

**AUTOMATIC SPEECH RECOGNITION BAHASA INDONESIA
MENGUNAKAN BIDIRECTIONAL LONG SHORT-TERM
MEMORY DAN CONNECTIONIST TEMPORAL
CLASSIFICATION**

SKRIPSI

RUSWAN EFENDI

141402085



**PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA**

MEDAN

2019

AUTOMATIC SPEECH RECOGNITION BAHASA INDONESIA
MENGUNAKAN BIDIRECTIONAL LONG SHORT-TERM
MEMORY DAN CONNECTIONIST TEMPORAL
CLASSIFICATION

SKRIPSI

Diajukan untuk melengkapi tugas dan memenuhi syarat memperoleh ijazah Sarjana
Komputer

RUSWAN EFENDI

141402085



PROGRAM STUDI S1 TEKNOLOGI INFORMASI
FAKULTAS ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

MEDAN

2019

PERSETUJUAN

Judul : AUTOMATIC SPEECH RECOGNITION BAHASA
INDONESIA MENGGUNAKAN BIDIRECTIONAL
LONG SHORT-TERM MEMORY DAN
CONNECTIONIST TEMPORAL CLASSIFICATION

Kategori : SKRIPSI

Nama : RUSWAN EFENDI

Nomor Induk Mahasiswa : 141402085

Program Studi : TEKNOLOGI INFORMASI

Fakultas : ILMU KOMPUTER DAN TEKNOLOGI INFORMASI
UNIVERSITAS SUMATERA UTARA

Komisi Pembimbing :

Pembimbing 2



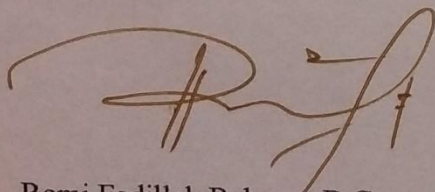
Baihaqi Siregar, S.Si., MT.
NIP. 19790108 201212 1 002

Pembimbing 1



Ulfi Andayani, S.Kom., M.Kom
NIP. 19860419 201504 2 004

Diketahui/disetujui oleh
Program Studi S1 Teknologi Informasi
Ketua,



Romi Fadillah Rahmat, B.Comp.Sc., M.Sc.
NIP. 19860303 201012 1 004

PERNYATAAN

AUTOMATIC SPEECH RECOGNITION BAHASA INDONESIA MENGUNAKAN BIDIRECTIONAL LONG SHORT-TERM MEMORY DAN CONNECTIONIST TEMPORAL CLASSIFICATION

SKRIPSI

Saya mengakui bahwa skripsi ini adalah hasil karya saya sendiri, kecuali beberapa kutipan dan ringkasan yang masing-masing telah disebutkan sumbernya.

Medan, 26 April 2019

Ruswan Efendi
141402085

UCAPAN TERIMA KASIH

Puji dan syukur penulis panjatkan kepada Tuhan Yesus Kristus yang telah memberikan kekuatan, kesehatan, dan tuntunan kepada penulis sehingga penulis dapat mengerjakan dan menyelesaikan skripsi ini sebagai syarat untuk memperoleh Sarjana Komputer di Universitas Sumatera Utara.

Penulis ingin mengucapkan terima kasih kepada Ibu Ulfi Andayani, S.Kom., M.Kom selaku dosen pembimbing pertama dan Bapak Baihaqi Siregar, S.Si., MT. selaku dosen pembimbing kedua yang telah meluangkan waktu untuk membimbing penulis dalam mengerjakan skripsi ini. Penulis juga ingin berterima kasih kepada Bapak Prof. Dr. Drs. Opim Salim Sitompul, M.Sc sebagai dosen penguji pertama dan Bapak Muhammad Anggia Muchtar, ST., MMIT sebagai dosen penguji kedua yang telah memberikan kritik dan masukan kepada penulis sehingga dapat menyelesaikan skripsi ini dengan baik.

Penulis juga ingin berterima kasih kepada Ketua dan Sekretaris Program Studi Teknologi Informasi, Dekan dan Wakil Dekan Fakultas Ilmu Komputer dan Teknologi Informasi, dan semua dosen dan pegawai yang telah membimbing dan membantu penulis selama masa perkuliahan.

Terima kasih juga penulis sampaikan kepada orangtua penulis, Tjia Dju Hin dan Lie Ling Ling, yang telah membesarkan penulis dengan cinta kasih, serta doa dan bantuan dari mereka yang selalu menyertai penulis selama ini. Penulis juga ingin berterima kasih kepada saudara penulis, Eric Suwarno dan Sherly Natalia, yang telah memberikan kekuatan dan dukungan kepada penulis dalam mengerjakan skripsi ini. Penulis tidak lupa berterima kasih kepada pihak berikut yang telah membantu penulis dalam menyelesaikan skripsi ini.

1. *Speech Resources Consortium of National Institute of Informatics* dan Ibu Marika Horiuchi yang telah membantu penulis dalam mengajukan permintaan penggunaan *speech corpus TITML-IDN*.
2. Bibi Li Chen, kakak Lilis Suryani, Tju Li Ni, dan keluarga lainnya yang telah memberikan bantuan dan dukungan kepada penulis dalam proses penyelesaian skripsi ini.

3. Teman-teman penulis yang selalu mendukung dan memberi semangat dalam proses pengerjaan skripsi ini, yaitu Alex Wijaya, Wendy Winata, Victoria Lonita Christy Tampubolon, Varuna Dewi, Cindy Pakpahan, Tama Loy Dennis Munthe, Esya Mahabbah, serta seluruh abang/kakak dan adik pada angkatan 2012 hingga 2018.
4. *zisseL*, *Stef*, *Selvi*, *Sheep*, *Spooder*, Eric, dan para *[DAD]* lainnya yang telah mendukung penulis dalam menyelesaikan skripsi ini.
5. Pihak yang terlibat secara langsung maupun tidak langsung yang tidak dapat dituliskan satu persatu.

Akhir kata, penulis berharap skripsi ini dapat berguna dan memberikan manfaat kepada pembaca.

Penulis,

Ruswan Efendi

ABSTRAK

Automatic speech recognition adalah sebuah proses untuk mengubah *input* dari bentuk suara ke dalam bentuk teks untuk mempermudah interaksi manusia dengan komputer. Pada umumnya, *Hidden Markov Model* digunakan dalam melakukan *automatic speech recognition*. Namun, *Hidden Markov Model* memiliki masalah dalam waktu yang dibutuhkan untuk melakukan pemrosesan data, khususnya pada proses segmentasi dan *tuning*. Oleh karena itu, dibutuhkan suatu metode yang dapat memperbaiki masalah waktu pemrosesan data pada *Hidden Markov Model*. Metode yang digunakan untuk melakukan *automatic speech recognition* adalah *bidirectional Long Short-Term Memory (bidirectional LSTM)* untuk proses segmentasi pada sinyal suara dan *Connectionist Temporal Classification (CTC)* untuk menerjemahkan hasil pengolahan yang dilakukan menggunakan jaringan *bidirectional LSTM* menjadi data teks. Data yang digunakan pada penelitian ini adalah *speech corpus* bahasa Indonesia dari *Tokyo Institute of Technology Multilingual Speech Corpus (TITML)* dan dataset tambahan. Hasil penelitian menunjukkan bahwa jumlah neuron dan *learning rate* yang digunakan dapat memengaruhi tingkat akurasi pengenalan ucapan. Hal ini dapat diketahui dari hasil percobaan yang dilakukan pada tiga *dataset* yang direkam pada kondisi yang berbeda. Nilai *WER (Word Error Rate)* pada *dataset testing* dari TITML adalah 3,01 %, sedangkan percobaan pada lingkungan hening menggunakan perangkat yang berbeda memberikan nilai *WER* sebesar 20,87 %. Di sisi lain, percobaan pada *dataset* dengan suara latar belakang menggunakan perangkat yang berbeda memberikan hasil pengenalan ucapan dengan nilai *WER* sebesar 75,08 %.

Kata kunci : pengenalan ucapan otomatis bahasa Indonesia, *bidirectional Long Short-Term Memory*, *Connectionist Temporal Classification*, interaksi manusia dan komputer

INDONESIAN AUTOMATIC SPEECH RECOGNITION WITH BIDIRECTIONAL LONG SHORT-TERM MEMORY AND CONNECTIONIST TEMPORAL CLASSIFICATION

ABSTRACT

Automatic speech recognition refers to a process of converting an input in form of audio signal into text for better human-computer interaction activity. Automatic speech recognition is typically done by using Hidden Markov Model. Despite of its common usage on automatic speech recognition, a processing duration problem is found on Hidden Markov Model implementation, especially for the segmentation and tuning process. Therefore, a method has to be implemented in order to resolve the processing duration problem on Hidden Markov Model. In this research, automatic speech recognition process is done by using bidirectional Long Short-Term Memory for audio signal segmentation and Connectionist Temporal Classification to convert the result of bidirectional Long Short-Term Memory into readable text. Tokyo Institute of Technology Multilingual (TITML) speech corpus and additional dataset was used to train the network. The result shows that the number of neuron and learning rate used in the process could affect the error rate of the speech recognition process. The experiment returns Word Error Rate (WER) value of 3.01 % on testing dataset, 24.49 % on speech recorded on quiet environment with different microphone, and 75.08 % on speech recorded on noisy environment.

Keywords : Indonesian automatic speech recognition, bidirectional Long Short-Term Memory, Connectionist Temporal Classification, human-computer interaction

DAFTAR ISI

	Hlm.
PERSETUJUAN	ii
PERNYATAAN	iii
UCAPAN TERIMA KASIH	iv
ABSTRAK	vi
ABSTRACT	vii
DAFTAR ISI	viii
DAFTAR TABEL	x
DAFTAR GAMBAR	xi
BAB 1 PENDAHULUAN	1
1.1. Latar Belakang	1
1.2. Rumusan Masalah	3
1.3. Tujuan Penelitian	3
1.4. Batasan Masalah	3
1.5. Manfaat Penelitian	3
1.6. Metode Penelitian	4
1.7. Sistematika Penulisan	4
BAB 2 LANDASAN TEORI	6
2.1. <i>Digital Signal Processing</i>	6
2.2. <i>Mel Frequency Cepstral Coefficients (MFCC)</i>	7
2.2.1. Preemphasis	7
2.2.2. Framing dan windowing	8
2.2.3. Fast Fourier Transform	8
2.2.4. Mel filter bank	9
2.2.5. Discrete Cosine Transform (DCT)	10
2.3. <i>Artificial Neural Network</i>	10
2.4. <i>Recurrent Neural Network</i>	13
2.5. <i>Long Short Term Memory</i>	14
2.5.1. Input Gates	15
2.5.2. Forget Gates	15
2.5.3. Cells	16

2.5.4. Output Gates	16
2.5.5. Cell Outputs	16
2.6. <i>Early Stopping</i>	16
2.7. <i>Language Model</i>	16
2.8. <i>Connectionist Temporal Classification</i>	18
2.9. Penelitian Terdahulu	21
BAB 3 ANALISIS DAN PERANCANGAN SISTEM	25
3.1. <i>Dataset</i>	25
3.2. Arsitektur Umum	26
3.3. Augmentasi Suara	27
3.4. <i>Preprocessing</i>	27
3.4.1. Preemphasis	27
3.4.2. Framing and Windowing	28
3.4.3. Fast Fourier Transform (FFT)	29
3.4.4. Mel filter bank	30
3.4.5. Discrete Cosine Transform	32
3.5. Implementasi Jaringan <i>Bidirectional LSTM</i>	33
3.6. <i>Connectionist Temporal Classification</i>	34
3.7. Perancangan Sistem	36
3.7.1. Perancangan sistem training	36
3.7.2. Perancangan aplikasi pengguna	37
BAB 4 IMPLEMENTASI DAN PENGUJIAN	40
4.1. Spesifikasi Perangkat Keras dan Perangkat Lunak	40
4.2. Implementasi Antarmuka Aplikasi	41
4.3. Pembuatan <i>Language Model</i>	42
4.4. Pelatihan Sistem	44
4.5. Pengujian Sistem	50
4.5.1. Pengujian sistem pada dataset tambahan	52
BAB 5 KESIMPULAN	54
5.1. Kesimpulan	54
5.2. Saran	55
DAFTAR PUSTAKA	56

DAFTAR TABEL

	Hlm.
Tabel 2.1 Penelitian Terdahulu	22
Tabel 3.1 Contoh Hasil <i>Preemphasis</i> pada <i>MFCC</i>	27
Tabel 3.2 Hasil <i>framing</i> dan <i>windowing</i> pada <i>MFCC</i>	29
Tabel 3.3 Nilai untuk pembuatan <i>filter bank</i>	31
Tabel 3.4 Contoh Output <i>Softmax Layer</i> untuk Setiap Waktu	35
Tabel 4.1 Parameter Pelatihan Jaringan <i>Bidirectional LSTM</i>	44
Tabel 4.2 Hasil Proses <i>Training</i> untuk Setiap Percobaan	46
Tabel 4.3 Lima Hasil Pengenalan Ucapan Terburuk pada Percobaan C	52
Tabel 4.4 Hasil Pengujian pada Lingkungan yang Berbeda	53

DAFTAR GAMBAR

Hlm.

Gambar 2.1. Skema <i>digital signal processing</i> (Oshana, 2012).....	6
Gambar 2.2. Diagram Blok <i>Mel Frequency Cepstral Coefficients</i>	7
Gambar 2.3. <i>Fast Fourier Transform</i> (Castillo, 2017)	8
Gambar 2.4. Struktur sel saraf (Soares & Souza, 2016)	11
Gambar 2.5 Sebuah <i>neuron</i> pada <i>artificial neural network</i> (Negnevitsky, 2005).....	11
Gambar 2.6 Fungsi aktivasi pada sebuah <i>neuron</i> (Negnevitsky, 2005).....	12
Gambar 2.7. Contoh <i>feed-forward network</i> (Soares & Souza, 2016).....	13
Gambar 2.8. Contoh <i>struktur recurrent network</i> (Servan-Schreiber <i>et al.</i> , 1991).....	13
Gambar 2.9 <i>Recurrent neural network</i> dan <i>bidirectional RNN</i> (Graves, Alex, 2013)	14
Gambar 2.10 Satu blok <i>LSTM</i> memiliki 4 <i>layer</i> (Olah, 2015).....	15
Gambar 2.11 <i>Prefix search decoding</i> untuk label X, Y (Graves, Alex <i>et al.</i> , 2006) ...	19
Gambar 2.12 <i>Pseudocode prefix beam search</i> (Hannun, A. Y. <i>et al.</i> , 2014).....	20
Gambar 3.1 Arsitektur Umum	26
Gambar 3.2 <i>Filter bank MFCC</i>	32
Gambar 3.3 Arsitektur jaringan	34
Gambar 3.4 Tampilan <i>Tensorboard</i> pada saat melakukan <i>training</i>	36
Gambar 3.5 <i>Logs</i> yang disediakan dan ditampilkan ke <i>terminal</i>	37
Gambar 3.6 Rancangan tampilan aplikasi pengguna	38
Gambar 4.1 Tampilan antarmuka aplikasi	41
Gambar 4.2 Pembuatan <i>language model</i> menggunakan <i>KenLM</i>	43
Gambar 4.3 Hasil pembuatan <i>language model</i> menggunakan <i>KenLM</i>	43
Gambar 4.4 Konversi <i>language model</i> ke dalam bentuk <i>file binary</i>	44
Gambar 4.5 Contoh <i>early stopping</i> pada percobaan A	45
Gambar 4.6 Perbandingan <i>loss</i> pada percobaan.....	47
Gambar 4.7 Grafik perbandingan <i>loss training</i> dan <i>validation</i> setiap percobaan.....	50
Gambar 4.8 Grafik <i>WER</i> (A) dan <i>testing loss</i> (B) pada percobaan C	51

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Automatic speech recognition adalah proses mengubah *input* dalam bentuk suara ke dalam bentuk teks. *Automatic speech recognition* merupakan bagian dalam bidang *pattern recognition* dan *neural language processing*. Melalui *automatic speech recognition*, fitur yang diperoleh dari *input* berupa suara akan dipecah untuk menghasilkan teks. *Automatic speech recognition* umumnya diimplementasikan untuk meningkatkan interaksi manusia dengan komputer dan memudahkan penggunaan suatu aplikasi dengan memberikan perintah menggunakan suara pengguna. Beberapa penelitian yang telah menerapkan *automatic speech recognition* untuk interaksi manusia dengan komputer adalah robot CleverNAO, yang menggunakan *.NET speech library* untuk berinteraksi dengan manusia (Serrano *et al.*, 2015) dan *bot* yang dapat digunakan untuk melakukan pencarian berdasarkan kata kunci dengan menggunakan *class SpeechRecognizer* pada *Android* (Naveen Kumar *et al.*, 2017).

