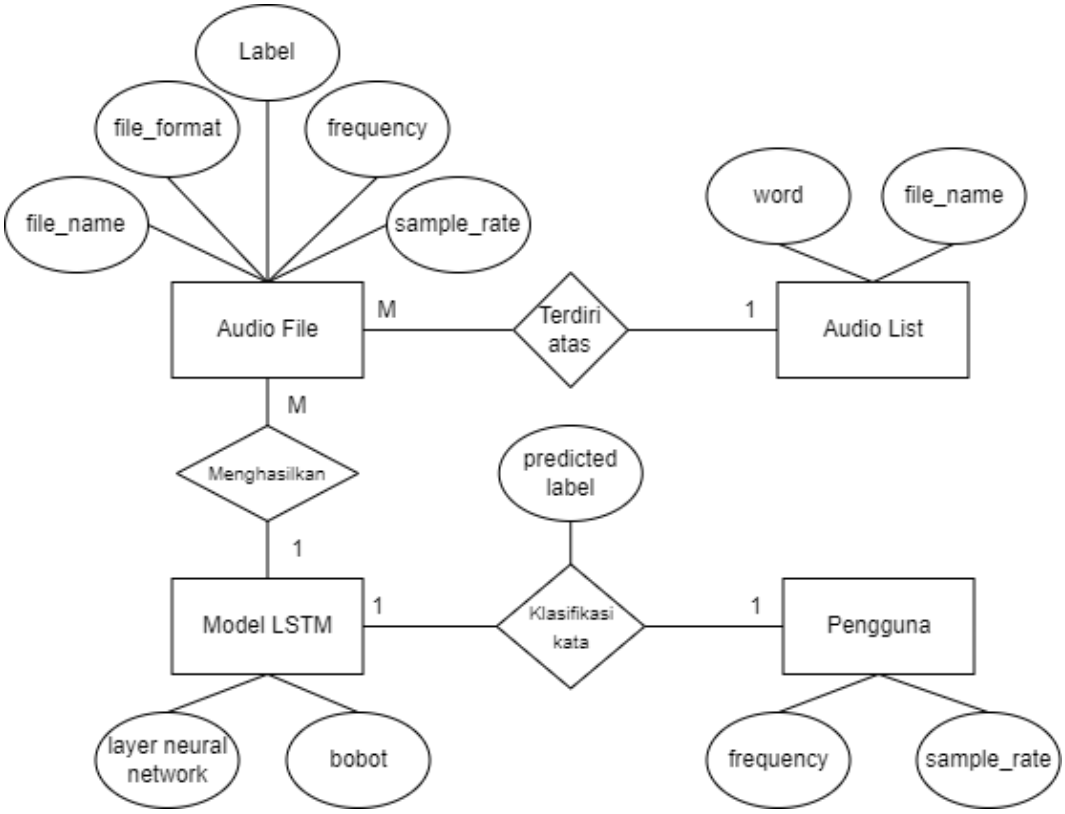


Gambar 4. 8 Diagram HIPO

Gambar 4. 8 merupakan diagram HIPO dari sistem yang dirancang. Hierarki tertinggi menunjukkan sistem secara keseluruhan digunakan untuk pengenalan suara yang dimanfaatkan pada pengoperasian perangkat lunak presentasi. Sistem ini memiliki 2 modul utama, yaitu dapat melakukan pengenalan ucapan dengan *livestream* secara *real-time* dan input file suara yang telah ada.

5. Entity Relationship Diagram (ERD)

Gambar 3.10 menunjukkan ERD yang menjelaskan entitas apa saja yang adadan bagaimana entitas tersbut dapat berelasi. Dari dataset yang diperoleh, tersedia list pembagian data (*dataset split*) menjadi *training, testing,* dan *validation*. List tersebut berisi kumpulan nama file suara dengan urutan *[kata]/[nama_file].wav*. Dengan pembagian data tersebut, audio file kemudian dapat dibaca sesuai dengan kebutuhan untuk dilakukan pelatihan model LSTM. Model *machine learning* yang telah dihasilkan akan digunakan *classifier* untuk menemukan kata pada suara.



Gambar 4. 9 *Entity Relationship Diagram* (ERD)

4.2.2 Perancangan Fisik

Perancangan fisik merupakan implementasi dari perancangan konseptual yang dapat digunakan atau diterapkan pada sistem serta berorientasi pada pengguna.

a) Perancangan Tabel

1) Struktur Tabel *Audio File*

Tabel *audio file* digunakan untuk menyimpan data suara yang tersedia pada direktori sehingga diketahui file suara memuat ucapan kata tertentu. Pembacaan file suara memerlukan *file path* direktori untuk dapat memuat file yang diinginkan. Pembagian penggunan file menjadi data uji, latih, dan validasi juga dapat ditentukan dengan mengambil referensi pada file *training_list.txt, testing_list.txt,* dan *validation.txt*.

Tabel 4. 2 Atribut Entitas *Audio File*

tb_audio_file

Atribut	Tipe Data	Length	Keterangan
id	int	10	primary key unique not null
file	Varchar	128	not null
file_path	Varchar	256	not null
word	Varchar	25	not null
usage	Int	2	nullable

2) Struktur Tabel Prediksi Data Uji

Tabel prediksi data uji digunakan untuk membandingkan kata sebenarnya yang muncul pada file suara dengan hasil prediksi kata. Perbandingan data uji dengan prediksi ini bertujuan untuk mengetahui berapa akurasi prediksi yang dihasilkan dari model LSTM yang telah dibuat.

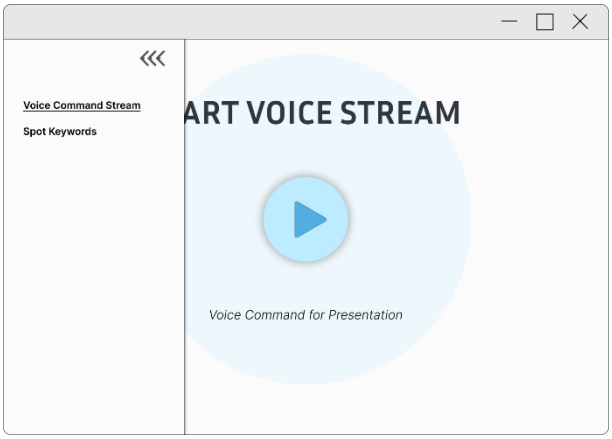
Tabel 4. 3 Atribut Entitas Prediksi Data Uji

tb_prediction_testing			
Atribut	**Tipe Data**	**Length**	**Keterangan**
file	Varchar	128	not null
file_path	Varchar	256	not null
word	Varchar	25	not null
predicted word	Varchar	25	not null

b) Perancangan Antar Muka (*Interface*)

Antar muka (*interface*) merupakan tampilan sistem yang akan digunakan oleh pengguna. Tampilan ini berisi informasi atau pilihan pengguna untuk dapat melakukan interaksi dengan sistem. Data-data yang diperlukan atau telah diolah dapat ditampilkan pada sistem sebagai bentuk informasi kepada pengguna. Adapaun rancangan antar muka untuk sistem yang dirancang adalah sebagai berikut:

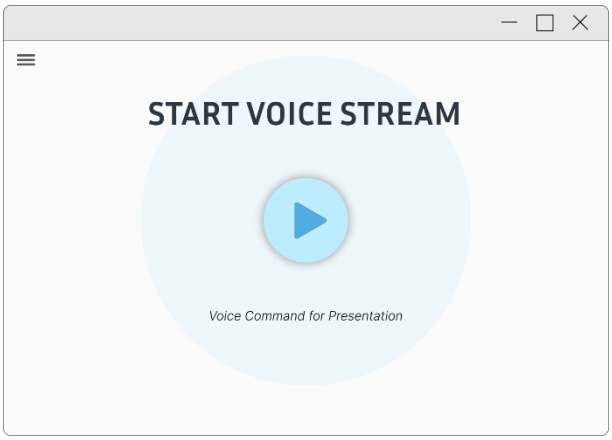
1) Menu Utama



Gambar 4. 10 Tampilan menu utama

Gambar 4. 10 menampilkan menu utama yang berisi pilihan fitur untuk pengguna dapat melakukan pengenalan suara secara *real-time* dengan *livestream* atau menginputkan file suara yang dimiliki.