Implementasi *automatic speech recognition* pertama dilakukan pada tahun 1952 oleh *AT&T Bell Laboratories* untuk mengenali angka atau perintah yang diucapkan dengan satu jenis suara dengan melakukan pencocokan pola pada suatu suara (Davis, K. H. *et al.*, 1952). Salah satu contoh implementasi *automatic speech recognition* pada masa tahun 1950 adalah verifikasi nomor kartu kredit oleh pedagang dengan mengucapkan angka pada kartu kredit secara perlahan untuk melakukan suatu transaksi.

Pada era tahun 1970, *dynamic time warping* digunakan untuk mengubah suara menjadi teks (O'Shaughnessy, 2008). Metode *dynamic time warping* menghitung jarak optimal antara dua seri waktu dan meminimalisir jarak antara suara dengan *template* yang telah disediakan dengan memanfaatkan *dynamic programming*. Namun, *dynamic time warping* tidak cocok digunakan untuk melakukan *automatic speech recognition* karena sinyal suara bersifat *non-stasioner* (berubah seiring waktu).

Metode *hidden markov model* mulai digunakan untuk *automatic speech recognition* pada tahun 1975 (O'Shaughnessy, 2008). *Hidden markov model* mengenali suara dengan memilih hasil dengan peluang kecocokan suara dengan *phone* dan peluang kata tersebut berdasarkan *language model*. Metode *Hidden markov model* dapat memberikan hasil lebih baik dibandingkan dengan metode *dynamic time warping* karena sinyal suara bersifat non stasioner (Sajjan & Vijaya, 2012). Salah satu kekurangan *hidden markov model* adalah membutuhkan proses segmentasi dan *tuning* suara untuk transkrip suara yang ingin digunakan yang membutuhkan proses yang panjang (Graves, Alex *et al.*, 2006).

Artificial neural network adalah jaringan yang meniru cara kerja sel saraf pada otak makhluk hidup dalam meneruskan informasi antar sel saraf (Heaton, 2008). *Artificial neural network* adalah salah satu metode yang digunakan untuk melakukan *automatic speech recognition* sejak tahun 1980-an. Penelitian mengenai penggunaan metode *hidden markov model* yang digabungkan dengan jaringan *deep neural network* telah digunakan untuk melakukan klasifikasi suara dengan *context-dependant states*, namun membutuhkan waktu dan sumber data suara yang banyak (Dahl *et al.*, 2012) dan menggunakan *Gaussian mixture model* (GMM) yang membutuhkan banyak langkah dan teknik *processing* (Miao *et al.*, 2016).

Penelitian ini menggunakan *Connectionist Temporal Classification* (CTC) yang memanfaatkan *output* setiap waktu pada jaringan saraf tiruan yang digunakan dan menggunakan proses *decoding* untuk melakukan pengenalan ucapan tanpa memerlukan proses segmentasi dan *tuning* suara. *Connectionist Temporal Classification* dengan jaringan *bidirectional long short term memory* dapat memberikan hasil *automatic speech recognition* yang lebih baik dengan nilai *label error rate* 30.5 1% dibandingkan dengan menggunakan *context-independent hidden markov model* dengan nilai *label error rate* 38.85 % dan *hidden markov model* dengan *bidirectional long short term memory* dengan nilai *label error rate* 33.84 % (Graves, A. & Schmidhuber, 2005). Penggunaan *hidden layer* tambahan pada jaringan *bidirectional long short-term memory* dapat meningkatkan akurasi pengenalan ucapan dengan *Word Error Rate* sebesar 6,5 % (Morais, 2017). *Automatic speech recognition* dengan *bidirectional long short-term memory* dapat ditingkatkan dengan menggunakan *language model* untuk memperbaiki hasil dari jaringan *bidirectional long short-term memory* sehingga dapat mengenali kata yang sulit dan jarang digunakan (Hannun, A. *et al.*, 2014).

Berdasarkan uraian di atas, penulis akan meneliti pada bidang pengenalan ucapan menggunakan *bidirectional long short term memory* dan *connectionist temporal classification* dengan judul “AUTOMATIC SPEECH RECOGNITION BAHASA INDONESIA MENGGUNAKAN BIDIRECTIONAL LONG SHORT-TERM MEMORY DAN CONNECTIONIST TEMPORAL CLASSIFICATION”.

1.2. Rumusan Masalah

Proses pengembangan *automatic speech recognition* membutuhkan proses yang panjang pada proses segmentasi dan *tuning*. Oleh karena itu, dibutuhkan suatu pendekatan untuk dapat melakukan *automatic speech recognition* tanpa membutuhkan proses segmentasi dan *tuning* suara dengan *bidirectional long short-term memory* dan *connectionist temporal classification*.

1.3. Tujuan Penelitian

Tujuan penelitian ini adalah mengembangkan aplikasi yang berfungsi untuk mengubah suara dari percakapan manusia ke dalam bentuk teks menggunakan metode *connectionist temporal classification* dan jaringan saraf tiruan *bidirectional long short term memory*.

1.4. Batasan Masalah

Penulis menentukan dua batasan masalah untuk mencegah meluasnya ruang lingkup permasalahan, yaitu:

1. Suara percakapan yang diubah ke bentuk teks hanya didapatkan melalui satu *audio device*.
2. Bahasa yang digunakan pada percakapan adalah Bahasa Indonesia.

1.5. Manfaat Penelitian

Manfaat yang diperoleh pada penelitian ini adalah.

1. Menghasilkan sebuah aplikasi yang dapat mengubah ucapan dari *input* ke dalam bentuk teks menggunakan *connectionist temporal classification* dan *bidirectional long short term memory*.
2. Menjadikan penelitian sebagai referensi untuk penelitian selanjutnya.

1.6. Metode Penelitian

Penulis akan menggunakan tahapan-tahapan berikut pada penelitian ini.

1. Studi Literatur

Penulis mengumpulkan referensi yang berkaitan dengan pengenalan ucapan (*speech recognition*), *mel frequency cepstral coefficients*, *recurrent neural network*, *bidirectional long short term memory*, dan *connectionist temporal classification* dari berbagai sumber yang dapat dipercaya.

2. Analisa Permasalahan

Pada tahap analisa permasalahan, penulis akan menganalisa referensi yang telah dikumpulkan pada tahap studi literatur.

3. Perancangan Sistem

Penulis akan melakukan perancangan arsitektur, pengumpulan data, pemisahan data yang telah diperoleh untuk *training* dan *testing*, dan perancangan antarmuka sesuai dengan hasil studi literatur yang telah diperoleh.

4. Implementasi

Pada tahap implementasi, penulis mengimplementasikan hasil analisis dan perancangan yang telah dilakukan pada tahap sebelumnya dalam membuat aplikasi.

5. Pengujian

Penulis akan melakukan pengujian terhadap aplikasi yang telah dibangun pada tahap sebelumnya. Pengujian bertujuan untuk memastikan aplikasi dapat berjalan dengan baik dan dapat memproses permintaan pengguna dengan baik.

6. Dokumentasi Sistem dan Penyusunan Laporan

Pada tahap dokumentasi sistem dan penyusunan laporan, penulis akan melakukan dokumentasi aplikasi dan menyusun laporan untuk hasil yang didapatkan pada tahap analisa permasalahan, perancangan sistem, implementasi, dan pengujian.

1.7. Sistematika Penulisan

Sistematika penulisan pada skripsi ini terdiri dari lima bagian utama sebagai berikut.

Bab 1: Pendahuluan

Bab ini berisi mengenai latar belakang dari penelitian yang dilaksanakan, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, dan sistematika penulisan.

Bab 2: Landasan Teori

Bab ini berisi teori yang diperlukan untuk menyelesaikan masalah pada penelitian ini. Teori yang berhubungan dengan pengenalan ucapan (*speech recognition*), *mel frequency cepstral coefficients*, jaringan saraf tiruan, *recurrent neural network*, *long short term memory*, dan *connectionist temporal classification* akan dibahas pada bab ini.

Bab 3: Analisis dan Perancangan

Bab ini menjelaskan data yang digunakan, arsitektur umum, *preprocessing*, serta analisis dan penerapan metode *Connectionist Temporal Classification* untuk mengubah suara yang ditangkap melalui alat mikrofon menjadi teks.

Bab 4: Implementasi dan Pengujian

Bab ini berisi implementasi dan pengujian dari analisis dan perancangan yang telah dijabarkan pada bab sebelumnya.

Bab 5: Kesimpulan dan Saran

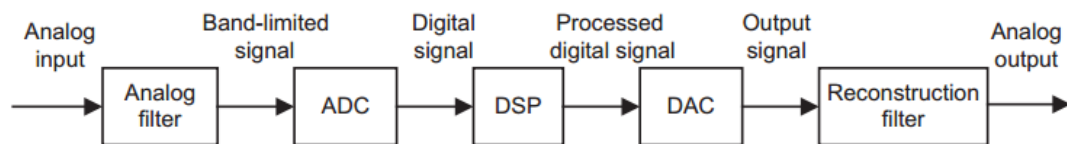
Bab ini berisi kesimpulan dari penelitian yang telah dilakukan dan saran yang diajukan oleh penulis untuk perbaikan dan pengembangan penelitian berikutnya.

BAB 2

LANDASAN TEORI

2.1. *Digital Signal Processing*

Digital signal processing adalah perhitungan, algoritma, dan teknik yang digunakan untuk memanipulasi sinyal yang didapatkan dari data sensor setelah diubah ke dalam bentuk digital (Smith, 1999). Sinyal yang didapatkan umumnya berasal dari data sensor pada kehidupan nyata, seperti getaran seismik, citra visual, gelombang suara, dan lain-lain. *Digital signal processing* banyak digunakan pada berbagai bidang, salah satunya pada bidang *speech recognition*.



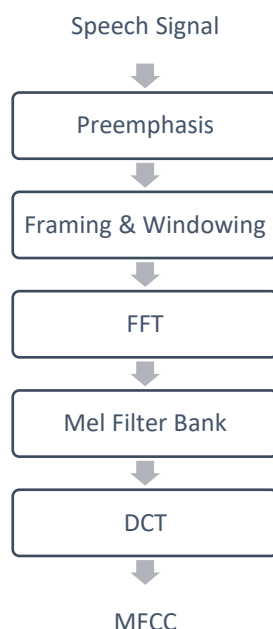
Gambar 2.1. Skema *digital signal processing* (Oshana, 2012)

Proses pengubahan suara dari bentuk analog menjadi bentuk digital ditunjukkan pada Gambar 2.1. Suara yang didapatkan melalui *input* analog akan disaring dan diubah menjadi *digital audio* pada proses *analog-to-digital conversion* (ADC). Suara yang telah diubah ke dalam bentuk digital akan diproses pada *digital signal processing* (DSP) dan dilanjutkan dengan mengubah suara bentuk digital tersebut ke dalam bentuk analog pada proses *digital-to-analog conversion* (DAC). Tidak semua implementasi *digital signal processing* melewati proses *analog-to-digital conversion* dan *digital-to-analog conversion*. *Speech recognition* tidak membutuhkan proses *digital-to-analog conversion* karena aplikasi cukup mengenali suara yang didapatkan dan mengubahnya menjadi teks. *Text-to-speech* membutuhkan proses *digital-to-analog conversion* untuk dapat mengucapkan kata-kata tersebut melalui alat *output* yang telah disediakan dan tidak membutuhkan *analog-to-digital conversion* untuk menerima *input* dari pengguna.

2.2. Mel Frequency Cepstral Coefficients (MFCC)

Mel Frequency Cepstral Coefficients (MFCC) adalah metode ekstraksi fitur yang digunakan untuk merepresentasikan sinyal suara. MFCC diperkenalkan oleh Davis dan Mermelstein pada tahun 1980. Alur pemrosesan MFCC mengikuti alur pemrosesan cara kerja sistem pendengaran manusia yang merespon suara berfrekuensi rendah secara linear dan suara berfrekuensi tinggi secara logaritmik sehingga hasil ekstraksi fiturnya mendekati persepsi yang dihasilkan indra pendengaran manusia (Davis, S. B. & Mermelstein, 1980).

Ada lima tahap dalam melakukan ekstraksi fitur sinyal suara menggunakan *Mel Frequency Cepstral Coefficients*, yaitu dengan melakukan *preemphasis*, *framing*, *windowing*, konversi ke domain frekuensi menggunakan *Fast Fourier Transform*, penerapan *mel filter bank*, dan konversi ke domain waktu menggunakan *Discrete Cosine Transform*. Urutan proses yang dilakukan dapat dilihat pada Gambar 2.2.



Gambar 2.2. Diagram Blok *Mel Frequency Cepstral Coefficients*

2.2.1. Preemphasis

Preemphasis digunakan untuk menekan suara dengan frekuensi tinggi pada sinyal suara yang diperoleh untuk mengurangi efek *damping* pada saat mekanisme produksi suara. Selain itu, *preemphasis* juga dapat menguatkan puncak spektral suara berfrekuensi

tinggi. Secara matematis, *preemphasis* dapat dinyatakan menggunakan persamaan 2.1 dengan y_n sebagai hasil dari *preemphasis* dan x_n sebagai *input* sinyal suara (Yucesoy *et al.*, 2013).

$$y_n = x_n - ax_{n-1}, a \approx 0.95 - 0.97 \quad (2.1)$$

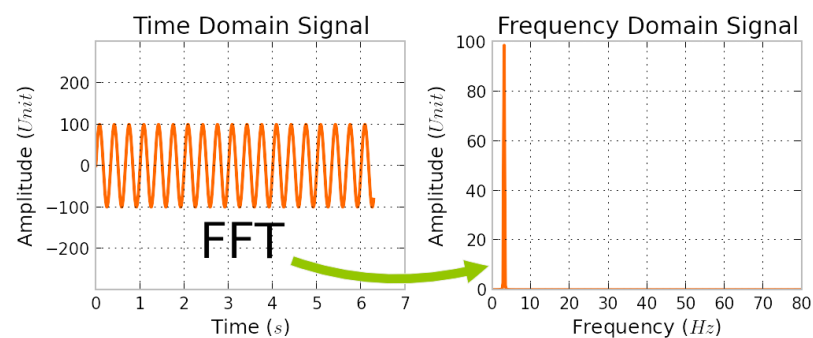
2.2.2. Framing dan windowing

Pada proses *framing*, sinyal suara disegmentasi menjadi beberapa bingkai dengan setiap bingkai berukuran 25 milidetik. Antara satu bingkai dengan bingkai lainnya, terdapat bagian yang bertumpang-tindih dengan ukuran 10 milidetik (*frame step*). Hal ini ditujukan untuk menjaga kesinambungan antarbingkai. Selanjutnya, *windowing* dilakukan untuk mencegah ketidaksinambungan sinyal suara pada setiap ujung awal dan setiap ujung akhir bingkai. *Window* yang umumnya digunakan adalah *hamming window* yang secara matematis dinyatakan berdasarkan persamaan 2.2.

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \quad (2.2)$$

2.2.3. Fast Fourier Transform

Fast Fourier Transform (FFT) digunakan untuk melakukan konversi dari *domain* waktu menjadi *domain* frekuensi dengan cepat. Penggunaan *Fast Fourier Transform* dikembangkan atas dasar bahwa penyelesaian masalah dapat dilakukan dengan mengubah atau memodelkan suatu permasalahan dalam representasi matematika dan memiliki keuntungan dalam segi efisiensi dibandingkan hal lainnya. Konsep konversi *Fast Fourier Transform* dapat diperhatikan pada Gambar 2.3.



Gambar 2.3. *Fast Fourier Transform* (Castillo, 2017)

Perhitungan *Fast Fourier Transform* dapat dilakukan menggunakan persamaan 2.3 dengan k adalah nilai 0 hingga $N - 1$ untuk merepresentasikan frekuensi f_k sesuai dengan persamaan 2.4 dengan f_s sebagai *sampling rate* dalam satuan hertz, x_m sebagai sinyal yang ingin diubah, dan N_m sebagai banyak sinyal yang ingin dikonversi (Sigurdson *et al.*, 2006). Estimasi kepadatan spektral energi dapat dihitung dengan menggunakan persamaan 2.5.

$$X_k = \left| \sum_{m=0}^{N_m-1} x_m e^{\frac{-j2\pi km}{N_m}} \right| \quad (2.3)$$

$$f_k = \frac{k f_s}{N} \quad (2.4)$$

$$P_k = \frac{1}{N} |X_k|^2 \quad (2.5)$$

2.2.4. Mel filter bank

Mel filter bank adalah sebuah *band-pass filter* yang saling bertumpang-tindih (Gupta *et al.*, 2013). Nilai frekuensi dalam spektrum *FFT* sangat lebar sehingga harus dipetakan ke dalam *Mel scale* untuk mengetahui energi yang tersedia pada setiap titik dengan bantuan *triangular filter bank*. Pada saat pemetaan, *mel scale* akan memiliki jarak yang bersifat linear jika frekuensi berada di bawah 1 kilohertz dan logaritmik jika berada di atas atau sama dengan 1 kilohertz.

Setiap *filter bank* pada *mel scale* dibuat dengan menggunakan persamaan 2.6 untuk mendapatkan nilai terendah dan tertinggi frekuensi suara dalam *mel frequency* dan membagikannya ke dalam N *filter bank*.

$$mel = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad (2.6)$$

Filter bank dapat dibuat setelah nilai terendah dan tertinggi *frekuensi* telah dibagi ke dalam N *filter bank* dengan jarak yang sama. *Filter bank* pertama akan berawal dari titik pertama, berpuncak pada titik kedua, dan kembali ke titik ketiga. Persamaan 2.7 digunakan untuk nilai untuk menyusun *filter bank* yang dibutuhkan.

$$H(k, m) = \begin{cases} 0 & \text{untuk } f(k) < f_c(m-1) \\ \frac{f(k) - f_c(m-1)}{f(m) - f_c(m-1)} & \text{untuk } f_c(m-1) \leq f(k) < f_c(m) \\ \frac{f(k) - f_c(m-1)}{f(m) - f_c(m-1)} & \text{untuk } f_c(m) \leq f(k) < f_c(m+1) \\ 0 & \text{untuk } f(k) \geq f_c(m+1) \end{cases} \quad (2.7)$$

Setiap nilai dari hasil proses *fast fourier transform* akan disaring menggunakan *filter bank*. Nilai untuk setiap *filter bank* dapat dihitung menggunakan persamaan 2.8 dengan k adalah indeks *FFT* yang ingin disaring, m sebagai indeks *filter bank* yang ingin digunakan, dan D_k sebagai nilai *FFT* pada indeks k .

$$D'_m = \ln \left(\sum_{k=0}^{N-1} |D_k| \cdot H(k, m) \right) \quad (2.8)$$

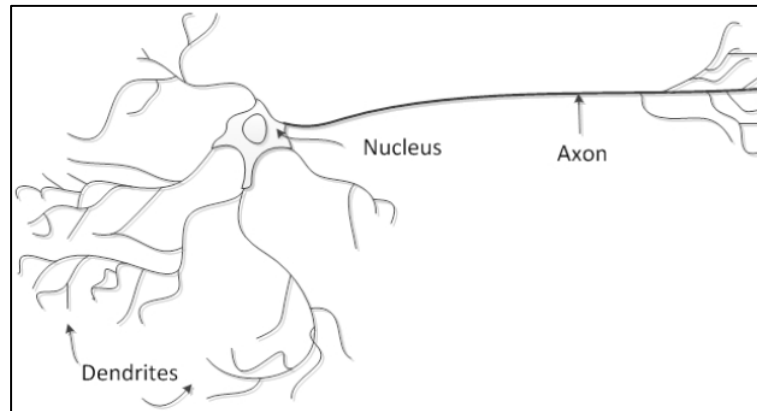
2.2.5. Discrete Cosine Transform (DCT)

Discrete Cosine Transform (DCT) digunakan untuk menghitung nilai koefisien dengan mengubah kembali nilai *Mel spectrum* ke dalam domain waktu. Proses *DCT* dapat dinyatakan melalui persamaan 2.9.

$$c_n = \sum_{k=1}^K D'_k \cos \left[n(k - 0.5) \frac{\pi}{K} \right] \quad (2.9)$$

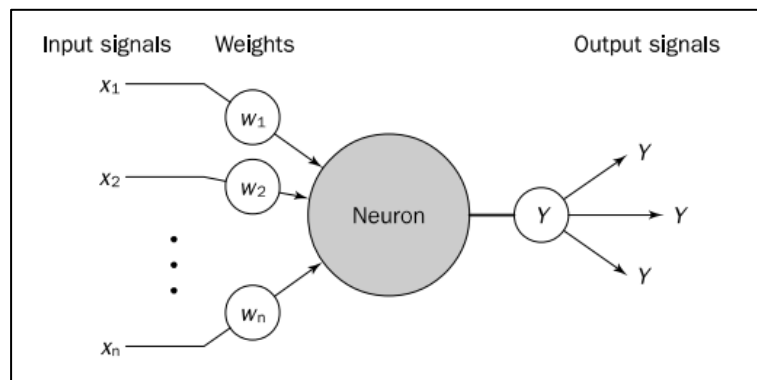
2.3. Artificial Neural Network

Artificial neural network adalah sebuah jaringan yang terinspirasi dari cara kerja sel saraf (*neuron*) pada otak makhluk hidup dalam meneruskan informasi dari/kepada sel saraf lain (Negnevitsky, 2005). Gambar 2.4 menunjukkan sebuah sel saraf yang terdiri dari sebuah nukleus, akson, dan dendrit. Dendrit berfungsi untuk menerima informasi dari sel saraf lain, sedangkan akson bertugas untuk menghubungkan sel saraf dengan sel saraf lainnya.



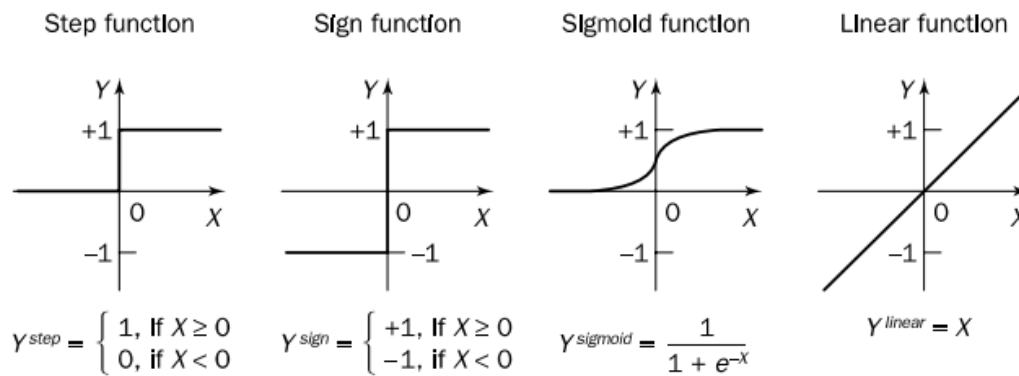
Gambar 2.4. Struktur sel saraf (Soares & Souza, 2016)

Artificial neural network memiliki arsitektur yang sama dengan sel saraf dengan memiliki sebuah nukleus, beberapa dendrit, dan satu akson. Sel saraf akan membuat akson meneruskan sinyal (informasi) kepada sel saraf lainnya ketika melewati *threshold potential* (Heaton, 2008), (Soares & Souza, 2016).



Gambar 2.5 Sebuah *neuron* pada *artificial neural network* (Negnevitsky, 2005)

Artificial neural netowrk memiliki beberapa *neuron* yang dapat terhubung dengan *neuron* lain dengan garis terarah (*directed link*). Setiap garis tersebut memiliki bobot w_i yang menandakan kuatnya garis tersebut. Jumlah dari perkalian antara beban setiap garis yang terhubung dengan *neuron* dan nilai *neuron* tersebut akan digunakan pada fungsi aktivasi dan diteruskan kepada neuron berikutnya.



Gambar 2.6 Fungsi aktivasi pada sebuah *neuron* (Negnevitsky, 2005)

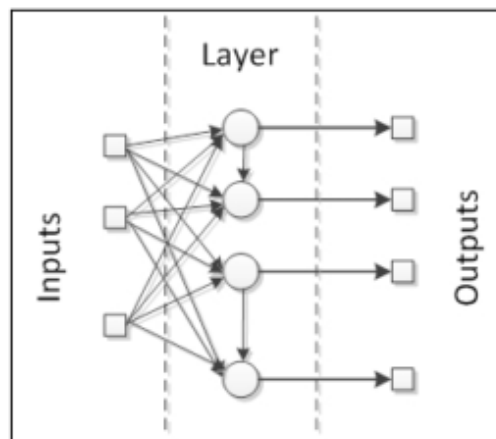
Fungsi aktivasi menentukan nilai *output* dari suatu *neuron* menuju *neuron* lain untuk mendapatkan hasil yang diharapkan (Negnevitsky, 2005). Empat fungsi aktivasi yang umumnya digunakan pada aplikasi *artificial neural network* adalah *step function*, *sign function*, *sigmoid function*, dan *linear function*. *Sigmoid function* umumnya digunakan untuk memetakan nilai *output* dari antara $-\infty$ dan $+\infty$ ke dalam nilai di antara -1 hingga 1. *Linear functions* akan memberikan nilai yang sama dengan jumlah garis dari *neuron* sebelumnya. Fungsi aktivasi *Rectified Linear Units (ReLU)* memetakan nilai *output* dengan mengabaikan nilai yang lebih kecil dari 0 yang dapat dinyatakan dengan persamaan 2.10 (Krizhevsky *et al.*, 2012).

$$f(x) = \max(0, x) \quad (2.10)$$

Softmax function umumnya digunakan pada model *artificial neural network* generatif untuk mendistribusikan *output* terhadap K kelas dengan nilai di antara 0 hingga atau sama dengan 1. Persamaan 2.11 menjelaskan bagaimana *softmax function* mendistribusikan nilai *output*.

$$y_k^t = \frac{e^{a_k^t}}{\sum_j e^{a_j^t}} \quad (2.11)$$

Artificial neural network dapat dibagi ke dalam dua jenis menurut cara pembelajarannya, yaitu *supervised* atau *active learning* dan *unsupervised learning*. *Supervised learning* membutuhkan *dataset training* untuk jaringan, sedangkan *unsupervised learning* melakukan klasifikasi berdasarkan pola dan fitur signifikan dari *input* yang diberikan.

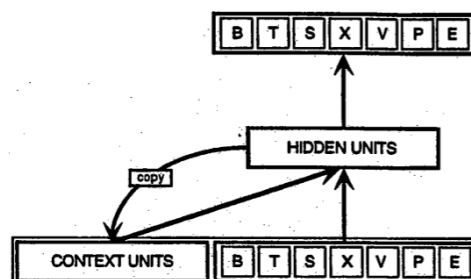


Gambar 2.7. Contoh *feed-forward network* (Soares & Souza, 2016)

Ada dua cara untuk menghubungkan *neuron* pada *artificial neural network* menurut Russel & Norvig. *Feed-forward network* memiliki *neuron* yang hanya memiliki satu alur. Setiap *neuron* yang terdapat pada *feed-forward network* tidak kembali kepada *neuron* sebelumnya atau dapat terhubung kepada *neuron* itu sendiri untuk *epoch* berikutnya. *Feed-forward network* biasanya menggunakan tiga *layer*, yaitu *input layer*, *hidden layer*, dan *output layer*. *Recurrent network* memiliki *neuron* yang mengarah kepada *neuron* pada *epoch* selanjutnya.

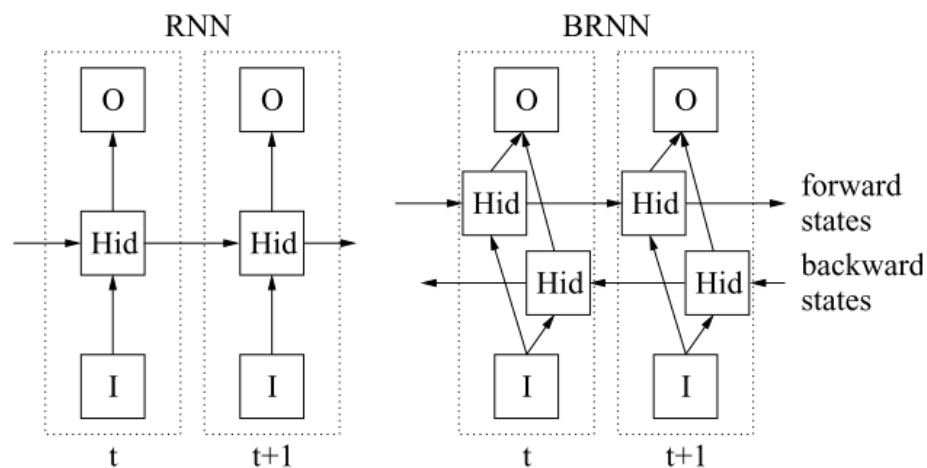
2.4. Recurrent Neural Network

Recurrent network adalah salah satu jenis jaringan saraf tiruan dengan *neuron* yang mengarah kepada *neuron* pada *epoch* selanjutnya. Gambar 2.8 menunjukkan salah satu contoh *recurrent neural network*, yaitu *state-space model*. *State-space model* mengirimkan *output* ke *input* selanjutnya sehingga memungkinkan *recurrent network* untuk memiliki ingatan jangka pendek atau *short-term memory* (Haykin, 2008).



Gambar 2.8. Contoh *struktur recurrent network* (Servan-Schreiber *et al.*, 1991)

Recurrent neural network dapat bersifat *bidirectional*, yaitu jaringan *recurrent neural network* dengan dua atau lebih *recurrent neural network* terpisah pada *timestep* yang berbeda sehingga dapat mengingat konteks sebelumnya (*past context*) dan sesudahnya (*future context*). Contoh arsitektur *bidirectional recurrent neural network* dapat dilihat pada Gambar 2.9.



Gambar 2.9 *Recurrent neural network* dan *bidirectional RNN* (Graves, Alex, 2013)

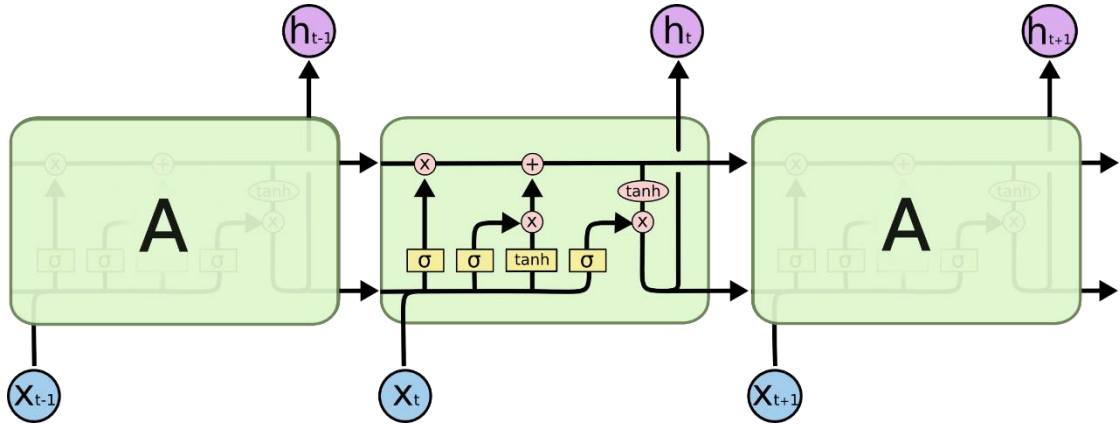
2.5. Long Short Term Memory

Recurrent neural network mengizinkan *artificial neural network* dalam memetakan *input* dan *output* secara kontekstual, tetapi nilai *error* pada *output* akan meningkat setiap kali *input* diberikan kepada jaringan. Salah satu cara untuk mengatasi masalah tersebut adalah dengan menggunakan *Long Short Term Memory*.

Long Short Term Memory digunakan pada masalah yang membutuhkan urutan kontekstual yang panjang. Beberapa penelitian telah menerapkan *Long Short Term Memory* pada banyak bidang dan permasalahan, seperti pengenalan suara (Sak *et al.*, 2015) dan pengenalan keaslian emosi (Kim & Huynh, 2017). *Long Short Term Memory* dapat diterapkan pada jaringan *bidirectional recurrent neural network* sehingga dapat menggunakan informasi baik dari *past context* maupun *future context* untuk mencegah nilai *error* yang meningkat pada setiap *input* yang terjadi pada *recurrent neural network*.