2) Real-Time Speech Recognition



Gambar 4. 11 Tampilan *real-time speech recognition*

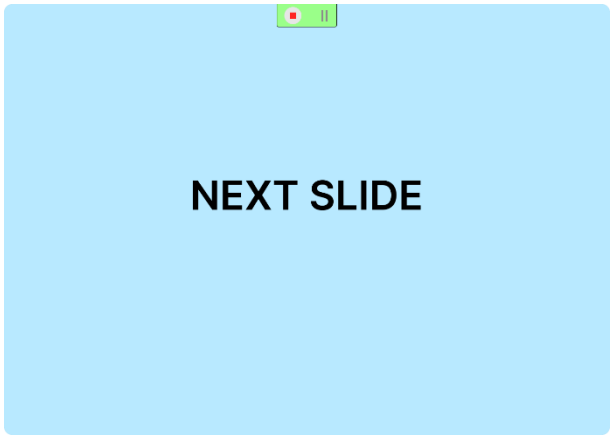
Gambar 4. 11 menampilkan *real-time speech recognition* telah diaktifkan. Pengguna dipastikan untuk telah membuka software presentasi pada desktop maupun web.

3) Voice recognition stream presentasi (tidak terdeteksi kata perintah)

Gambar 4. 12 Tampilan *audio stream* tidak terdeteksi kata perintah

Gambar 4. 12 menunjukkan fitur *voice recognition stream* telah berjalan ditunjukkan dengan adanya layer tampilan kecil di tengah atas. Pengguna dapat melakukan penjelasan presentasi dan mengucapkan kata perintah tertentu untuk dapat merubah konten slide sesuai dengan yang diinginkan.

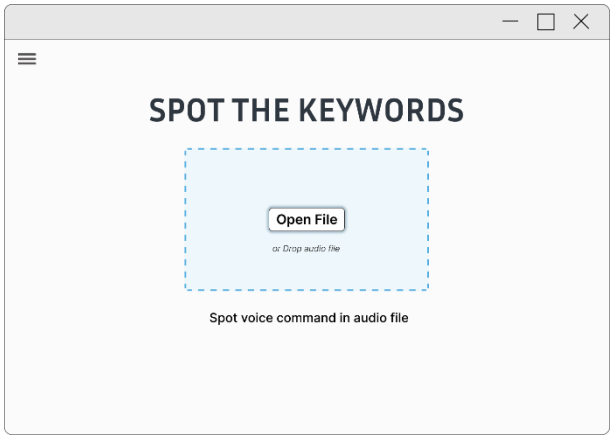
4) Voice recognition stream presentasi (terdeteksi kata perintah)



Gambar 4. 13 Tampilan *audio stream* terdeteksi kata perintah

Gambar 4. 13 menunjukkan terdeteksinya kata perintah pada sinyal suara yang masuk saat *streaming* suara. Ketika sistem mendeteksi adanya kata perintah, maka akan memberikan tanda kepada pengguna dengan merubah warna layer menjadi hijau sebagai indikator. Kata perintah yang terdeksi sistem kemudian akan dieksekusi untuk mengubah konten tampilan slide *software* presentasi.

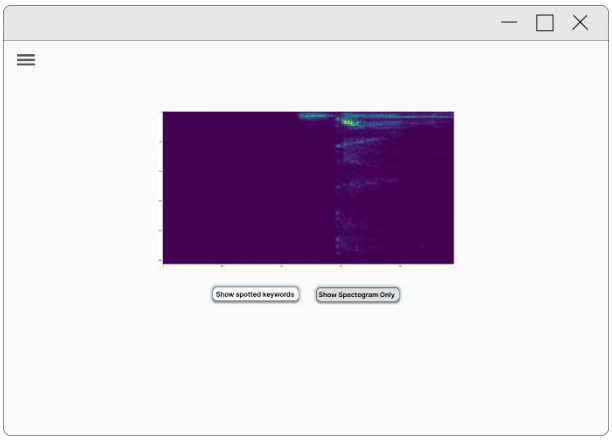
5) Input File Suara



Gambar 4. 14 Tampilan open file audio

Gambar 4. 14 menampilkan proses untuk pengguna dapat memasukkan file suara yang dimiliki untuk mendeteksi apakah terdapat kata perintah yang muncul pada file suara tersebut.

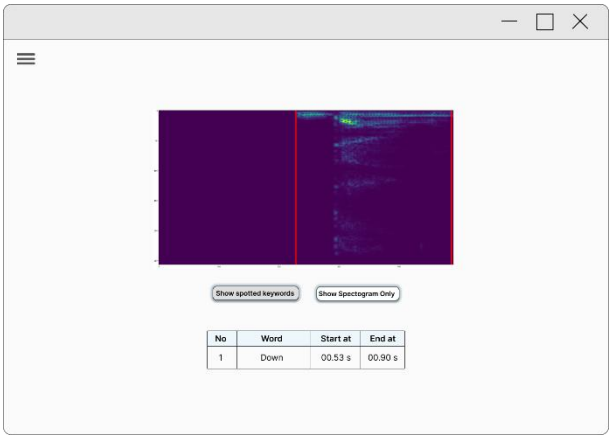
6) Spectogram



Gambar 4. 15 Tampilan spectogram dari file suara

Gambar 4. 15 menampilkan vidsualisasi spectrogram dari file audio yang dimasukkan.

7) Spectogram (kata terdeteksi)



Gambar 4. 16 Tampilan spectogram dengan deteksi dari file suara

Gambar 4. 16 menunjukkan kata perintah yang muncul pada spectogram dari file yang dimasukkan pengguna.

BAB V
IMPLEMENTASI DAN HASIL SERTA PEMBAHASAN

5.1 Implementasi

5.1.1 Kebutuhan *dependencies*

Dependencies merupakan ketergantungan sistem terhadap *library* atau *package* pendukung yang menyediakan berbagai fungsionalitas. Python menyediakan banyak *dependencies* yang dapat digunakan sehingga mampu memberikan efisiensi bagi pengembang karena tidak perlu membuat suatu algoritma dari dasar (*from scratch*). Untuk mengelola *packages* pada sistem, digunakan dependensi python, yaitu PIP (pip Install Packages).

```
import os
import pathlib
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.losses import SparseCategoricalCrossentropy
from IPython import display
import pynput
import pyaudio
```

Gambar 5. 1 Kebutuhan *dependencies* sistem

Potongan kode pada Gambar 5. 1 Kebutuhan *dependencies* sistem menggunakan beberapa *dependencies* dengan kegunaan sebagai berikut:

Tabel 5. 1 Kegunaan *dependencies* sistem

No	*Dependencies*	Kegunaan
1	os	Berinteraksi dengan sistem operasi, yaitu fungsi untuk akses files dan direktori, serta tugas lainnya
2	pathlib	Mengakses *filesystem path*
3	matplotlib.pyplot	Plotting data
4	numpy	Komputasi numerical
5	seaborn	Visualisasi data
6	tensorflow	*Machine learning framework*
7	tensorflow.keras.layers	Membangun *layers* model *deep learning*
8	tensorflow.keras.models	Membangun *layers* model *deep learning*
9	SparseCategoricalCrossentropy	Menghitung loss klasifikasi multi kelas
10	IPython.display	Menampilkan data secara dinamis dan interaktif
11	pynput	Kontrol perangkat masukan seperti keyboard dan mouse
12	pyaudio	Kontrol perangkat masukan *microphone*