Arsitektur *Long Short Term Memory* terdiri dari satu atau beberapa *subnet* yang terhubung secara *recurrent* atau dapat disebut sebagai *memory blocks*. Setiap *memory blocks* memiliki satu atau lebih *memory cell* yang terhubung dengan dirinya sendiri dan

tiga unit multiplikatif, yaitu gerbang *input*, *output*, dan *forget*. Gerbang *input*, *output*, dan *forget* digunakan untuk operasi *write*, *read*, dan *reset* untuk sel. (Graves, Alex, 2013)



Gambar 2.10 Satu blok *LSTM* memiliki 4 layer (Olah, 2015)

2.5.1. Input Gates

Input gate berfungsi untuk menentukan apakah nilai yang dimasukkan pada waktu t dapat digunakan. Nilai *input gate* dapat dinyatakan dengan persamaan 2.12 dengan x_t sebagai nilai *input* pada *memory block*, w_{xi} sebagai beban antara neuron asal dengan *input gate*, w_{hi} sebagai beban antara *input gate* dengan nilai *hidden layer* pada waktu sebelumnya ($t - 1$), h_{t-1} sebagai nilai *input* dari *memory block* pada waktu sebelumnya, w_{ci} sebagai beban antara *input gate* dan sel, c_{t-1} sebagai nilai sel pada waktu sebelumnya, dan b_i sebagai nilai bias untuk *input gate*. σ merupakan fungsi aktivasi *sigmoid* untuk memetakan nilai *input gate* ke dalam nilai antara 0 dan 1.

$$i_t = \sigma(w_{xi}x_t + w_{hi}h_{t-1} + w_{ci}c_{t-1} + b_i) \quad (2.12)$$

2.5.2. Forget Gates

Forget gates berfungsi untuk menentukan penggunaan nilai *neuron* yang didapatkan dari waktu $t - 1$ pada waktu t . Nilai *forget gates* dapat dihitung dengan menggunakan persamaan 2.13.

$$f_t = \sigma(w_{xf}x_t + w_{hf}h_{t-1} + w_{cf}c_{t-1} + b_f) \quad (2.13)$$

2.5.3. Cells

Cell pada *memory block* berfungsi untuk menyimpan informasi untuk setiap waktu. Persamaan 2.14 dapat digunakan untuk menghitung nilai *memory cell* atau informasi yang perlu disimpan.

$$c_t = f_t c_{t-1} + i_t \tanh(w_{xc} x_t + w_{hc} h_{t-1} + b_c) \quad (2.14)$$

2.5.4. Output Gates

Output gate berfungsi untuk menentukan pemberian nilai *memory cell* ke waktu $t + 1$ dan titik selanjutnya. Nilai *output gates* dapat dihitung dengan persamaan 2.15 dengan b_o sebagai bias pada *output gate*.

$$o_t = \sigma(w_{xo} x_t + w_{ho} h_{t-1} + w_{co} c_t + b_o) \quad (2.15)$$

2.5.5. Cell Outputs

Nilai *output gates* akan menentukan pemberian nilai kepada *neuron* selanjutnya dan kepada *neuron* pada waktu $t + 1$. Nilai dari *output neuron* dapat dihitung menggunakan persamaan 2.16

$$h_t = o_t \tanh(c_t) \quad (2.16)$$

2.6. Early Stopping

Neural network umumnya rentan dengan *overfitting*, yaitu jaringan yang memiliki performa yang baik dengan data *training* dan memburuk pada titik tertentu. *Early stopping* adalah salah satu teknik yang digunakan untuk mencegah *overfitting* pada *neural network* (Prechelt, 2012). Penerapan *early stopping* pada jaringan *LSTM* berfungsi untuk mencegah jaringan untuk mengenal *noise* pada data yang diberikan dengan membagi *dataset* menjadi *training set* dan *validation set*. Proses *training* akan dihentikan jika *error* pada *validation set* semakin memburuk pada N *epoch* selanjutnya.

2.7. Language Model

Statistical language model adalah salah satu model sistem temu balik informasi (*information retrieval system*) yang digunakan untuk menghitung probabilitas urutan kata atau kalimat berdasarkan *corpus* teks yang disediakan. *Language model* umumnya

digunakan pada penerapan *pengenalan ucapan* dan pemecahan sintaksis kalimat. Peluang suatu *query* umumnya dihitung berdasarkan *n-gram model*. *Unigram (1-gram model)* menggunakan asumsi bahwa suatu kata hanya bergantung dengan kata tersebut, sedangkan *bigram (2-gram model)* dan *trigram (3-gram model)* menghitung peluang suatu kata baru sesuai dengan kata sebelumnya (Song & Croft, 1999).

$$P(w_{1,n}) = P(w_1)P(w_2) \dots P(w_n) \quad (2.17)$$

$$P(w_{1,n}) = P(w_1)P(w_2|w_1) \dots P(w_n|w_{n-1}) \quad (2.18)$$

$$P(w_{1,n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1,2}) \dots P(w_n|w_{n-2,n-1}) \quad (2.19)$$

Persamaan 2.20 menunjukkan metode untuk melakukan estimasi peluang suatu kata atau kalimat menggunakan *maximum likelihood estimate* dengan $n_{t,d}$ sebagai banyaknya kata t pada dokumen d dan N_d sebagai banyaknya kata pada dokumen d .

$$P_{ML}(t|d) = \frac{n_{t,d}}{N_d} \quad (2.20)$$

Language model dengan *maximum likelihood estimate* tidak dapat mengetahui peluang pasangan kata yang tidak terdapat pada dokumen yang digunakan (*unseen*) jika peluang suatu kalimat dihitung menggunakan persamaan 2.20. *Back-off smoothed n-gram model* mengatasi masalah tersebut dengan menggunakan nilai *backoff* untuk mendistribusikan peluang setiap pasangan kata yang terdapat pada dokumen ke setiap pasangan kata yang tidak terdapat pada dokumen (Katz, 1987). Peluang untuk suatu kata atau kalimat pada *back-off smoothed n-gram model* dapat dihitung menggunakan persamaan 2.21 dengan $b(w_1^{n-1})$ sebagai nilai *backoff weight* $b(w_1^{n-1})$ dapat dihitung menggunakan persamaan 2.22.

$$p(w_n|w_1^{n-1}) = \begin{cases} \bar{p}(w_n|w_1^{n-1}) & \text{if } w_1^n \text{ was seen} \\ b(w_1^{n-1})p(w_n|w_2^n) & \text{otherwise} \end{cases} \quad (2.21)$$

$$b(w_1^{n-1}) = \frac{1 - \sum_{w_n: c(w_2^n) > 0} \bar{p}(w_n|w_1^{n-1})}{1 - \sum_{w_n: c(w_2^n) > 0} \bar{p}(w_n|w_2^{n-1})} \quad (2.22)$$

Nilai $\bar{p}(w_n|w_1^{n-1})$ merupakan nilai peluang yang telah mengalami *discount* yang dapat dihitung dengan menggunakan persamaan 2.23 dengan $c(w_1^n)$ sebagai banyaknya pasangan kata w_1^n pada dokumen.

$$\bar{p}(w_n|w_1^{n-1}) = d_{c(w_1^n)} \frac{c(w_1^n)}{c(w_1^{n-1})} \quad (2.23)$$

Koefisien *discount* d_r dapat diperoleh dengan menggunakan persamaan 2.24 dengan d_r sebagai nilai *discount* yang diberikan pada pasangan kata yang terjadi sebanyak r kali pada dokumen. Nilai k yang baik digunakan adalah 5.

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}, \text{ for } 1 \leq r \leq k \quad (2.24)$$

r^* merupakan estimasi *Good-Turing* yang menyatakan bahwa untuk setiap n -gram yang terjadi sebanyak r kali, sistem harus menganggap bahwa n -gram tersebut terjadi sebanyak r^* kali yang dapat dinyatakan dengan persamaan 2.25.

$$r^* = (r + 1) \frac{n_{r+1}}{n_r} \quad (2.25)$$

2.8. Connectionist Temporal Classification

Menurut Graves (2006), *Connectionist Temporal Classification (CTC)* mengizinkan suatu *neural network* untuk melakukan klasifikasi label tanpa mengetahui *alignment* dari data yang diberikan. *CTC* dapat digunakan dengan semua arsitektur *recurrent neural network*, namun arsitektur *bidirectional* lebih disarankan karena setiap probabilitas label pada suatu waktu memiliki keterhubungan dengan *input* pada waktu sebelum dan sesudahnya. Sebuah jaringan *CTC* terdiri dari X *input sequence* sepanjang T dan menggunakan *softmax output layer* yang terdiri dari $L + 1$ (atau L') unit yang menyatakan probabilitas untuk label k pada waktu t (dinyatakan dengan y_k^t). L' terdiri dari L label yang perlu dikenali oleh jaringan dan label *blank* yang berfungsi untuk menyatakan tidak ada label pada waktu t . Setiap y_k^t akan membentuk sebuah daftar probabilitas untuk setiap waktu ($\pi \in L'^T$) sebagai *paths* dengan total nilai probabilitas yang dapat dihitung menggunakan persamaan 2.26.

$$p(\pi|x, S) = \prod_{t=1}^T y_{\pi_t}^t \quad (2.26)$$

Langkah berikutnya adalah membuat sebuah peta *many-to-one* $B: L'^T \mapsto L^{\leq T}$, dimana $L^{\leq T}$ adalah set label yang memungkinkan, dengan menghapus label yang berulang terlebih dahulu dan menghapus label *blank*.

$$B(aa - -abb) = B(a - ab) = aab \quad (2.27)$$

Dengan peta B , probabilitas bersyarat untuk *labelling* $l \in L^{\leq T}$ dapat dihitung dengan melakukan penjumlahan terhadap semua *paths* yang dipetakan dari B .

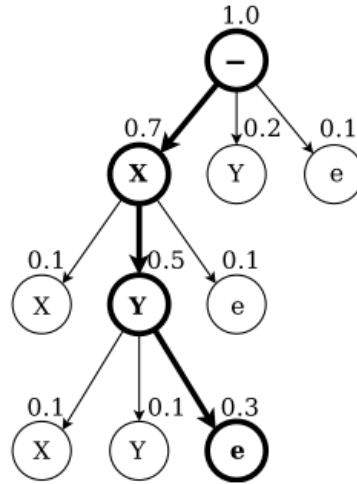
$$p(l|x) = \sum_{\pi \in B^{-1}(l)} p(\pi|x) \quad (2.28)$$

Dengan persamaan 2.28, *output* yang dihasilkan berupa *labelling* yang paling memungkinkan untuk *input* yang diberikan.

$$h(x) = \arg \max_{l \in L^{\leq T}} p(l|x) \quad (2.29)$$

Fungsi objektif pada *CTC* memiliki prinsip *maximum likelihood*, yaitu untuk memaksimalkan probabilitas *log* untuk klasifikasi yang benar pada *training set* dengan meminimalisir nilai O pada persamaan 2.30 (Yi *et al.*, 2017).

$$O = -\ln \left(\prod_{(x,z) \in S} p(z|x) \right) = - \sum_{(x,z) \in S} \ln p(z|x) \quad (2.30)$$



Gambar 2.11 *Prefix search decoding* untuk label X, Y (Graves, Alex *et al.*, 2006)

Proses *decoding* pada *CTC* dilakukan untuk mendapatkan label yang paling memungkinkan untuk *input* yang diberikan. Salah satu teknik *decoding* yang umumnya digunakan dengan *CTC* adalah *prefix beam search* dengan memanfaatkan *language model* untuk melakukan *decoding* (Hannun, A. Y. *et al.*, 2014). Gambar 2.11 menunjukkan setiap titik pada *tree labelling* memiliki nilai probabilitas dan memiliki anak dengan jumlah nilai probabilitas yang sama dengan titik tersebut. *Child* (anak) dari *tree labelling* merupakan *prefix* untuk *parent* yang telah ditentukan. Persamaan 2.31

menunjukkan hasil *decoding* yang digunakan adalah *path* dengan probabilitas paling besar pada *language model* (p_{LM}) dan probabilitas suatu *label* dari hasil jaringan tersebut (p_{Net}) dengan beban α dan β untuk mengatur pengaruh *language model* dan banyak kata $wc(y)$ pada hasil *decoding*.

$$Q(y) = \log(p_{Net}(y|x)) + \alpha \log(p_{LM}(y)) + \beta wc(y) \quad (2.31)$$

Gambar 2.12 menunjukkan *pseudocode* implementasi *prefix beam search* untuk mendapatkan *labeling* dengan peluang tertinggi. Daftar A_{prev} digunakan untuk menyimpan kalimat *prefix*. p_b dan p_{nb} adalah nilai peluang untuk setiap *time step* dengan nilai awal 1 untuk peluang *blank* pada semua waktu dan 0 untuk peluang *non-blank* pada semua waktu. Untuk setiap waktu dan setiap A_{prev} , peluang untuk setiap *labeling* dengan karakter baru dihitung. Jika karakter c adalah *blank*, maka karakter tersebut tidak ditambahkan ke A_{next} . Jika karakter c adalah spasi, peluang untuk *labeling* baru dapat dihitung menggunakan persamaan 2.31. Jika karakter c adalah karakter lainnya, *labelling* tersebut akan ditambahkan ke A_{next} dan peluang *labeling* baru akan dihitung jika *labeling* tersebut belum tersedia pada daftar A_{prev} . k *labelling* terbaik pada A_{next} akan dimasukkan ke A_{prev} dan *labelling* terbaik pada A_{prev} akan digunakan sebagai hasil proses *decoding*.

```

 $p_b(\emptyset; x_{1:0}) \leftarrow 1, p_{nb}(\emptyset; x_{1:0}) \leftarrow 0$ 
 $A_{prev} \leftarrow \{\emptyset\}$ 
for  $t = 1, \dots, T$  do
   $A_{next} \leftarrow \{\}$ 
  for  $\ell$  in  $A_{prev}$  do
    for  $c$  in  $\Sigma$  do
      if  $c = \text{blank}$  then
         $p_b(\ell; x_{1:t}) \leftarrow p(\text{blank}; x_t)(p_b(\ell; x_{1:t-1}) + p_{nb}(\ell; x_{1:t-1}))$ 
        add  $\ell$  to  $A_{next}$ 
      else
         $\ell^+ \leftarrow \text{concatenate } \ell \text{ and } c$ 
        if  $c = \text{end}$  then
           $p_{nb}(\ell^+; x_{1:t}) \leftarrow p(c; x_t)p_b(\ell; x_{1:t-1})$ 
           $p_{nb}(\ell; x_{1:t}) \leftarrow p(c; x_t)p_b(\ell; x_{1:t-1})$ 
        else if  $c = \text{space}$  then
           $p_{nb}(\ell^+; x_{1:t}) \leftarrow p(W(\ell^+)|W(\ell))^\alpha p(c; x_t)(p_b(\ell; x_{1:t-1}) + p_{nb}(\ell; x_{1:t-1}))$ 
        else
           $p_{nb}(\ell^+; x_{1:t}) \leftarrow p(c; x_t)(p_b(\ell; x_{1:t-1}) + p_{nb}(\ell; x_{1:t-1}))$ 
        end if
        if  $\ell^+$  not in  $A_{prev}$  then
           $p_b(\ell^+; x_{1:t}) \leftarrow p(\text{blank}; x_t)(p_b(\ell^+; x_{1:t-1}) + p_{nb}(\ell^+; x_{1:t-1}))$ 
           $p_{nb}(\ell^+; x_{1:t}) \leftarrow p(c; x_t)p_{nb}(\ell^+; x_{1:t-1})$ 
        end if
        add  $\ell^+$  to  $A_{next}$ 
      end if
    end for
  end for
   $A_{prev} \leftarrow k \text{ most probable prefixes in } A_{next}$ 
end for
return 1 most probable prefix in  $A_{prev}$ 

```

Gambar 2.12 *Pseudocode prefix beam search* (Hannun, A. Y. et al., 2014)

Tingkat kesalahan jaringan yang digunakan dapat dihitung dengan menggunakan *label error rate*. *Label error rate* dapat diperoleh dengan persamaan 2.32 dimana ED adalah banyaknya penambahan, pengurangan, dan substitusi label untuk mengubah hasil menjadi label target dengan Z sebagai panjang total label target.

$$E(h, S') = \frac{1}{Z} \sum_{(x,z) \in S'} ED(h(x)) \quad (2.32)$$

2.9. Penelitian Terdahulu

Penelitian terdahulu pada topik *speech recognition* menggunakan metode *dynamic time warping* atau *hidden markov model* untuk mengubah suara menjadi teks. Metode *dynamic time warping* menghitung jalur optimal antar dua seri waktu, sedangkan *hidden markov model* mengenali suara dengan memilih hasil dengan peluang kecocokan suara dengan *phone* dan peluang kata tersebut berdasarkan *language model*. *Dynamic time warp* memberikan hasil dengan tingkat akurasi yang lebih rendah dibandingkan dengan metode *hidden markov model* karena suara umumnya tidak bergerak dan kemampuan *hidden markov model* dalam mengenali pola dan urutan *hidden states* berdasarkan *output* yang dimiliki (Sajjan & Vijaya, 2012).