5.1.2 Pengolahan dan Persiapan Data

1) Menentukan *random seed*

Seed digunakan untuk memastikan reproduktibilitas saat pembangkitan angka acak. Reproduktibilitas dapat diartikan kode mampu menghasilkan hasil yang sama guna memvalidasi temuan, dan mampu dibandingkan dengan metode yang berbeda. Dengan mengatur nilai *seed*, urutan angka acak yang didapat akan sama setiap kali menjalankan kode.

```
# Set the seed value for experiment reproducibility.
seed = 42
tf.random.set_seed(seed)
np.random.seed(seed)
```

Gambar 5. 2 Potongan kode *random seed*

2) Membaca dataset

Data disimpan pada direktori ‘data/speech_commands’ yang berisi kumpulan file audio ucapan yang dipisahkan menjadi folder yang berbeda-beda dengan nama sesuai dengan label datanya.

```
DATASET_PATH = 'data/speech_commands'
data_dir = pathlib.Path(DATASET_PATH)
train_ds, val_ds = tf.keras.utils.audio_dataset_from_directory(
   directory=data_dir,
   batch_size=64,
   validation_split=0.2,
   seed=0,
   output_sequence_length=16000,
   subset='both')
```

Gambar 5. 3 Potongan kode membaca dataset

tf.keras.utils audio_dataset_from_directory merupakan fungsi untuk membaca file audio dalam suatu direktori. Fungsi ini membagi dataset menjadi *training* dan *validaiton* dengan menentukan *split rate* dari data validasi. Output_sequence_length merupakan panjang dari audio yang dibaca. Panjang *sequence* 16000 berarti audio akan dimuat dalam durasi 1 detik.

3) *Data splitting* (pembagian data)

Data splitting merupakan proses untuk memisahkan data utuh menjadi data uji, latih, dan validasi.

```
Train_ds = train_ds
test_ds = val_ds.shard(num_shards=2, index=0)
```

```
val_ds = val_ds.shard(num_shards=2, index=1)
```

Gambar 5. 4 Potongan kode *splitting* data

fungsi *shard* berasal dari *library* tensorflow yang digunakan untuk membagi data menjadi beberapa bagian tertentu. Pada kode di atas, data validasi akan digunakan setengahnya (50%) untuk data uji.

4) Reduksi dimensi

File audio yang dimuat sebelumnya memiliki bentuk 3 dimensi, yaitu (None, 16000, None). Terdapat 1 dimensi yang redundan sehingga dapat direduksi dengan menggunakan fungsi *squezee*

```
def squeeze(audio, labels):
  audio = tf.squeeze(audio, axis=-1)
  return audio, labels

train_ds = train_ds.map(squeeze, tf.data.AUTOTUNE)
val_ds = val_ds.map(squeeze, tf.data.AUTOTUNE)
test_ds = test_ds.map(squeeze, tf.data.AUTOTUNE)
```

Gambar 5. 5 Potongan kode reduksi dimensi

5) Mengubah waveform menjadi *spectrogram*

File audio yang dimuat mula-mulanya akan menjadi data *waveform* dengan panjang 16000 *sample rate* (durasi 1 detik). Data *waveform* ini perlu diubah menjadi bentuk *spectrogram* untuk dapat dianalisis frekuensi sinyal suara berdasarkan waktu.

```
def get_spectrogram(waveform):
  # Convert the waveform to a spectrogram via a STFT.
  spectrogram = tf.signal.stft(
      waveform, frame_length=255, frame_step=128)
  # Obtain the magnitude of the STFT.
  spectrogram = tf.abs(spectrogram)
  # Add a `channels` dimension, so that the spectrogram can be used
  # as image-like input data with convolution layers (which expect
  # shape (`batch_size`, `height`, `width`, `channels`).
  spectrogram = spectrogram[..., tf.newaxis]
  return spectrogram

def make_spec_ds(ds):
  return ds.map(
      map_func=lambda audio,label: (get_spectrogram(audio), label),
      num_parallel_calls=tf.data.AUTOTUNE)

train_spectrogram_ds = make_spec_ds(train_ds)
val_spectrogram_ds = make_spec_ds(val_ds)
test_spectrogram_ds = make_spec_ds(test_ds)
```

Gambar 5. 6 Potongan kode *waveform to spectrogram*

6) Normalisasi

Normalisasi merupakan metode untuk mengubah data numerical pada data

sehingga sesuai dengan skala yang telah ditetapkan.

```
# Instantiate the `tf.keras.layers.Normalization` layer.
norm_layer = layers.Normalization()
# Fit the state of the layer to the spectrograms
# with `Normalization.adapt`.
norm_layer.adapt(data=train_spectrogram_ds.map(map_func=lambda spec, label: spec))
```

Gambar 5. 7 Potongan kode normalisasi data training

5.1.3 Visualisasi Data

1) Visualisasi *waveform*

Waveform divisualisasikan ke dalam bentuk plot untuk mengetahui pola data yang terjadi. Setiap kata ucapan memiliki intonasi dan penekanan yang berbeda, namun pola yang dihasilkan akan sama untuk setiap kata.

```
for example_audio, example_labels in train_ds.take(1):
  break

rows = 4
cols = 2
n = rows * cols
fig, axes = plt.subplots(rows, cols, figsize=(16, 9), )
fig.tight_layout()

labels_found = []
for i in range (n):
  for label_idx, label in enumerate(example_labels):
    if (label not in labels_found) & (label == i):
      r = i // cols
      c = i % cols
      ax = axes[r][c]
      ax.plot(example_audio[label_idx].numpy())
      label_name = label_names[example_labels[label_idx]]
      ax.set_title(label_name)
      ax.set_ylim([-1.1,1.1])
      labels_found.append(label)
      break
```

Gambar 5. 8 Potongan kode visualisasi *waveform*

2) Visualisasi *spectrogram*

Spectrogram divisualisaikan untuk mendapat gambaran mengenai bagaimana pola distribusi frekuensi sinyal suara dari file audio yang dimuat.

```
def plot_spectrogram(spectrogram, ax):
  if len(spectrogram.shape) > 2:
    assert len(spectrogram.shape) == 3
    spectrogram = np.squeeze(spectrogram, axis=-1)
  # Convert the frequencies to log scale and transpose, so that the time is
  # represented on the x-axis (columns).
  # Add an epsilon to avoid taking a log of zero.
  log_spec = np.log(spectrogram.T + np.finfo(float).eps)
  height = log_spec.shape[0]
```

```
  width = log_spec.shape[1]
  X = np.linspace(0, np.size(spectrogram), num=width, dtype=int)
  Y = range(height)
  ax.pcolormesh(X, Y, log_spec)

for example_spectrograms, example_spect_labels in train_spectrogram_ds.take(1):
  break

rows = 4
cols = 2
n = rows*cols
fig, axes = plt.subplots(rows, cols, figsize=(16, 9))
fig.tight_layout()

labels_found = []
for i in range (n):
  for label_idx, label in enumerate(example_spect_labels):
    if (label not in labels_found) & (label == i):
      r = i // cols
      c = i % cols
      ax = axes[r][c]
      plot_spectrogram(example_spectrograms[label_idx].numpy(), ax)
      label_name = label_names[example_spect_labels[label_idx].numpy()]
      ax.set_title(label_name)
      labels_found.append(label)
      break
plt.show()
```

Gambar 5. 9 Potongan kode visualisasi *spectrogram*

5.1.4 Pembuatan Model *Machine Learning* LSTM

1) Membuat *sequential layers* dengan LSTM

Penelitian ini dilakukan dengan mengekperimenkan model LSTM dengan susunan layer yang berbeda. Untuk membatasi variabel eksperimen, maka ditetukan model yang akan dibuat dibedakan dengan jumlah unit dan penggunaan regularisasi *dropout*. Pada Tabel 5. 2 menunjukkan model-model yang dibuat dengan susunan layer dan jumlah unit yang berbeda:

Tabel 5. 2 Model LSTM yang dikembangkan

Nama model	Layers	Unit	Parameter
LSTM_1	LSTM	128	*return_sequences* = True, *input_shape*
	LSTM	64	
	Dense	6	*activation* = “softmax”
LSTM_2	LSTM	128	*return_sequences* = True, *input_shape*
	Dropout	-	*rate* = 0.5
	LSTM	64	
	Dense	6	*activation* = “softmax”
LSTM_3	LSTM	256	*return_sequences* = True, *input_shape*
	LSTM	128	

	Dense	6	*activation* = “softmax”
LSTM_4	LSTM	256	*return_sequences* = True, *input_shape*
	Dropout	-	*rate* = 0.5
	LSTM	128	
	Dense	6	*activation* = “softmax”

Tabel 5. 2 menunjukkan 4 model LSTM untuk ekperimen sebagai perbandingan untuk menentukan model yang terbaik berdasarkan hasil *fitting* dan evaluasi. Implementasi untuk pembuatan model di atas dibentuk dengan *layers* sequential menggunakan *library* tenfsorflow keras. Adapun contoh potongan kode pada Gambar 5. 10 merupakan model untuk LSTM_4:

```
input_shape = example_spectrograms.shape[1:]
label_names = np.array(train_ds.class_names)
num_labels = len(label_names)

model = tf.keras.Sequential([
    layers.LSTM(units=256, return_sequences=True, input_shape=(124, 129)),
    layers.Dropout(0.5),
    layers.LSTM(units=128),
    layers.Dense(num_labels, activation='softmax')
])
model.summary()
```

Gambar 5. 10 Potongan kode membangun model dengan LSTM

2) *Compiling model*

Compile model merupakan tahapan untuk mengkonfirugasi beberapa pengaturan pada model sebelum proses *fitting* dijalankan. Pada penelitian ini ditentukan optimizer Adam dengan nilai learning rate default (0,001), fungsi *loss Sparse Categorical Crossentropy*, dan metrik akurasi.

```
model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss=SparseCategoricalCrossentropy(),
    metrics=['accuracy'],
)
```

Gambar 5. 11 Potongan kode compile model

3) *Fitting model*

Proses *fitting* merupakan tahapan untuk melatih data sehingga model deep learning yang dibuat mampu menghasilkan hasil akhir bobot yang terbaik. Bobot ini akan diperbaharui dengan perhitungan sesuai dengan layer yang menyusun pada setiap epoch-nya.

```
EPOCHS = 10
history = model.fit(
```

```
    train_spectrogram_ds,
    validation_data=val_spectrogram_ds,
    epochs=EPOCHS,
    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=2),
)
```

Gambar 5. 12 Potongan kode *fitting* model

4) Evaluasi model

Model yang telah melalui proses *fitting* akan dievaluasi untuk mengetahui seberapa baik hasil akhir pelatihan model yang diperoleh. Akurasi klasifikasi dan nilai *loss* pada data uji digunakan sebagai tolak ukur baik atau tidaknya model yang dihasilkan. Model akan dievaluasi dengan meghitung akurasi prediksi klasifikasi terhadap data asli dan confusion matrix.

```
model.evaluate(test_spectrogram_ds, return_dict=True)


confusion_mtx = tf.math.confusion_matrix(y_true, y_pred)
plt.figure(figsize=(10, 8))
sns.heatmap(confusion_mtx,
            xticklabels=label_names,
            yticklabels=label_names,
            annot=True, fmt='g')
plt.xlabel('Prediction')
plt.ylabel('Label')
plt.show()
```

Gambar 5. 13 Potongan kode evaluasi model

5) Menyimpan model

Model dengan hasil evaluasi yang baik akan disimpan sehingga dapat digunakan untuk pengenalan ucapan secara *real-time*.

```
class ExportModel(tf.Module):
  def __init__(self, model):
    self.model = model

    # Accept either a string-filename or a batch of waveforms.
    # YOu could add additional signatures for a single wave, or a ragged-batch.
    self.__call__.get_concrete_function(
        x=tf.TensorSpec(shape=(), dtype=tf.string))
    self.__call__.get_concrete_function(
       x=tf.TensorSpec(shape=[None, 16000], dtype=tf.float32))


  @tf.function
  def __call__(self, x):
    # If they pass a string, load the file and decode it.
    if x.dtype == tf.string:
      x = tf.io.read_file(x)
      x, _ = tf.audio.decode_wav(x, desired_channels=1, desired_samples=16000,)
      x = tf.squeeze(x, axis=-1)
      x = x[tf.newaxis, :]
```

```
    x = get_spectrogram(x)
    result = self.model(x, training=False)

    class_ids = tf.argmax(result, axis=-1)
    class_names = tf.gather(label_names, class_ids)
    return {'predictions':result,
            'class_ids': class_ids,
            'class_names': class_names}

export = ExportModel(model)
tf.saved_model.save(export, "LSTM_")
```

Gambar 5. 14 Potongan kode menyimpan model LSTM

Class ExportModel dengan turunan kelas tf.Module digunakan untuk melakukan preprocessing secara langsung untuk data yang ingin diprediksi. Preprocessing ini memudahkan untuk dapat mengubah bentuk data sesuai yang diinginkan hanya dengan memasukkan data sinyal suara pada model.

5.1.5 Implementasi Sistem

1) Fungsi rekam

Untuk mendapatkan input suara dari *microphone* perangkat, python menyediakan library pyaudio. Pyaudio mampu melakukan streaming audio sehingga akan sesuai denggan sistem real-time *speech recognition* yang dibangun. Gambar 5. 15 berikut adalah fungsi bantu untuk sistem dapat menerima inputan suara secara real-time:

```
import pyaudio

FRAMES_PER_BUFFER = 3200
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 16000
p = pyaudio.PyAudio()

def record_audio():
    stream = p.open(
        format=FORMAT,
        channels=CHANNELS,
        rate=RATE,
        input=True,
        frames_per_buffer=FRAMES_PER_BUFFER
    )

    #print("start recording...")

    frames = []
    seconds = 1
    for i in range(0, int(RATE / FRAMES_PER_BUFFER * seconds)):
        data = stream.read(FRAMES_PER_BUFFER)
        frames.append(data)
```

```
    # print("recording stopped")

    stream.stop_stream()
    stream.close()

    return np.frombuffer(b''.join(frames), dtype=np.int16)


def terminate():
    p.terminate()
```

Gambar 5. 15 Potongan kode *recording helper*

Gambar 5. 15 menunjukkan algoritma untuk dapat menerima inputan secara real-time melalui audio streaming. Sinyal suara akan ditangkap sebagai audio buffer dan diubah menjadi sequence angka dengan sample rate 16000 berdurasi 1 detik. Audiobuffer ini kemudian disimpan dalam tipe data int16.

2) *Audiobuffer Preprocessing*

Audiobuffer yang dihasilkan dari input *microphone* belum dapat secara langsung diolah untuk menetukan kata perintah apa yang muncul. Preprocessing diperlukan untuk mengubah audiobuffer menjadi waveform dengan shape dan ukuran data yang sesuai.

```
def preprocess_audiobuffer(waveform):
    #  normalize from [-32768, 32767] to [-1, 1]
    waveform =  waveform / 32768
    waveform = tf.convert_to_tensor(waveform, dtype=tf.float32)
    return waveform
```

Gambar 5. 16 Potongan kode *audiobuffer preprocessing*

Potongan kode pada Gambar 5. 16 melakukan normalisasi pada waveform kemudian mengubah tipedanya menjadi float32. Format bentuk ini telah sesuai pada export model yang ditunjukkan pada Gambar 5. 14.

3) *Speech to command*

Menggunakan pynput, sistem dapat mengontrol keyboard untuk mengendalikan shortcut pada *software* presentasi.

```
from pynput.keyboard import Controller as KeyboardController, KeyCode
from pynput.keyboard import Key

def speech_to_command(key:str, keywords:list, keyboard:object):
    if key in keywords:
        if key == 'right':
            # Press and release the Right arrow key
            keyboard.press(KeyCode.from_vk(0x27))
            keyboard.release(KeyCode.from_vk(0x27))
        elif key == 'left':
            # Press and release the Left arrow key
```

```
            keyboard.press(KeyCode.from_vk(0x25))
            keyboard.release(KeyCode.from_vk(0x25))
        elif key == 'up':
            # Press and release the Up arrow key
            keyboard.press(KeyCode.from_vk(0x26))
            keyboard.release(KeyCode.from_vk(0x26))
        elif key == 'down':
            # Press and release the Down arrow key
            keyboard.press(KeyCode.from_vk(0x28))
            keyboard.release(KeyCode.from_vk(0x28))
        elif key == 'go':
            # Simulate pressing the spacebar key
            keyboard.press(Key.space)
            keyboard.release(Key.space)
```

Gambar 5. 17 Potongan kode *speech to command helper*

4) *Real-time Speech Recognition*

Fungsi-fungsi pendukung diatas kemudian disatukan menjadi rangkaian yang utuh untuk membangun sistem *real-time speech recognition*. Gambar 5. 18 berikut menunjukkan program utama yang dijalankan dengan tampilan pada terminal.

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'  # 0: all messages are logged (default), 1: INFO messages are not printed, 2: INFO and WARNING messages are not printed, 3: INFO, WARNING, and ERROR messages are not printed

MODEL_DIR = "saved_model"
MODEL_NAME = "LSTM_"
loaded_model = tf.saved_model.load(os.path.join(MODEL_DIR, MODEL_NAME))

def predict_mic():
    audio = record_audio()
    waveform = preprocess_audiobuffer(audio)[tf.newaxis, :]

    # Set the confidence threshold
    threshold = 0.7
    prediction = loaded_model(waveform)['predictions']
    confidence = tf.reduce_max(prediction).numpy()
    if confidence >= threshold:
        command = loaded_model(waveform)['class_names'].numpy()[0].decode('utf-8')
        return command, confidence
    else : return None, None

if __name__ == "__main__":
    # from turtle_helper import move_turtle
    print("===REAL-TIME SPEECH RECOGNITION==")
    print("Say ON to start the app")
    print()
    keyboard = KeyboardController()
    commands = ['down','go','left','off','on','right','stop','up']

    app_state = True
    speech_state = False
    while app_state:
        command, confidence = predict_mic()
        if (command == 'on') and (speech_state == False):
            speech_state = True
            print("The APP is ON")
```

```
            print("We Are hearing You right now!")
            print("====================================")
            print()
        if (command == 'off') and (speech_state == True):
            speech_state = False
            print("The APP is OFF")
            print("Say ON so we can hear your command!")
            print("====================================")
            print()
        if speech_state:
            if command != None:
                speech_to_command(command, commands, keyboard)
                print("Predicted label:", command)
                print("Confidence:", confidence)
                print()
                if command == "stop":
                    terminate()
                    app_state = False
exit()
```

Gambar 5. 18 Kode utama sistem *real-time speech recognition*

Gambar 5. 18 menjadi potongan kode yang utama dengan menggabungkan beberapa helper function lainnya. Fungsi predict_mic berguna untuk memprediksi kata kunci yang muncul pada ucapan berdurasi 1 detik. Nilai ambang batas (*threshold*) untuk *confidence* sebesar 0,7 ditentukan untuk mencegah sistem unutk tidak memprediksi ucapan yang selain 8 kata kunci yang tersedia. Semakin besar nilai *confidence* yang dihasilkan, maka semakin tepat kata kunci yang dikenali.

5.2 Hasil

5.2.1 Data Splitting

Kumpulan file suara yang berisi ucapan kata kunci ini dapat dibaca menggunakan audio_dataset_from_directory. Satu direktori ini dikhususkan untuk menampung folder-folder file ucapan.