Hidden markov model dapat digabungkan dengan *deep neural network* untuk melakukan klasifikasi suara menjadi *context-dependant states*, namun membutuhkan waktu dan sumber data suara yang banyak (Dahl *et al.*, 2012) dan masih tergantung dengan *Gaussian mixture model* (GMM) yang membutuhkan banyak langkah dan teknik *processing* (Miao *et al.*, 2016). Penelitian sebelumnya juga telah membandingkan tingkat akurasi jaringan yang menggunakan *connectionist temporal classification* dan *hidden markov model* untuk melabelkan hasil dari *recurrent neural network* dengan kesimpulan bahwa *connectionist temporal classification* dengan *prefix search decoding* dapat memberikan tingkat kesalahan yang lebih rendah dengan *label error rate* 30.51 % dibandingkan *hidden markov model (context-independent)* dengan *label error rate* 38.85 % (Graves, Alex *et al.*, 2006). *Bidirectional long short term memory* dapat melakukan klasifikasi menggunakan konteks sebelum dan sesudahnya sehingga dapat digunakan pada data yang saling berhubungan seperti suara. Penggunaan *hidden layer* tambahan pada jaringan *bidirectional long short-term memory* dapat meningkatkan akurasi pengenalan ucapan dengan *Word Error Rate*

sebesar 6,5 % (Morais, 2017). *Automatic speech recognition* dengan *bidirectional long short-term memory* dapat ditingkatkan dengan menggunakan *language model* untuk memperbaiki hasil dari jaringan *bidirectional long short-term memory* sehingga dapat mengenali kata yang sulit dan jarang digunakan (Hannun, A. *et al.*, 2014).

Speech recognition juga dapat dilakukan dengan menggunakan *Microsoft Speech SDK* pada sistem operasi Windows (Serrano *et al.*, 2015) dan *SpeechRecognizer* pada Android (Naveen Kumar *et al.*, 2017) dengan memanfaatkan *library* dari sistem operasi, namun metode tersebut hanya dapat digunakan pada sistem operasi yang memiliki *library* tersebut. Penggunaan *Microsoft Speech SDK* juga memerlukan hak akses berupa kunci API.

Tabel 2.1 Penelitian Terdahulu

No	Peneliti	Tahun	Metode	Keterangan
1	Sajjan & Vijaya	2012	<i>Dynamic time warping</i> dan <i>hidden markov model</i> dengan <i>MFCC</i> dan <i>LPC</i>	<ul style="list-style-type: none"> • Perbandingan akurasi <i>dynamic time warping</i> dan <i>hidden markov model</i>. • Akurasi <i>HMM</i> lebih tinggi dibandingkan <i>DTW</i>. • Metode ekstraksi fitur <i>MFCC</i> dapat meningkatkan tingkat akurasi <i>speech recognition</i> dibandingkan dengan menggunakan <i>LPC (linear predictive coding)</i>.
2	Miao <i>et al.</i>	2016	<i>Recurrent neural network</i> dan <i>WFST-based decoding</i>	<ul style="list-style-type: none"> • <i>Speech recognition</i> pada <i>framework Eesen</i> yang menggunakan model <i>bidirectional RNN</i> dan <i>WFST-based decoding</i> dan dibandingkan dengan metode lainnya.

No	Peneliti	Tahun	Metode	Keterangan
				<ul style="list-style-type: none"> • <i>Bidirectional RNN</i> lebih cepat dalam melakukan <i>speech recognition</i> dibandingkan dengan metode <i>hybrid HMM-DNN</i>.
3	Graves <i>et al.</i>	2006	<i>Bidirectional LSTM</i> dan <i>Connectionist Temporal Classification</i>	<ul style="list-style-type: none"> • <i>Speech recognition</i> menggunakan jaringan <i>bidirectional LSTM</i> dan <i>connectionist temporal classification</i>. • <i>Label error rate</i> untuk <i>Speech recognition</i> menggunakan metode CTC (<i>prefix search</i>) lebih rendah dibandingkan metode <i>context-dependent HMM</i>, <i>context-independent HMM</i>, dan <i>BLSTM HMM</i>.
4	Serrano <i>et al</i>	2015	<i>Microsoft Speech SDK</i>	<ul style="list-style-type: none"> • Pengembangan robot pintar yang dapat melakukan percakapan. • Metode yang digunakan adalah <i>Microsoft Speech SDK</i> yang hanya tersedia pada sistem operasi Windows. <i>Microsoft Speech SDK</i> tersedia pada sistem operasi lain dengan biaya tambahan.
5	Naveen Kumar <i>et al.</i>	2017	<i>SpeechRecognizer</i> pada <i>Android</i>	<ul style="list-style-type: none"> • Pengembangan aplikasi <i>chatbot</i> untuk pengguna tuna netra.

No	Peneliti	Tahun	Metode	Keterangan
				<ul style="list-style-type: none"> • <i>SpeechRecognizer</i> hanya tersedia pada sistem operasi Android. • Aplikasi menggunakan AIML untuk mencari jawaban untuk pertanyaan pengguna.
6	Dahl <i>et al.</i>	2012	<i>Context-dependent deep neural network (deep belief network)</i> dan <i>HMM</i>	<ul style="list-style-type: none"> • <i>Context-dependent HMM</i> memiliki tingkat kesalahan lebih rendah dibandingkan dengan menggunakan <i>context-dependent deep neural network</i>.
7	Hannun, A. <i>et al.</i> ,	2014	<i>Bidirectional recurrent neural network</i> dan <i>Connectionist Temporal Classification</i>	<ul style="list-style-type: none"> • Hasil dari jaringan <i>bidirectional recurrent neural network</i> dapat ditingkatkan dengan bantuan <i>language model</i> untuk mengenal kata rumit dan kata yang jarang digunakan. • Pengenalan ucapan pada lingkungan dengan <i>noise</i> dapat dilakukan dengan nilai <i>WER</i> 19,06 %.

Perbedaan penelitian yang dilakukan dengan penelitian terdahulu adalah pelatihan dan penggunaan *speech recognition* ditujukan untuk pengguna berbahasa Indonesia. Adapun metode yang diimplementasikan oleh penulis pada penelitian ini adalah.

1. *Mel Frequency Cepstral Coefficient* sebagai metode untuk melakukan ekstraksi fitur dari suara yang diinput pada aplikasi.
2. *Bidirectional LSTM (Long Short-Term Memory)* untuk melakukan klasifikasi terhadap suara yang telah diinput (*framewise classification*).
3. *Connectionist Temporal Classification* dengan *prefix beam search* sebagai metode untuk melakukan *decoding* pada hasil dari jaringan *bidirectional LSTM*

BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Bab ini akan membahas mengenai implementasi metode yang digunakan untuk melakukan *speech recognition*, yaitu proses augmentasi suara, *preprocessing* menggunakan *mel frequency cepstral coefficient*, implementasi jaringan *bidirectional Long Short-Term Memory*, *Connectionist Temporal Classification*, dan perancangan sistem. Tahap perancangan sistem akan membahas tampilan antarmuka sistem.

3.1. Dataset

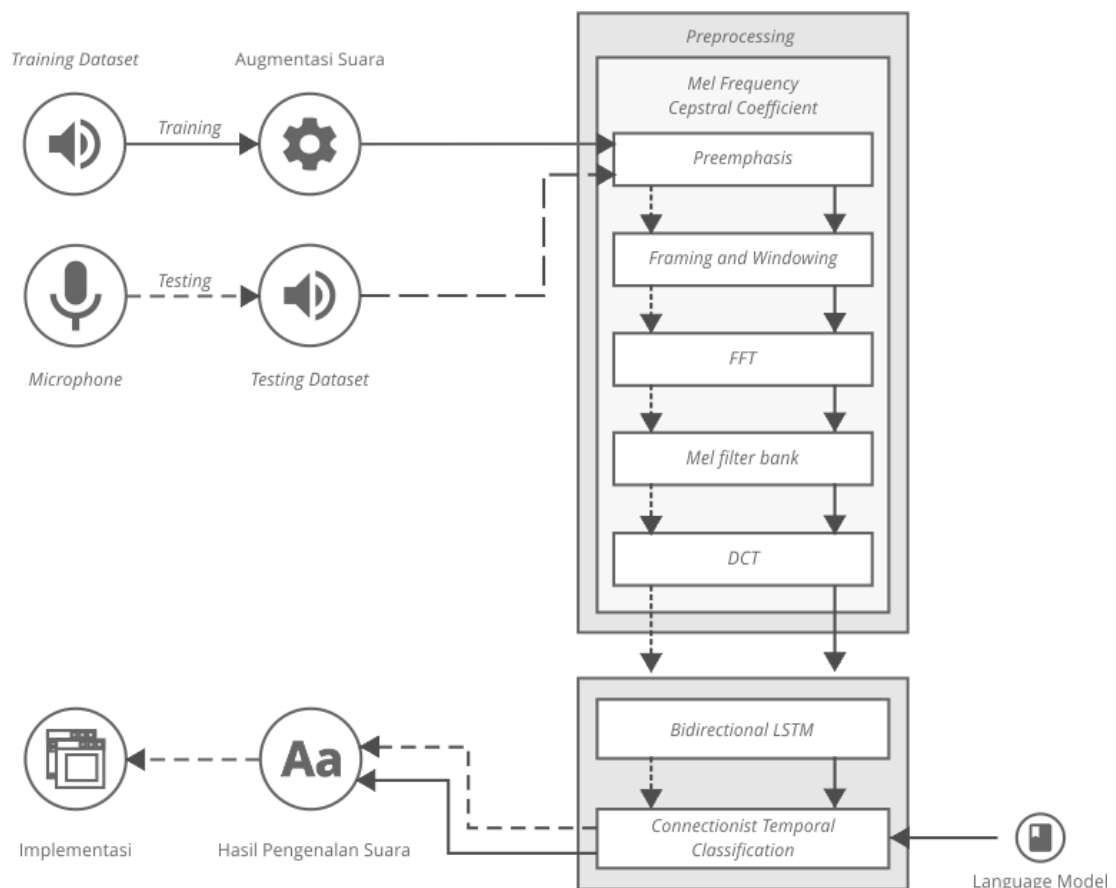
Data yang digunakan pada penelitian ini diperoleh dari *Tokyo Institute of Technology Multilingual – Indonesia Speech Corpus* atau *TITML-IDN* (Lestari *et al.*, 2011) yang terdiri dari dua puluh *speaker* yang berasal dari Indonesia (sebelas orang laki-laki dan sembilan orang perempuan) yang mengucapkan 343 kalimat terpilih dari grup *Information and Language Processing System* dan kalimat yang menggunakan bunyi *biphone* dan menggunakan frekuensi tinggi. Data direkam menggunakan *Sony DAT-recorder DTC-2000ES* dan mikrofon *Sennheiser HMD 25-1*. Suara dari 20 *speaker* pada *speech corpus* TITML dibagi ke tiga *dataset*, yaitu 80 % untuk *dataset training*, 10 % untuk *dataset validation*, dan 10 % untuk *dataset testing*.

Sebuah *dataset* tambahan direkam dari suara seorang pria berumur 22 tahun dan bersuku Tionghoa menggunakan mikrofon *Taffware MK-F100TL* pada dua kondisi yang berbeda, yaitu pada ruangan hening dan pada ruangan dengan suara latar belakang. *Dataset* tambahan akan digunakan untuk menguji metode yang digunakan dalam melakukan pengenalan ucapan pada suasana dan perangkat yang berbeda.

Setiap berkas audio pada *dataset TITML* dan *dataset* tambahan disimpan dengan menggunakan format WAV (*Waveform Audio File Format*) dan *sample rate* 16 kilohertz dan 1 *channel*.

3.2. Arsitektur Umum

Aplikasi akan merekam suara ketika aplikasi mendeteksi suara melebihi *threshold* yang telah ditentukan dan akan berhenti ketika tidak ada suara yang diberikan terhadap *input*. Suara yang diterima dari mikrofon diproses menggunakan metode *MFCC* (*Mel Frequency Cepstral Coefficients*) untuk melakukan ekstraksi fitur dengan 13 *cepstral coefficient* dan *frame* sepanjang 25 milidetik dengan 10 milidetik *window step*. Hasil dari ekstraksi fitur menggunakan *Mel Frequency Cepstral Coefficients* akan digunakan sebagai *input* untuk jaringan *bidirectional LSTM* (*Long Short-Term Memory*) untuk mengetahui probabilitas setiap *label* untuk suara pada waktu tertentu. Metode *CTC* (*Connectionist Temporal Classification*) digunakan untuk menghitung *loss* jaringan *bidirectional LSTM* dan melakukan *decoding* menggunakan *output* dari jaringan *bidirectional LSTM*. Gambar 3.1 menunjukkan arsitektur umum yang digunakan pada penelitian *Automatic Speech Recognition Bahasa Indonesia Menggunakan Long Short Term Memory dan Connectionist Temporal Classification*.



Gambar 3.1 Arsitektur Umum

3.3. Augmentasi Suara

Dataset training yang tersedia dimodifikasi dengan mengubah tempo dan *pitch* suara dengan skala sebesar 90 %, 110 %, dan 120 % dari suara asli sehingga dapat membentuk sembilan dataset baru (Ko *et al.*, 2015).

3.4. Preprocessing

Sistem akan melakukan *preprocessing* sebelum melakukan *training* dan *testing* dengan menggunakan metode *Mel Frequency Cepstral Coefficient (MFCC)* dengan *window length* sebesar 25 milidetik, *frame step* sebesar 10 milidetik, dan 13 koefisien *cepstral* untuk melakukan ekstraksi fitur dari suara yang akan diproses sehingga dapat digunakan oleh jaringan *bidirectional long short-term memory* untuk melakukan *labelling*.

3.4.1. Preemphasis

Pada tahap *preemphasis*, suara berfrekuensi tinggi ditekankan untuk mengurangi efek *damping* dari suara *input* pengguna. Efek *damping* terjadi pada saat *audio device* yang digunakan memperkecil suara dengan frekuensi tinggi pada saat perekaman berlangsung. Tabel 3.1 menunjukkan hasil tahap *preemphasis* pada beberapa sampel suara yang dapat dilakukan dengan mengubah setiap sampel suara menggunakan persamaan 2.1.

$$\begin{aligned} y_2 &= x_2 - ax_1 \quad a = 0.97 \\ &= -3 - (0.97 * 1) \\ &= -3.97 \end{aligned}$$

Tabel 3.1 Contoh Hasil *Preemphasis* pada MFCC

<i>Sample</i>	Asli	<i>Preemphasis</i>
0	1	1
1	-3	-3.97
2	4	6.91
3	-4	-7.88
4	3	6.88
5	-2	-4.91

<i>Sample</i>	<i>Asli</i>	<i>Preemphasis</i>
6	0	1.94
7	1	1
8	-1	-1.97
9	1	1.97
10	-1	-1.97
11	0	0.97
12	0	0
13	1	1
14	-2	-2.97
15	3	4.94
...		

3.4.2. Framing and Windowing

Pada proses *framing*, suara yang telah diproses pada tahap *preemphasis* akan dibagi ke dalam beberapa bagian dengan panjang 25 milidetik dengan bagian saling bertumpang-tindih sepanjang 10 milidetik. Tahap *windowing* dilakukan untuk menjaga kesinambungan pada setiap *frame*.