```
Found 30561 files belonging to 8 classes.
Using 24449 files for training.
Using 6112 files for validation.
```

Gambar 5. 19 Hasil pembacaan file audio pada direktori

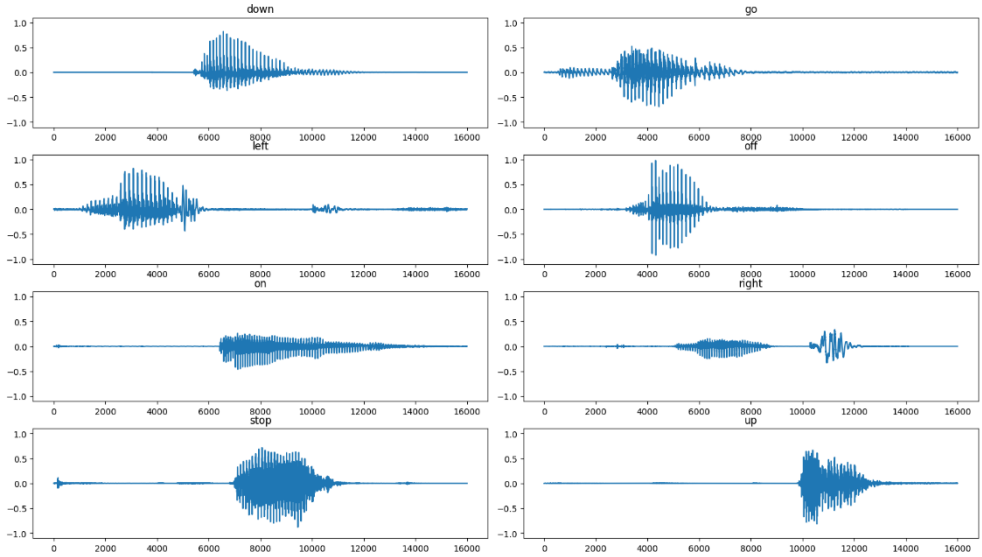
Gambar 5. 19 menunjukkan bahwa terdapat 30561 file suara dengan tersedia 8 kelas (kata kunci). Dengan *validation split rate* sebesar 0,2 didapatkan 24449 file untuk training, dan 6112 untuk validasi. Data testing diperoleh dari pembagian data validasi sebesar 50% menggunakan fungsi *shard*. Tabel 5. 3 menunjukkan pembagian data secara lengkap.

Tabel 5. 3 Pemisahan dataset

Pemisahan data	Jumlah data
Training	24449
Validation	3056
Testing	3056

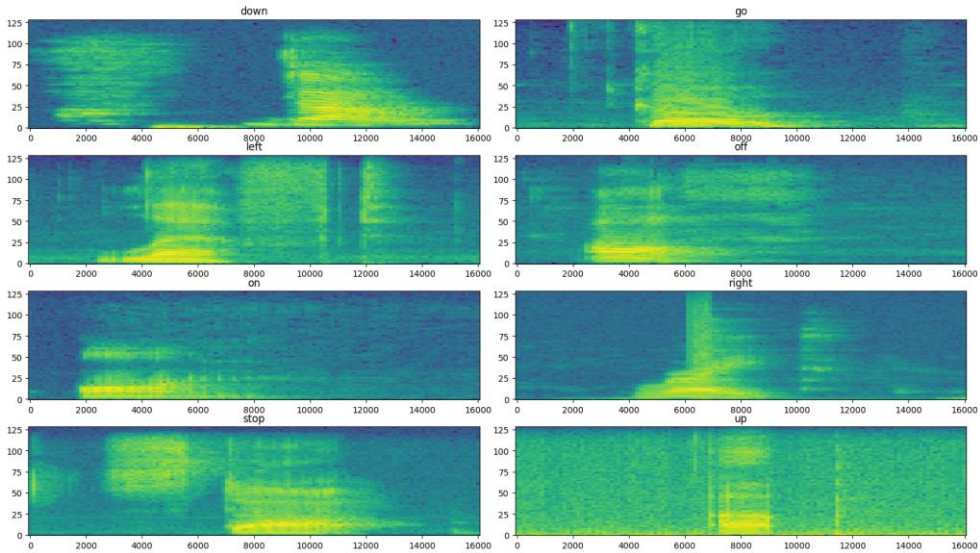
5.2.2 Visualisasi Data

1) *Waveform*



Gambar 5. 20 Visualisasi *waveform*

2) *Spectrogram*



Gambar 5. 21 Visualisasi *spectrogram*

5.2.3 Hasil *summary* model

Tabel 5. 4 *Model summaries*

Nama model	Layers	Unit	*Output shape*	*Param*	*Total params*
LSTM_1	LSTM	128	(None, 124, 128)	132096	593.416
	LSTM	64	(None, 64)	49408	
	Dense	6	(None, 8)	520	
LSTM_2	LSTM	128	(None, 124, 128)	132096	593.416
	Dropout	-	(None, 124, 128)	0	
	LSTM	64	(None, 128)	49408	
	Dense	6	(None, 8)	520	
LSTM_3	LSTM	256	(None, 124, 256)	395264	593.416
	LSTM	128	(None, 128)	197120	
	Dense	6	(None, 8)	1032	
LSTM_4	LSTM	256	(None, 124, 256)	395264	593.416
	Dropout	-	(None, 124, 256)	0	
	LSTM	128	(None, 128)	197120	
	Dense	6	(None, 8)	1032	

5.2.4 Hasil *fitting* model

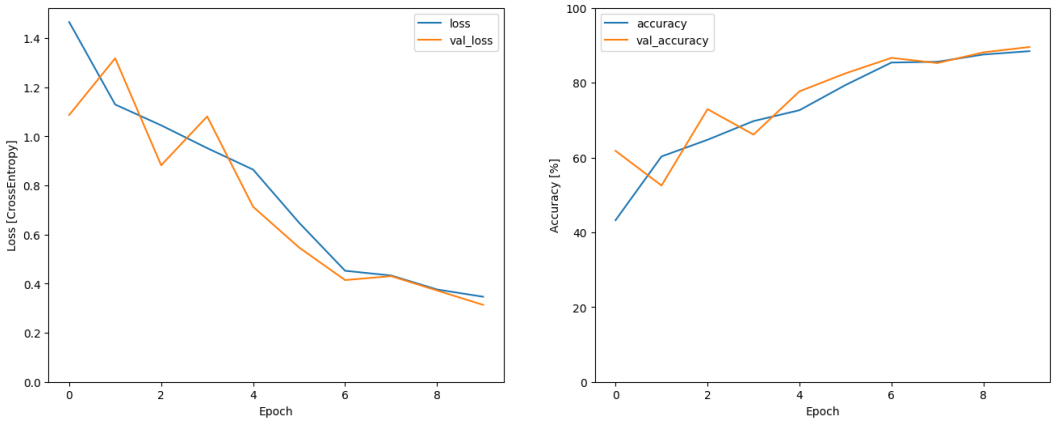
Tabel 5. 5 Hasil *fitting* model

Nama Model	Epochs	Waktu Epoch	Total waktu	Training		Validation		Testing	
				Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
LSTM_1	1	425s	80m 47.3s	1.4648	0.4324	1.0864	0.6178	0.3188	0.8896
	2	402s		1.1291	0.6028	1.3173	0.5250		
	3	419s		1.0443	0.6474	0.8818	0.7293		
	4	401s		0.9516	0.6974	1.0804	0.6612		
	5	578s		0.8640	0.7265	0.7126	0.7770		
	6	682s		0.6478	0.7938	0.5468	0.8250		
	7	589s		0.4522	0.8539	0.4144	0.8664		
	8	466s		0.4330	0.8560	0.4305	0.8526		
	9	420s		0.3761	0.8754	0.3719	0.8813		