Penerapan *window* akan dilakukan setelah suara dibagi ke dalam satu atau beberapa *frame* dengan ukuran yang telah ditentukan. Hasil dari *windowing* dapat dihitung dengan perkalian antara nilai pada *frame* dan *window hamming*. Nilai *window hamming* didapatkan menggunakan persamaan 2.2 dengan N sebagai banyak nilai *hamming* yang ingin didapatkan dan $n = 0, 1, 2 \dots N - 1$ adalah indeks nilai *window hamming* yang ingin didapatkan. Tabel 3.2 merupakan contoh hasil dari proses *framing* dan *windowing* untuk tiga *frame* pertama dengan panjang *frame* sebesar 1 milidetik dan *frame step* sebesar 0.25 milidetik.

$$\begin{aligned}
 w_0 &= 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right) \\
 &= 0.54 - 0.46 \cos(0) \\
 &= 0.08
 \end{aligned}$$

Tabel 3.2 Hasil *framing* dan *windowing* pada MFCC

Nilai	Frames			Hamming	Windowing		
	1	2	3		1	2	3
0	1	6,88	-1,97	0,080	0,08	0,55	-0,16
1	-3,97	-4,91	1,97	0,120	-0,48	-0,59	0,24
2	6,91	1,94	-1,97	0,232	1,60	0,45	-0,46
3	-7,88	1	0,97	0,398	-3,14	0,40	0,39
4	6,88	-1,97	0	0,588	4,05	-1,16	0,00
5	-4,91	1,97	1	0,770	-3,78	1,52	0,77
6	1,94	-1,97	-2,97	0,912	1,77	-1,80	-2,71
7	1	0,97	4,94	0,990	0,99	0,96	4,89
8	-1,97	0	-6,91	0,990	-1,95	0,00	-6,84
9	1,97	1	7,88	0,912	1,80	0,91	7,19
10	-1,97	-2,97	-4,88	0,770	-1,52	-2,29	-3,76
11	0,97	4,94	-2,03	0,588	0,57	2,90	-1,19
12	0	-6,91	7,91	0,398	0,00	-2,75	3,15
13	1	7,88	-10,85	0,232	0,23	1,83	-2,52
14	-2,97	-4,88	11,82	0,120	-0,36	-0,59	1,42
15	4,94	-2,03	-9,82	0,080	0,40	-0,16	-0,79

3.4.3. Fast Fourier Transform (FFT)

Fast Fourier Transform digunakan untuk melakukan konversi sinyal dari domain waktu ke domain frekuensi dengan cepat dengan persamaan 2.3. Banyak nilai yang dihitung pada proses *fast fourier transform* adalah 512, namun banyak nilai *FFT* yang digunakan untuk perhitungan kepadatan spektral energi adalah $\frac{N}{2} + 1$.

$$X_k = \sum_{m=0}^{N_m-1} x_m e^{\frac{-j2\pi km}{N_m}} \quad N_m = 16$$

$$X_0 = 0.08e^{\frac{-j2\pi*0*0}{16}} - 0.47e^{\frac{-j2\pi*0*1}{16}} + 1.60e^{\frac{-j2\pi*0*2}{16}} - 3.14e^{\frac{-j2\pi*0*3}{16}}$$

$$+ 4.05e^{\frac{-j2\pi*0*4}{16}} - 3.78e^{\frac{-j2\pi*0*5}{16}} + 1.77e^{\frac{-j2\pi*0*6}{16}}$$

$$+ 0.99e^{\frac{-j2\pi*0*7}{16}} - 1.95e^{\frac{-j2\pi*0*8}{16}} + 1.8e^{\frac{-j2\pi*0*9}{16}}$$

$$- 1.52e^{\frac{-j2\pi*0*10}{16}} + 0.57e^{\frac{-j2\pi*0*11}{16}} + 0e^{\frac{-j2\pi*0*12}{16}}$$

$$+ 0.23e^{\frac{-j2\pi*0*13}{16}} - 0.36e^{\frac{-j2\pi*0*14}{16}} + 0.40e^{\frac{-j2\pi*0*15}{16}}$$

$$X_0 = 0.27$$

$$X_1 = \left| 0.08e^{\frac{-j2\pi*1*0}{16}} - 0.47e^{\frac{-j2\pi*1*1}{16}} + 1.60e^{\frac{-j2\pi*1*2}{16}} - 3.14e^{\frac{-j2\pi*1*3}{16}} \right.$$

$$+ 4.05e^{\frac{-j2\pi*1*4}{16}} - 3.78e^{\frac{-j2\pi*1*5}{16}} + 1.77e^{\frac{-j2\pi*1*6}{16}}$$

$$+ 0.99e^{\frac{-j2\pi*1*7}{16}} - 1.95e^{\frac{-j2\pi*1*8}{16}} + 1.8e^{\frac{-j2\pi*1*9}{16}}$$

$$- 1.52e^{\frac{-j2\pi*1*10}{16}} + 0.57e^{\frac{-j2\pi*1*11}{16}} + 0e^{\frac{-j2\pi*1*12}{16}}$$

$$\left. + 0.23e^{\frac{-j2\pi*1*13}{16}} - 0.36e^{\frac{-j2\pi*1*14}{16}} + 0.40e^{\frac{-j2\pi*1*15}{16}} \right|$$

$$X_1 = |0.2 + 0.019i| = \sqrt{(0.2)^2 + (0.019)^2} = \sqrt{0.04 + 0.000361} = 0.201$$

Kepadatan spektral energi dibutuhkan untuk dapat memetakan nilai *FFT* ke *filter bank* yang akan digunakan. Kepadatan spektral energi dapat dihitung menggunakan persamaan 2.5 untuk nilai *FFT* $k = 0, 1, 2 \dots \frac{N}{2} + 1$.

$$P_k = \frac{1}{N} |X_k|^2$$

$$P_0 = \frac{1}{16} (0.27)^2 = 4.556 * 10^{-3}$$

3.4.4. Mel filter bank

Hasil dari *Fast Fourier Transform* akan disaring menggunakan *mel scale* dengan bantuan *triangular filter bank* untuk mengetahui energi yang tersedia pada setiap titik. *Mel scale* akan memiliki jarak linear untuk frekuensi berada di bawah 1 kilohertz dan

logaritmik untuk frekuensi di atas atau sama dengan 1 kilohertz. Nilai frekuensi terendah dan tertinggi yang dapat dimiliki oleh sinyal suara diubah ke dalam nilai *mel frequency* menggunakan persamaan 2.6 sebelum dapat digunakan untuk membuat daftar f_c yang merupakan daftar titik tengah untuk setiap *filter bank* yang dibutuhkan.

$$mel = 2595 \log_{10} \left(1 + \frac{f}{700} \right) \quad f_{low} = 0, f_{high} = 8000$$

$$mel_{low} = 2595 \log_{10} \left(1 + \frac{0}{700} \right) = 0$$

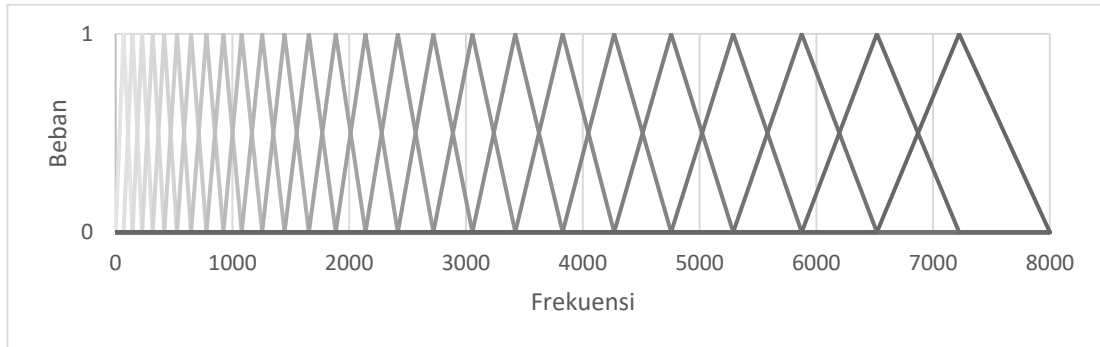
$$\begin{aligned} mel_{high} &= 2595 \log_{10} \left(1 + \frac{8000}{700} \right) \\ &= 2595 * 1.0944 \\ &= 2840.023 \end{aligned}$$

Tabel 3.3 Nilai untuk pembuatan *filter bank*

c	1	2	3	4	5	6	7
f_c (mel)	0,00	105,19	210,37	315,56	420,74	525,93	631,12
f_c (Hz)	0,00	68,48	143,66	226,19	316,80	416,27	525,47
f_c (bin)	0	0	0	0	0	0	0
c	8	9	10	11	12	13	14
f_c (mel)	736,30	841,49	946,67	1051,86	1157,05	1262,23	1367,42
f_c (Hz)	645,35	776,97	921,46	1080,08	1254,22	1445,40	1655,27
f_c (bin)	0	0	0	1	1	1	1
c	15	16	17	18	19	20	21
f_c (mel)	1472,60	1577,79	1682,98	1788,16	1893,35	1998,53	2103,72
f_c (Hz)	1885,69	2138,64	2416,33	2721,20	3055,88	3423,31	3826,69
f_c (bin)	2	2	2	2	3	3	4
c	22	23	24	25	26	27	28
f_c (mel)	2208,91	2314,09	2419,28	2524,46	2629,65	2734,84	2840,02
f_c (Hz)	4269,52	4755,68	5289,39	5875,32	6518,57	7224,74	8000,00
f_c (bin)	4	5	5	6	6	7	8

Nilai untuk setiap f_c dapat dihitung menggunakan nilai mel_{low} dan mel_{high} dengan jarak yang sama. Tabel 3.3 menunjukkan nilai titik tengah untuk pembuatan 26 *filter bank* dengan frekuensi terendah sebesar 0 Hz dan frekuensi tertinggi sebesar 8000

Hz. Gambar 3.2 menunjukkan *filter bank* yang dapat dibuat dengan menggunakan setiap nilai f_c yang diperoleh pada Tabel 3.3.



Gambar 3.2 *Filter bank* MFCC

Sinyal suara pada setiap *frame* dapat disaring dengan menggunakan persamaan 2.8 setelah nilai titik tengah untuk setiap *filter bank* didapatkan.

$$\begin{aligned}
 D'_m &= \ln \left(\sum_{k=0}^{N-1} |D_k| \cdot H(k, m) \right) \\
 D'_8 &= \ln \left(\sum_{k=0}^{N-1} |D_k| \cdot H(k, 8) \right) \\
 &= \ln(0.004 * 1 + 0.002 * 0 + 0.034 * 0 + 0.071 * 0 + 0.09 * 0 + 0.647 \\
 &\quad * 0 + 9.475 * 0 + 16.804 * 0 + 3.136 * 0) \\
 &= -5.521
 \end{aligned}$$

3.4.5. Discrete Cosine Transform

Discrete Cosine Transform digunakan untuk mengubah *mel spectrum* ke dalam frekuensi waktu dan mendapatkan nilai untuk setiap koefisien dengan menggunakan persamaan 2.9.

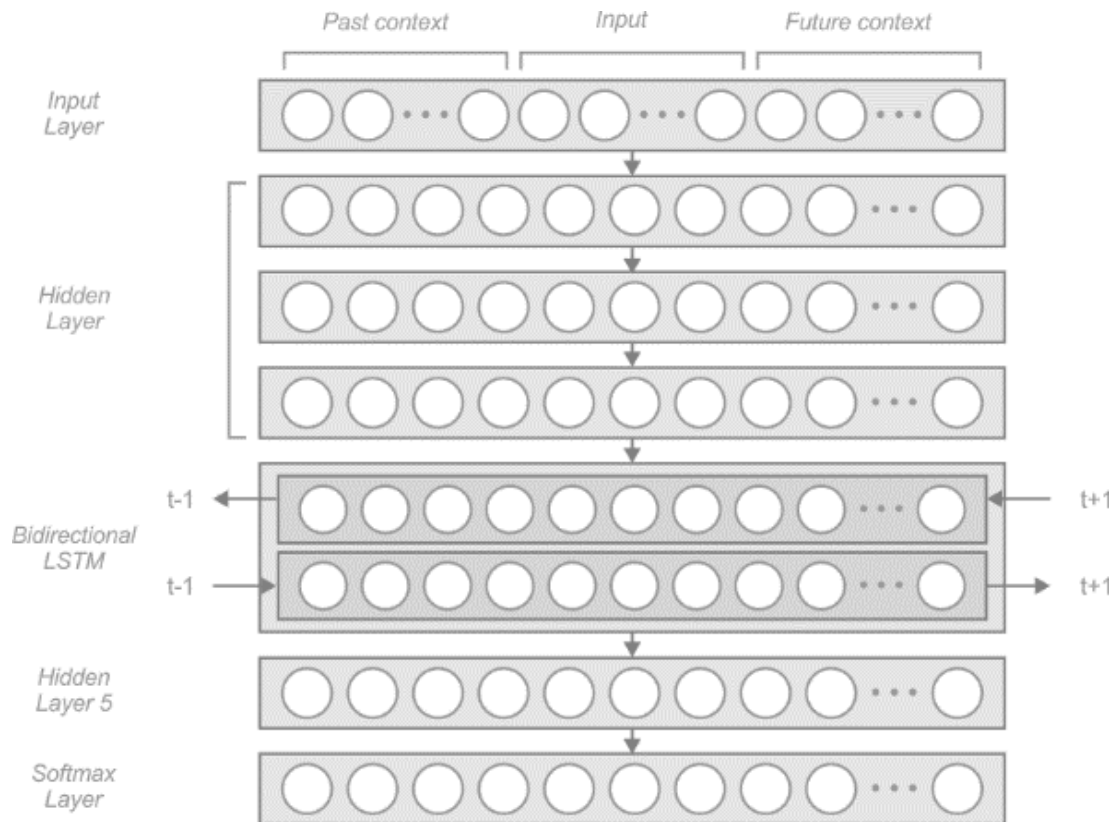
$$c_n = \sum_{k=1}^K D'_k * \cos \left[n(k - 0.5) \frac{\pi}{K} \right]$$

$$\begin{aligned}
c_0 &= \sum_{k=1}^K D'_k * \cos \left[0(k - 0.5) \frac{\pi}{K} \right] = \sum_{k=1}^K D'_k \\
&= -36,044 - 36,044 - 36,044 - 36,044 - 36,044 - 36,044 - 36,044 \\
&\quad - 36,044 - 5,386 - 36,044 - 36,044 - 36,044 - 5,952 \\
&\quad - 36,044 - 36,044 - 36,044 - 3,354 - 36,044 - 2,639 \\
&\quad - 36,044 - 2,333 - 36,044 - 0,435 - 36,044 + 2,249 \\
&\quad + 2,822 \\
&= -1327,185
\end{aligned}$$

3.5. Implementasi Jaringan *Bidirectional LSTM*

Jaringan yang digunakan dapat dilihat pada Gambar 3.3. *Layer* yang digunakan pada jaringan saraf tiruan adalah *layer input* dengan *neuron* sebanyak *cepstral* yang digunakan, lima *hidden layer*, dan satu *layer output softmax* dengan 28 unit yang terdiri dari 26 unit untuk label huruf *a* hingga *z*, satu unit untuk menandakan label spasi dan satu unit untuk menandakan label *blank*. *Hidden layer* yang digunakan terdiri dari tiga *linear hidden layer*, satu *hidden layer* blok *bidirectional LSTM*, dan satu *linear hidden layer* (Hannun, A. *et al.*, 2014). Fungsi aktivasi yang digunakan untuk semua *neuron* pada jaringan adalah *Clipped Rectified Linear Unit (ReLU)* dengan nilai *clipping* 20. Setiap *layer* memiliki nilai *dropout* untuk mengabaikan *neuron* secara acak pada *layer* tersebut.

Input yang diberikan kepada jaringan adalah nilai *MFCC* pada waktu *t*, sembilan nilai *MFCC* dari *t - 9* hingga *t - 1* (*past context*), dan sembilan nilai *MFCC* untuk frame *t + 1* hingga *t + 9* (*future context*). *Layer output* pada jaringan akan menggunakan fungsi aktivasi *softmax* untuk menghasilkan nilai probabilitas setiap label yang dapat diidentifikasi pada waktu *t*.



Gambar 3.3 Arsitektur jaringan

Performa jaringan *long short term memory* dapat dinilai dengan nilai *loss* pada setiap *step* yang merupakan hasil dari perhitungan fungsi objektif dengan menggunakan persamaan 2.30 pada hasil *labelling* dengan nilai probabilitas tertinggi. Teknik *early stopping* akan digunakan pada proses *training* untuk mencegah *overfitting* pada jaringan *bidirectional LSTM*, yaitu dengan membandingkan nilai *loss* pada *dataset validation* pada empat *epoch* terakhir. Frekuensi pengecekan nilai *loss* pada *dataset validation* diatur menggunakan parameter *validation step*.

3.6. Connectionist Temporal Classification

Connectionist Temporal Classification (CTC) akan melakukan *decoding* dengan menggunakan metode *prefix beam search* untuk hasil dari jaringan *LSTM* sesuai dengan probabilitas yang dihasilkan dari *softmax layer* dari jaringan *LSTM* dan *language model* yang disediakan sesuai dengan yang dijelaskan pada bagian 2.8.

Tabel 3.4 adalah contoh hasil dari jaringan *bidirectional LSTM* untuk waktu satu hingga enam dengan spasi dilambangkan dengan simbol `_` dan *blank* dilambangkan

dengan simbol *. Hasil yang akan diambil dari jaringan *bidirectional LSTM* untuk setiap waktu adalah *label* dengan tingkat probabilitas tertinggi sehingga dapat disimpulkan bahwa output dari jaringan adalah *IIYA ***. Label yang sama pada output Tabel 3.4 akan digabungkan menjadi satu dan label *blank* dihapus dari output tersebut.

$$P(IIYA **) = P(IYA)$$

Tabel 3.4 Contoh Output Softmax Layer untuk Setiap Waktu

T	Peluang ($\times 10^{-1}$)																												Out put
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	-	*	
1					1				5							4													I
2	2								6		1																1		I
3	3								1				2												4				Y
4	7								2																1				A
5																											5	6	*
6	1																										2	7	*

Label dengan nilai peluang yang sangat rendah

Nilai peluang dari *softmax output layer* jaringan *bidirectional LSTM* untuk setiap *time step* pada Tabel 3.4 dan nilai peluang untuk setiap *labelling* berdasarkan *language model* akan digunakan oleh *prefix beam search* untuk menemukan hasil pengenalan ucapan yang paling memungkinkan berdasarkan *pseudocode* pada Gambar 2.12. Tingkat kesalahan hasil jaringan *bidirectional LSTM* dalam melakukan pengenalan ucapan dapat dihitung menggunakan WER (*Word Error Rate*), yaitu dengan menghitung banyaknya penambahan, pengurangan, dan pergantian kata yang dibutuhkan untuk mengubah hasil pengenalan ucapan *h* menjadi target hasil pengenalan ucapan *S'* dengan *Z* sebagai panjang total target.

$$E(h, S') = \frac{1}{Z} \sum_{(x,z) \in S'} ED(h(x))$$

Tabel 3.5 menunjukkan contoh perhitungan *Word Error Rate* (WER) pada hasil pengenalan ucapan. Banyaknya perubahan yang diperlukan (*edit distance*) antara target dan hasil pengenalan ucapan pada Tabel 3.5 adalah sebanyak 4, yaitu kata “apa” yang ditambahkan ke hasil pengenalan ucapan, kata “ar” yang dihapus dari hasil pengenalan ucapan, dan kata “sama” dan “tadi” yang diubah menjadi kata “selamat” dan “pagi”. Dengan menggunakan persamaan 2.32, nilai WER yang didapatkan adalah 80 %.

$$E = \frac{1}{5} * 4 * 100 \% = 80 \% ; Z = 5$$

Tabel 3.5 Contoh Perhitungan *Edit Distance* pada Hasil Pengenalan Ucapan

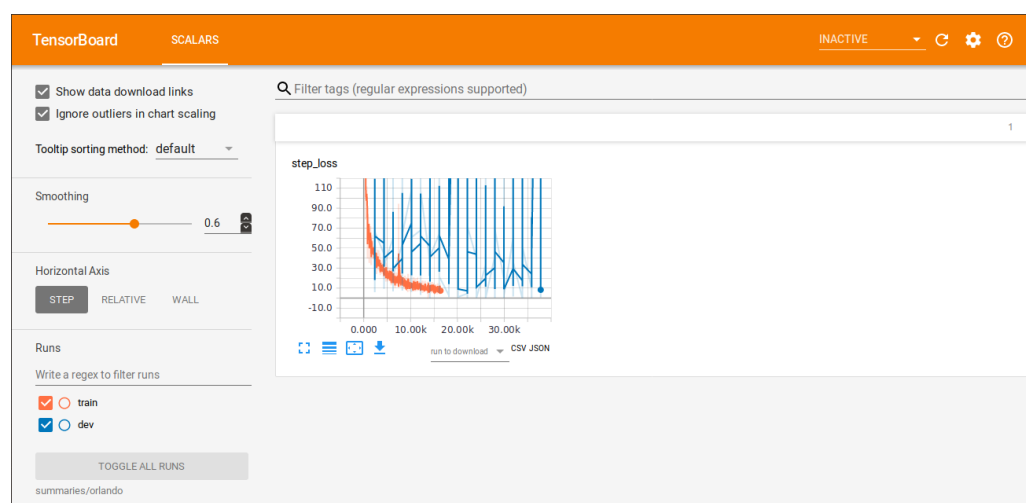
Target	hai <u>selamat</u> <u>pagi</u> apa kabar
Hasil	hal <u>sama</u> <u>tadi</u> kabar ar
Keterangan : <u>Aa</u> → kata yang perlu diubah Aa → kata yang perlu ditambah Aa → kata yang perlu dihapus	

3.7. Perancangan Sistem

Aplikasi yang dikembangkan pada penelitian ini adalah aplikasi untuk melakukan *training* dan *testing speech recognition* berbahasa Indonesia menggunakan *checkpoint* dari hasil *training*.

3.7.1. Perancangan sistem training

Perancang sistem untuk *training* tidak menggunakan *graphical user interface* (GUI) dan hanya digunakan untuk melatih jaringan yang akan digunakan, namun pengguna dapat menggunakan *Tensorboard* untuk melihat perkembangan jaringan saraf pada setiap *step*. Gambar 3.4 menunjukkan tampilan antarmuka *Tensorboard* yang digunakan untuk menampilkan *training loss* dan *testing loss* pada setiap *step*.



Gambar 3.4 Tampilan *Tensorboard* pada saat melakukan *training*

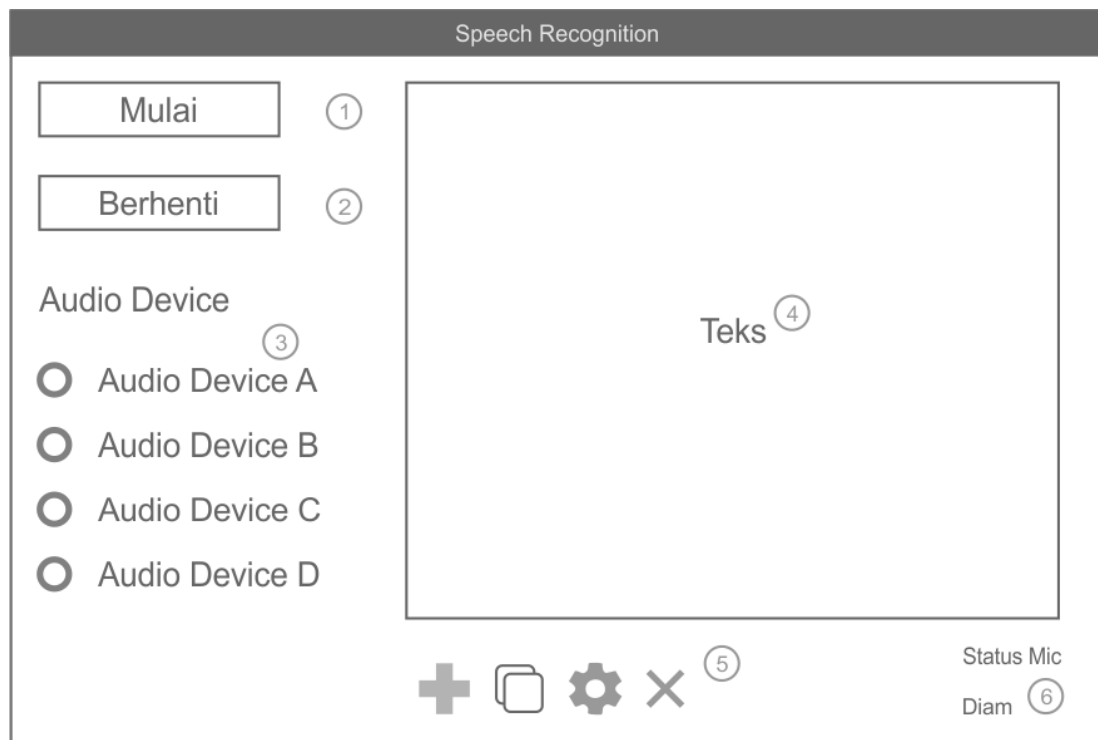
Pengguna dapat melihat *logs* yang ditampilkan ke layar pada saat melakukan pelatihan untuk jaringan *bidirectional long short-term memory*. Gambar 3.5 menunjukkan *logs* yang ditampilkan pada *terminal* berupa proses yang sedang dijalankan dan *loss* yang dihasilkan pada setiap *epoch*.

```
Preprocessing ['../idn-speech/data/niijp/niijp_ds_train.csv']
Loaded from cache at caches/train.hdf5
Preprocessing ['../idn-speech/data/niijp/niijp_ds_valid.csv']
Loaded from cache at caches/dev.hdf5
I STARTING Optimization
I Training epoch 0...
I Training of Epoch 0 - loss: 184.345404
I Training epoch 1...
I Training of Epoch 1 - loss: 92.584780
I Training epoch 2...
I Training of Epoch 2 - loss: 69.363743
I Training epoch 3...
I Training of Epoch 3 - loss: 57.594128
I Training epoch 4...
I Training of Epoch 4 - loss: 49.986641
I Training epoch 5...
I Training of Epoch 5 - loss: 44.389731
I Validating epoch 5...
I Validation of Epoch 5 - loss: 63.970230
I Training epoch 6...
I Training of Epoch 6 - loss: 40.123819
I Training epoch 7...
I Training of Epoch 7 - loss: 36.512690
I Training epoch 8...
I Training of Epoch 8 - loss: 33.606283
```

Gambar 3.5 *Logs* yang disediakan dan ditampilkan ke *terminal*

3.7.2. Perancangan aplikasi pengguna

Perancangan aplikasi pengguna dikembangkan sehingga pengguna dapat mencoba dan melakukan *automatic speech recognition* menggunakan *checkpoint* yang telah dihasilkan dari proses *training*. Tampilan untuk aplikasi pengguna dapat dilihat pada Gambar 3.6.



Gambar 3.6 Rancangan tampilan aplikasi pengguna

Penjelasan dari rancangan aplikasi yang digunakan adalah sebagai berikut.

1. Tombol Mulai pada aplikasi akan memulai proses *automatic speech recognition* dan mendengar suara yang diterima dari *audio device*.
2. Tombol Berhenti akan menghentikan proses *automatic speech recognition* dan *status mic* dan jaringan akan diperbaharui.
3. Pilihan *audio device* digunakan untuk memilih sumber *input* suara yang akan digunakan. Aplikasi akan mengeluarkan sebuah *pop-up* jika pengguna belum memilih *audio device* yang akan digunakan.
4. Sebuah kotak teks di tengah berfungsi untuk menampilkan hasil dari proses *automatic speech recognition* menggunakan metode *bidirectional LSTM* dan *connectionist temporal classification*.
5. Sebuah daftar tombol yang terdiri dari:
 - a. Tombol Tambah untuk menambahkan ucapan yang ingin diubah ke dalam teks.
 - b. Tombol Salin untuk menyalin seluruh hasil pengenalan ucapan ke dalam *clipboard*.
 - c. Tombol Pengaturan untuk menampilkan sebuah *dialog* untuk mengubah *mic threshold* yang digunakan.

- d. Tombol Hapus untuk menghapus seluruh hasil pengenalan ucapan yang telah dilakukan
6. *Status mic* akan menampilkan teks “Mendengar” ketika aplikasi mendengar suara pengguna dan “Diam” jika aplikasi hanya mendengar suara diam.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

Bab ini akan membahas hasil implementasi metode yang telah diajukan, yaitu *connectionist temporal classification* dengan menggunakan *mel frequency cepstral coefficient* sebagai metode untuk melakukan *feature extraction* dan *bidirectional long-short term memory* sebagai metode untuk melakukan *framewise classification* yang telah dibahas pada Bab 3.

4.1. Spesifikasi Perangkat Keras dan Perangkat Lunak

Spesifikasi perangkat keras yang digunakan untuk pengembangan aplikasi pada penelitian ini adalah sebagai berikut.

1. Prosesor *Intel® Core™ i5-4200U*
2. *Hard drive* dengan kapasitas sebesar *500 GB*
3. Kapasitas memori *RAM* sebesar *12 GB*
4. *Blue Yeti USB Microphone*
5. *Microphone Taffware MK-F100TL*

Spesifikasi perangkat keras yang digunakan untuk proses *training* jaringan *bidirectional LSTM* adalah sebagai berikut..

1. Prosesor *Intel® Xeon E5 v3 (Haswell) x 2 core*
2. *Hard drive* dengan kapasitas sebesar *500 GB*
3. Memori *RAM* sebesar *16 GB*
4. *Graphical Processing Unit (GPU) NVIDIA GeForce GTX1030 DDR4 8GB*
5. *Sistem operasi Linux (Ubuntu 18.04 LTS Bionic Weaver amd64)*

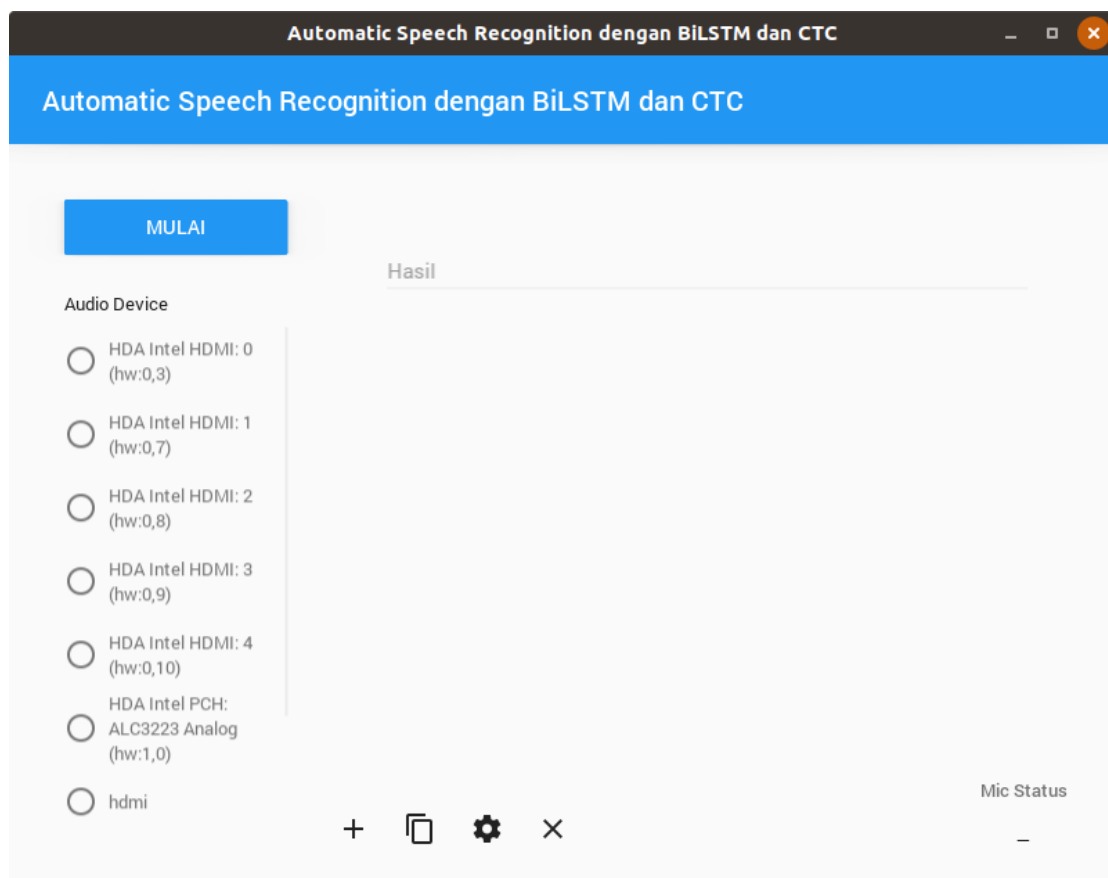
Spesifikasi perangkat lunak yang digunakan untuk pengembangan aplikasi pada penelitian ini adalah sebagai berikut.