	10	465s		0.3466	0.8844	0.3136	0.8954		
LSTM_2	1	953s	106m 37.7s	1.5606	0.3823	1.2370	0.5474	0.3889	0.8691
	2	707s		1.2790	0.5489	1.3920	0.5319		
	3	591s		1.1798	0.6014	0.9644	0.6724		
	4	586s		0.8267	0.7401	0.7315	0.7743		
	5	605s		0.8089	0.7511	0.8214	0.7477		
	6	581s		0.6507	0.7980	0.5486	0.8211		
	7	586s		0.5171	0.8400	0.4849	0.8457		
	8	587s		0.4820	0.8498	0.4542	0.8562		
	9	604s		0.5137	0.8422	0.6807	0.8000		
	10	596s		0.4438	0.8616	0.3717	0.8793		
LSTM_3	1	2151s	172m 3.9s	1.6440	0.3636	1.5287	0.4211	0.2528	0.9167
	2	912s		1.5990	0.4120	1.7188	0.3559		
	3	881s		1.2692	0.5473	0.9816	0.6882		

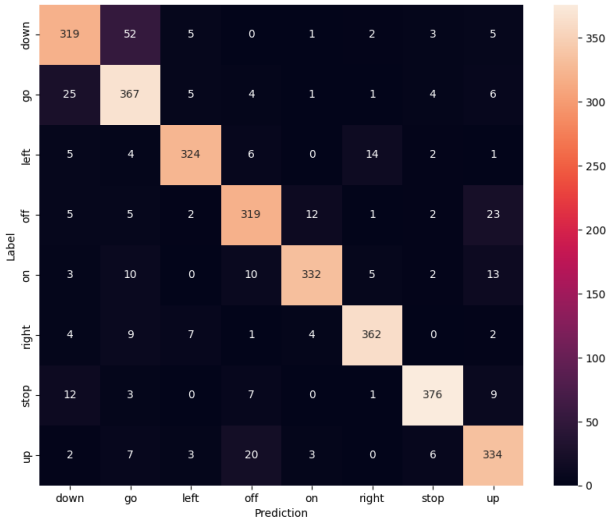
	4	896s		0.6599	0.7888	0.5501	0.8188		
	5	912s		0.4690	0.8469	0.4044	0.8714		
	6	912s		0.3492	0.8840	0.3019	0.9010		
	7	907s		0.2795	0.9069	0.2584	0.9125		
	8	912s		0.2466	0.9176	0.2447	0.9171		
	9	919s		0.2003	0.9327	0.2458	0.9168		
	10	921s		0.1850	0.9375	0.2286	0.9243		
LSTM_4	1	833s	143m 45.2s	1.7593	0.3005	1.6986	0.3303	0.1883	0.9398
	2	849s		1.6498	0.3642	1.2970	0.5382		
	3	847s		0.8801	0.7051	0.5215	0.8339		
	4	866s		0.5934	0.8084	0.4068	0.8599		
	5	858s		0.3880	0.8710	0.3571	0.8868		
	6	870s		0.3059	0.8989	0.3004	0.9000		
	7	879s		0.2550	0.9157	0.2254	0.9237		
	8	879s		0.2228	0.9245	0.2338	0.9234		
	9	867s		0.1858	0.9360	0.2162	0.9306		
	10	876s		0.1704	0.9418	0.1752	0.9434		

5.2.5 Evaluasi Model

1) LSTM_1

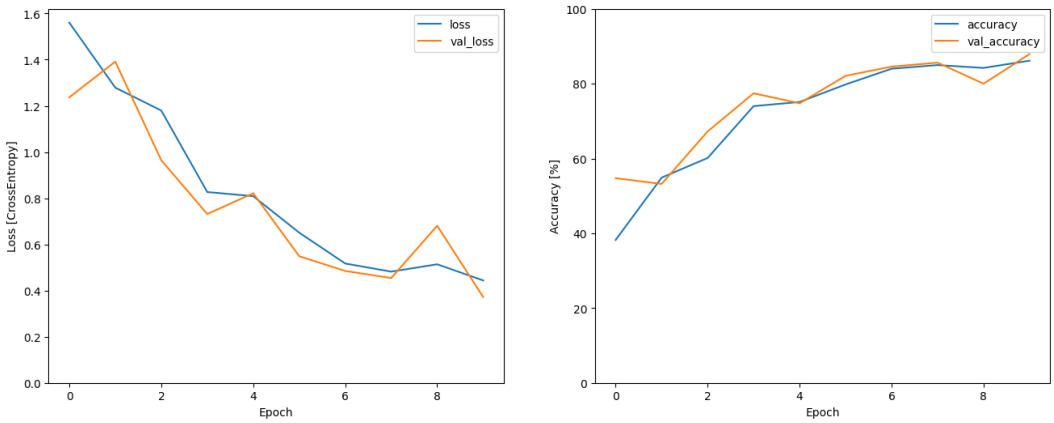


Gambar 5. 22 Riwayat *fitting* LSTM_1

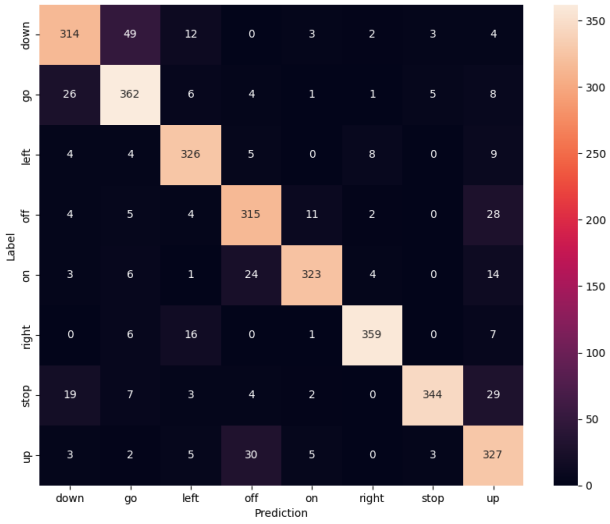


Gambar 5. 23 *Confusion matrix* LSTM_1

2) LSTM_2

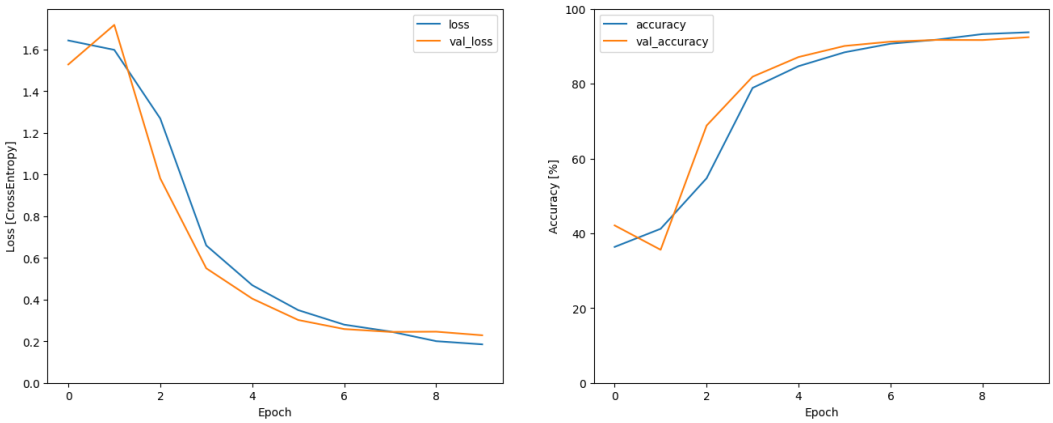


Gambar 5. 24 Riwayat *fitting* LSTM_2

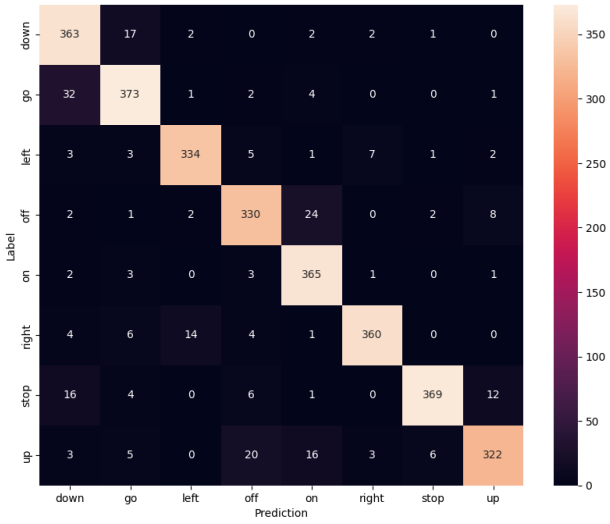


Gambar 5. 25 *Confusion matrix* LSTM_2

3) LSTM_3

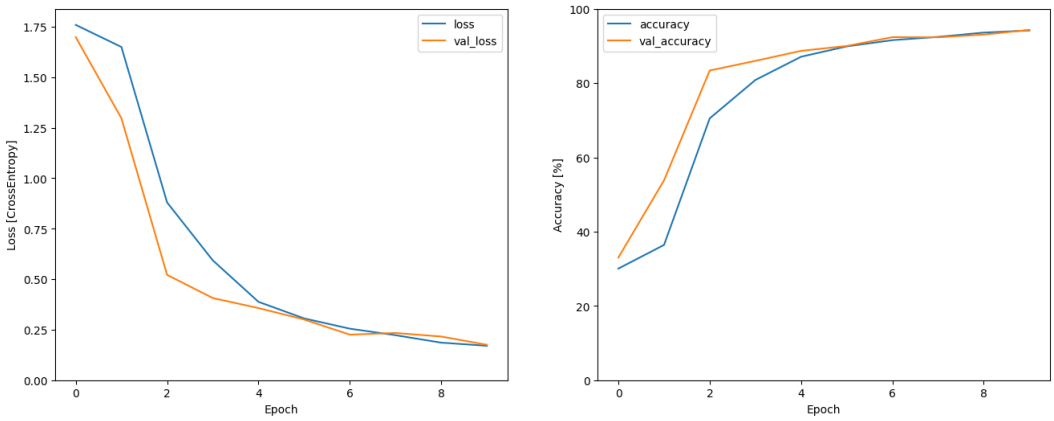


Gambar 5. 26 Riwayat *fitting* LSTM_3

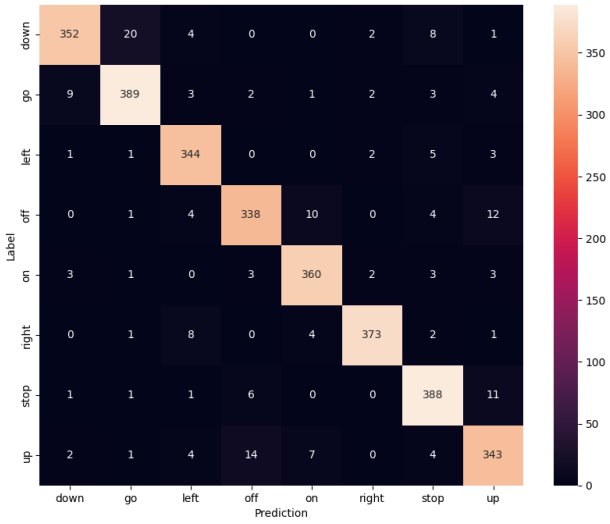


Gambar 5. 27 *Confusion matrix* LSTM_3

4) LSTM_4



Gambar 5. 28 Riwayat *fitting* LSTM_4



Gambar 5. 29 *Confusion matrix* LSTM_4

5.2.6 Sistem *Real-time Speech Recognition*

```
The APP is ON
We Are hearing You right now!
==================================

Predicted label: on
Confidence: 0.99679

Predicted label: up
Confidence: 0.98637354

Predicted label: go
Confidence: 0.9742933

Predicted label: down
Confidence: 0.95447415

Predicted label: left
Confidence: 0.99756026

Predicted label: right
Confidence: 0.93937904

The APP is OFF
Say ON so we can hear your command!
==================================

The APP is ON
We Are hearing You right now!
==================================

Predicted label: on
Confidence: 0.9728793

Predicted label: stop
Confidence: 0.7927887

==================================
The APP has Stopped!
Thank you for using me!
```

Gambar 5. 30 Hasil eksekusi sistem pengenalan suara pada terminal

5.3 Pembahasan

Tabel 5. 5 menunjukkan hasil eksperimen yang dilakukan ketika membangun model *deep learning* menggunakan LSTM sebagai layer-nya. Penelitian ini mengujikan 4 model dengan struktur yang berbeda dari jumlah unit yang digunakan dan penggunaan *dropout regularization*. Konfigurasi *compile* model dan epoch ditentukan dengan jumlah yang sama.

Pada model LSTM_1 dan LSTM_2 dibuat model menggunakan 2 layer LSTM. Layer LSTM pertama berisi 128 unit dan layer LSTM kedua berisi 64 unit. LSTM_2 memiliki perbedaan yaitu terdapat tambahan layer *Dropout* sebagai regularisasi yang diletakkan di antara kedua layer LSTM. Hasil menunjukkan LSTM_1 lebih baik dari LSTM_2 dengan mengungguli total waktu *fitting* selama 80 menit 47,3 detik dibandingkan 106 menit 37,7 detik. Nilai loss yang dihasilkan LSTM_1 lebih kecil, yaitu 0,3466 untuk training, 0,3136 untuk validation, dan 0,3188 untuk testing. Akurasi yang dihasilkan LSTM menugguli sekitar 2% untuk setiap datanya, dengan rincian 88,44% untuk training, 89,54% untuk validation, dan 88,96% untuk testing. Penggunaan dropout layer pada LSTM_2 memengaruhi hasil modelnya, sehingga regularisasi tidak diperlukan untuk struktur model tersebut.

LSTM_3 dan LSTM_4 memiliki aturan pengujian yang sama seperti 2 model pendahulunya. Jumlah unit menjadi pembeda dengan layer pertama berjumlah 256 unit dan kedua 128 unit. Hasil eksperimen menunjukkan LSTM_4 dengan *dropout layer* menungguli LSTM_3. LSTM_4 memakan waktu *fitting* selama 143 menit 45,2 detik, unggul sekitar 30 menit dari LSTM_3. Performa model LSTM_4 sedikit mengungguli dengan nilai loss 0,1704 untuk training, 0,1752 untuk validation, dan 0,1883 untuk testing. Akurasi yang dihasilkan LSTM_4 menungguli model-model terdahulunya, yaitu memiliki akurasi 94,18% untuk training, 94,34% untuk validation, dan 98,98% untuk testing.

Hasil eksperimen 4 model dengan struktur yang berbeda menunjukkan bahwa LSTM_4 jauh mengungguli performa LSTM_1 dan LSTM_2, serta sedikit lebih baik daripada LSTM_3. Jumlah unit dan penggunaan layer dropout terbukti berpengaruh terhadap performa dan lama *fitting* model.

Regularisasi dropout hanya dapat berpengaruh baik untuk model dengan jumlah unit yang banyak untuk mencegah terjadinya overfitting sebagai akibat terlalu banyaknya unit yang terhubung. *Fully connected layer* menyebabkan pelatihan yang berlangsung menjadi lebih lama dan cenderung menyebabkan *overfit*. Namun, hasil eksperimen dengan dropout tidak selalu dapat ditambahkan untuk segala jenis struktur model. LSTM_4 dapat dengan baik menggunakan regularisasi dropout sehingga performa modelnya dapat jauh mengungguli model lain.

BAB VI
PENUTUP

6.1 Simpulan

Penelitian yang telah dilakukan tentang **penerapan speech recognition menggunakan LSTM untuk presentasi dinamis** dihasilkan kesimpulan sebagai berikut:

1. Sistem mampu mendeteksi inputan suara kata perintah melalui *microphone*;
2. Empat model eksperimen menghasilkan 1 model terbaik, yaitu LSTM_4 dengan nilai loss 0.1704 untuk training, 0.1752 untuk validation, dan 0.1883 untuk testing. Akurasi yang dihasilkan model ini adalah 94.18% untuk training, 94.34% untuk validation, dan 93.98% untuk testing;
3. Model LSTM_4 dapat digunakan dengan baik untuk diterapkan pada sistem pengenalan ucapan;
4. Jumlah unit dan penggunaan dropout layer berpengaruh pada performa model;
5. Sistem pengenalan ucapan dapat digunakan dengan baik sebagai perintah untuk mengoperasikan *software* presentasi.

6.2 Saran

Penelitian ini masih dapat dikembangkan dalam hal berikut:

1. Membandingkan metode LSTM dengan metode *machine learning* yang lain;
2. Penggunaan perangkat untuk pelatihan model dan pembuatan sistem disarankan untuk menggunakan GPU (*Graphical Processing Unit*) untuk mencapai waktu latih yang lebih cepat dan mampu menambah kosa kata perintah;
3. Membuat *user interface* yang sesuai dengan rancangan karena saat ini tampilan hanya berupa terminal.

DAFTAR PUSTAKA

Arief, R., Iriawan, N. A., & Lawi, A. (2021). Klasifikasi Audio Ucapan Emosional Menggunakan Model LSTM. *Konferensi Nasional Ilmu Komputer (KONIK)*, 524–529.

Atcheson, M., Sethu, V., & Epps, J. (2019). Using Gaussian Processes with LSTM Neural Networks to Predict Continuous-Time, Dimensional Emotion in Ambiguous Speech. *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, 718–724. https://doi.org/10.1109/ACII.2019.8925450

Cahuantzi, R., Chen, X., & Güttel, S. (2021). *A comparison of LSTM and GRU networks for learning symbolic sequences*.

Fendji, J. L. K. E., Tala, D. C. M., Yenke, B. O., & Atemkeng, M. (2022). Automatic Speech Recognition Using Limited Vocabulary: A Survey. *Applied Artificial Intelligence*, *36*(1). https://doi.org/10.1080/08839514.2022.2095039

Jabir, B., & Falih, N. (2021). Dropout, a basic and effective regularization method for a deep learning model: a case study. *Indonesian Journal of Electrical Engineering and Computer Science*, *24*(2), 1009. https://doi.org/10.11591/ijeecs.v24.i2.pp1009-1016

Kamath, U., Liu, J., & Whitaker, J. (2019). *Deep Learning for NLP and Speech Recognition*. Springer International Publishing. https://doi.org/10.1007/978-3-030-14596-5

Lisnawati, I., & Ertinawati, Y. (2019). Literat Presentasi. *Jurnal Metaedukasi*, *1*(1), 1–12.

Oruh, J., Viriri, S., & Adegun, A. (2022). Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition. *IEEE Access*, *10*, 30069–30079. https://doi.org/10.1109/ACCESS.2022.3159339

Pricillia, T., & Zulfachmi. (2021). Perbandingan Metode Pengembangan Perangkat Lunak (Waterfall, Prototype, RAD). *Jurnal Bangkit Indonesia*, *10*(1), 6–12. https://doi.org/10.52771/bangkitindonesia.v10i1.153

Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. *Neurocomputing*, *323*, 203–213. https://doi.org/10.1016/j.neucom.2018.09.082

Sen, S., Dutta, A., & Dey, N. (2019). *Audio Processing and Speech Recognition*. Springer Singapore. https://doi.org/10.1007/978-981-13-6098-5

Warden, P. (2018). *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition*.

Xiang, L., Lu, S., Wang, X., Liu, H., Pang, W., & Yu, H. (2019). Implementation of LSTM Accelerator for Speech Keywords Recognition. *2019 IEEE 4th International Conference on Integrated Circuits and Microsystems (ICICM)*, 195–198. https://doi.org/10.1109/ICICM48536.2019.8977176