1. *Sistem operasi Linux (Ubuntu 18.04 LTS Bionic Weaver amd64)*
2. *PyCharm Community 2018.1.4.*
3. *Python* versi *3.6*

4. *Library tensorflow, numpy, scipy, kenlm, deepspeech, pysndfx*, dan *python_speech_feature* untuk ekstraksi fitur dan pelatihan jaringan saraf tiruan.
5. *Library Kivy, KivyMD, PyAudio*, dan *xenon* untuk pembuatan GUI aplikasi dan *PyAudio* untuk mengatur dan memproses suara dari *input* mikrofon.

4.2. Implementasi Antarmuka Aplikasi

Implementasi antarmuka aplikasi dibangun berdasarkan perancangan aplikasi pengguna pada Gambar 3.6. Antarmuka yang disediakan terdiri dari dua tampilan, yaitu halaman utama dan *dialog* pengaturan. Ilustrasi halaman utama aplikasi dapat dilihat pada Gambar 4.1.



Gambar 4.1 Tampilan antarmuka aplikasi

Pengguna harus memilih *audio device* (perangkat audio) yang ingin digunakan sebagai sumber suara sebelum dapat memulai proses *automatic speech recognition*. Untuk memulai proses *automatic speech recognition*, pengguna harus menekan tombol

“Mulai”. Aplikasi akan merekam suara melalui *audio device* yang telah ditentukan dan menyimpan suara yang didapatkan melalui *audio device* tersebut jika suara yang diperoleh lebih tinggi dibandingkan dengan *threshold* yang telah ditentukan. Suara yang disimpan adalah suara yang ditangkap pada saat suara tersebut lebih tinggi dari *threshold* yang telah ditentukan, satu detik sebelum dan sesudah suara yang melebihi *threshold* terekam, dan satu detik suara hening sebelum dan sesudah suara tersebut terekam. *Threshold* suara yang ditangkap dapat diatur dengan membuka *dialog* pengaturan melalui tombol roda gigi pada aplikasi. Suara yang direkam akan dihapus secara otomatis setelah proses *automatic speech recognition* untuk suara tersebut selesai dilakukan dan hasil dari proses *automatic speech recognition* akan ditampilkan pada *textfield* yang ditampilkan pada bagian kanan atas aplikasi. Untuk menghentikan proses *automatic speech recognition* dan berhenti merekam suara melalui *audio device* yang telah ditentukan, pengguna dapat menekan tombol “Berhenti” pada bagian kiri atas aplikasi.

Tombol salin pada aplikasi berfungsi untuk menyalin seluruh teks pada *textfield* yang merupakan hasil dari seluruh *automatic speech recognition* ke *clipboard*. Aplikasi juga menyediakan tombol hapus untuk menghapus seluruh teks yang tersedia pada *textfield*. *Mic status* yang terletak pada bagian kanan bawah aplikasi berfungsi untuk menampilkan status *audio device* pada saat proses *automatic speech recognition* berlangsung.

4.3. Pembuatan *Language Model*

Language model dibutuhkan untuk melakukan *decoding* pada output jaringan *bidirectional LSTM*. Pembuatan *language model* membutuhkan *corpus* teks yang lebih besar daripada transkrip teks pada *dataset* suara agar *language model* dapat memiliki nilai probabilitas yang tepat untuk setiap kata pada *corpus*. *KenLM* (Heafield, 2011) digunakan untuk membuat sebuah *5-gram language model* dengan *corpus* teks dari surat kabar pada tahun 2016 (Goldhahn *et al.*, 2012). Teks yang digunakan pada proses pembuatan *language model* hanya terdiri dari karakter alfabet dan spasi yang didapatkan melalui proses *cleaning*, yaitu menyaring seluruh simbol dan mengubah seluruh angka ke dalam bentuk kata dan frasa. Gambar 4.2 menunjukkan langkah pembuatan *language model* menggunakan *KenLM* menggunakan teks yang telah dibersihkan.

```

(deepspeech) zenith@inspiron:~/Documents/kenlm/build$ bin/lmplz -o 5 --prune 0 0 0 1 --text ../../DeepSpeech/news_pruned.txt --arpa ../../DeepSpeech/models/ind.2.arpa
=== 1/5 Counting and sorting n-grams ===
Reading /home/zenith/Documents/DeepSpeech/news_pruned.txt
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
*****
Unigram tokens 208546 types 21481
=== 2/5 Calculating and sorting adjusted counts ===
Chain sizes: 1:257772 2:973635456 3:1825566592 4:2920906240 5:4259655424
Statistics:
1 21481 D1=0.650885 D2=1.06467 D3+=1.46053
2 130387 D1=0.822548 D2=1.18418 D3+=1.39346
3 185566 D1=0.942115 D2=1.36323 D3+=1.46073
4 3267/191419 D1=0.980684 D2=1.39894 D3+=1.4722
5 1258/185292 D1=0.989813 D2=1.51711 D3+=1.43172
Memory estimate for binary LM:
type      kB
probing   8049 assuming -p 1.5
probing  10003 assuming -r models -p 1.5
trie      4112 without quantization
trie      2284 assuming -q 8 -b 8 quantization
trie      2696 assuming -a 22 array pointer compression
trie      1868 assuming -a 22 -q 8 -b 8 array pointer compression and quantization
=== 3/5 Calculating and sorting initial probabilities ===
Chain sizes: 1:257772 2:2086192 3:3711320 4:78408 5:35224
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
#####
=== 4/5 Calculating and writing order-interpolated probabilities ===
Chain sizes: 1:257772 2:2086192 3:3711320 4:78408 5:35224
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100
#####
=== 5/5 Writing ARPA model ===
----5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80---85---90---95---100

```

Gambar 4.2 Pembuatan *language model* menggunakan KenLM

Gambar 4.3 menunjukkan 16 baris pertama hasil pembuatan *language model* yang disimpan ke dalam bentuk *file ARPA*. *Format language model* yang dihasilkan terdiri dari `\data\` yang memberikan banyak entri untuk masing-masing *N-gram*. *N* bagian selanjutnya memberikan kata dan pasangan yang terdapat pada *corpus* teks yang digunakan. Setiap baris *n-gram* terdiri dari nilai peluang dalam bentuk \log_{10} , kata atau pasangan, dan nilai *backoff weight*.

```

\data\
ngram 1=21481
ngram 2=130387
ngram 3=185566
ngram 4=3267
ngram 5=1258

\1-grams:
-5.1343307      <unk>      0
0              <s>      -0.586059
-1.5129033      </s>      0
-2.8330736      kami      -0.25503045
-3.2609499      memang    -0.20292047
-4.7179112      nggak     -0.08483864
-3.541244       punya     -0.12637474
-3.5650494      alat      -0.17814374

```

Gambar 4.3 Hasil pembuatan *language model* menggunakan KenLM

Language model yang diperoleh menggunakan *KenLM* diubah ke dalam bentuk *file binary* untuk meningkatkan performa *language model* dalam proses *automatic speech recognition*. Gambar 4.4 menunjukkan proses yang dilakukan untuk mengubah *language model* yang telah dihasilkan ke dalam bentuk *file binary*.

```
(deepspeech) zenith@inspiron:~/Documents/kenlm/build$ bin/build_binary -T -s ../
../DeepSpeech/models/ind.2.arpa ../DeepSpeech/models/ind.2.binary
Reading ../DeepSpeech/models/ind.2.arpa
---5---10---15---20---25---30---35---40---45---50---55---60---65---70---75---80
---85---90---95---100
*****
*****
SUCCESS
```

Gambar 4.4 Konversi *language model* ke dalam bentuk *file binary*

4.4. Pelatihan Sistem

Pelatihan dijalankan dengan menggunakan *dataset training* dan *parameter* pada Tabel 4.1 dengan *dataset training* sebanyak 78304 suara yang diperoleh dari *Tokyo Institute of Technology Multilingual Speech Corpus*, data pribadi, dan hasil dari augmentasi suara dengan modifikasi *tempo* dan *speed* sebanyak 90 % hingga 120 %. *Epoch* adalah banyaknya perulangan yang dilakukan dengan *dataset* yang digunakan untuk melatih jaringan *LSTM*. *Batch size* mengatur banyaknya data yang digunakan dalam mengeksekusi satu *step*. *Learning rate* adalah salah satu parameter yang mengatur perubahan beban pada jaringan. *Validation step* menentukan frekuensi pengecekan *loss* dengan menggunakan *dataset validation*.

Tabel 4.1 Parameter Pelatihan Jaringan *Bidirectional LSTM*

Parameter	Percobaan				
	A	B	C	D	E
Unit pada setiap <i>hidden layer</i>	128	300	256	200	128
<i>Epoch</i>	100	100	100	100	100
<i>Train batch size</i>	70	100	200	200	80
<i>Validation batch size</i>	1	1	1	1	1
<i>Testing batch size</i>	1	1	1	1	1
<i>Learning Rate</i>	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-4}

Parameter	Percobaan				
	A	B	C	D	E
<i>Validation step</i>	2	5			

Sistem akan melakukan *preprocessing* dengan menggunakan *mel frequency cepstral coefficient* dan menambah *zero padding* sebagai *past context* dan *future context* kosong. Hasil dari *preprocessing* akan disimpan ke dalam sebuah file sebagai *cache* untuk mempercepat eksekusi percobaan selanjutnya.

Step adalah perulangan pada setiap *epoch* untuk memproses seluruh data yang terdapat pada *dataset* tersebut. Pada percobaan A, banyak *step* pada setiap *epoch training* adalah 1119 dengan 78304 banyak data yang digunakan pada *dataset training* dan *batch size* 70. *Loss* pada setiap *epoch* didapatkan dengan menghitung rata-rata nilai *loss* pada setiap *step* pada *epoch* tersebut.

Proses *training* akan dilanjutkan setelah ekstraksi fitur selesai dieksekusi. Proses *training* akan dihentikan jika telah mencapai banyak *epoch* yang diinginkan atau memenuhi syarat *early stopping*, yaitu nilai *loss* dengan *dataset validation* tidak membaik pada empat *validation step* terakhir. Gambar 4.5 adalah contoh *log* pada *terminal* yang ditampilkan pada saat nilai *loss* pada *dataset validation* memburuk selama empat *validation step* terakhir.

```

I Validating epoch 20,,,
I Validation of Epoch 20 - loss: 57,803554
I Training epoch 21,,,
I Training of Epoch 21 - loss: 35,914049
I Training epoch 22,,,
I Training of Epoch 22 - loss: 35,412928
I Validating epoch 22,,,
I Validation of Epoch 22 - loss: 58,719720
I Training epoch 23,,,
I Training of Epoch 23 - loss: 34,982370
I Training epoch 24,,,
I Training of Epoch 24 - loss: 34,576018
I Validating epoch 24,,,
I Validation of Epoch 24 - loss: 59,104309
I Training epoch 25,,,
I Training of Epoch 25 - loss: 34,367155
I Training epoch 26,,,
I Training of Epoch 26 - loss: 33,853112
I Validating epoch 26,,,
I Validation of Epoch 26 - loss: 65,137712
I Early stop triggered as (for last 4 steps) validation loss: 65,137712 with
standard deviation: 0,545612 and mean: 58,542527
I FINISHED Optimization - training time: 11:52:42

```

Gambar 4.5 Contoh *early stopping* pada percobaan A

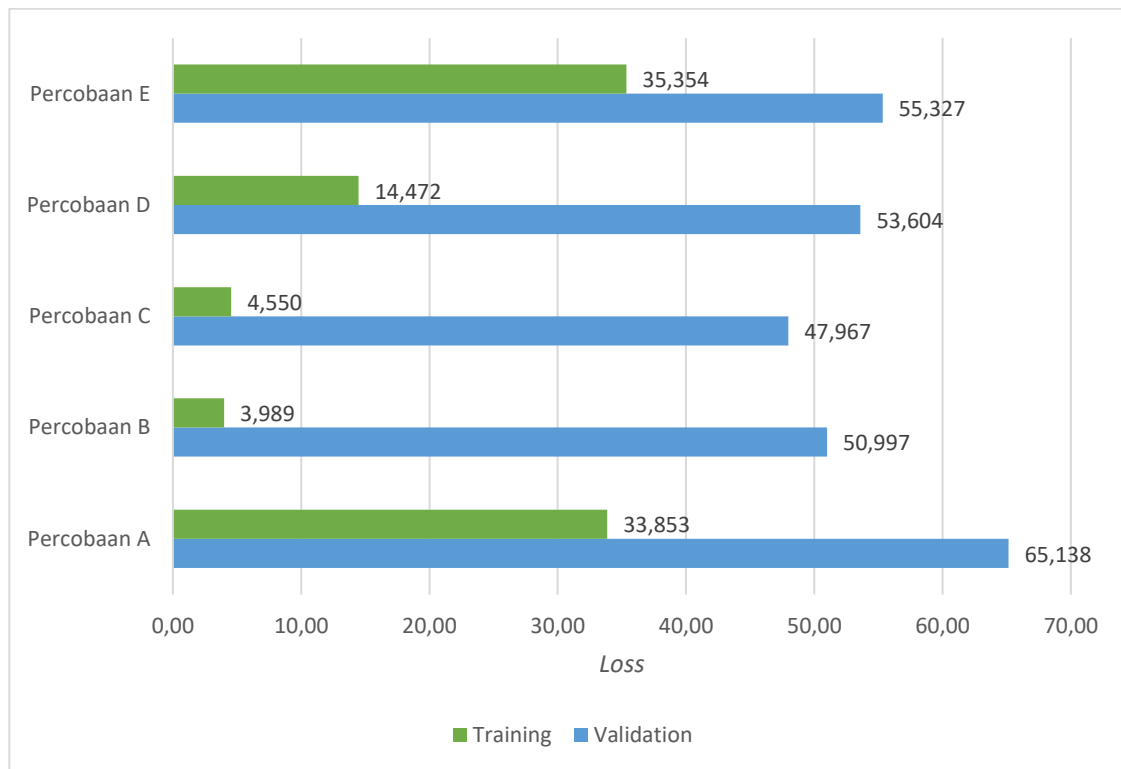
Hasil proses *training* dengan parameter pada Tabel 4.1 dapat dilihat pada Tabel 4.2. Hasil percobaan menunjukkan banyak unit pada setiap hidden layer dapat menurunkan nilai loss pada dataset training, namun tidak menjamin nilai loss yang lebih baik pada dataset yang berbeda dan dapat menyebabkan overfitting dan underfitting pada jaringan bidirectional LSTM. Hal ini dapat dilihat pada percobaan D dengan 200 *neuron* pada setiap *hidden layer* memiliki *train loss* 14,4723 dan *validation loss* 53,6044, sedangkan percobaan A dengan 128 *neuron* memiliki *training loss* 33,8531 dan *validation loss* sebesar 65,1377. Penambahan banyak neuron pada percobaan C menjadi sebanyak 256 *neuron* pada setiap *hidden layer* memberikan hasil *training loss* sebesar 4,5503 dan *validation loss* sebesar 47,9669, namun penambahan banyak neuron pada setiap *hidden layer* memperburuk nilai *validation loss* pada percobaan B. Hal ini menunjukkan bahwa jumlah *neuron* pada *hidden layer* tidak boleh terlalu sedikit dan terlalu banyak untuk mencegah terjadinya *overfitting* dan *underfitting* pada jaringan *bidirectional LSTM*.

Tabel 4.2 Hasil Proses *Training* untuk Setiap Percobaan

Hasil	Percobaan				
	A	B	C	D	E
Unit pada setiap <i>hidden layer</i>	128	300	256	200	128
<i>Learning Rate</i>	10^{-3}	10^{-3}	10^{-3}	10^{-3}	10^{-4}
<i>Actual epoch</i> ¹	27	71	100	46	91
<i>Train loss</i>	33,8531	3,9887	4,5503	14,4723	35,3542
<i>Validation loss</i>	65,1377	50,9969	47,9669	53,6044	55,3272

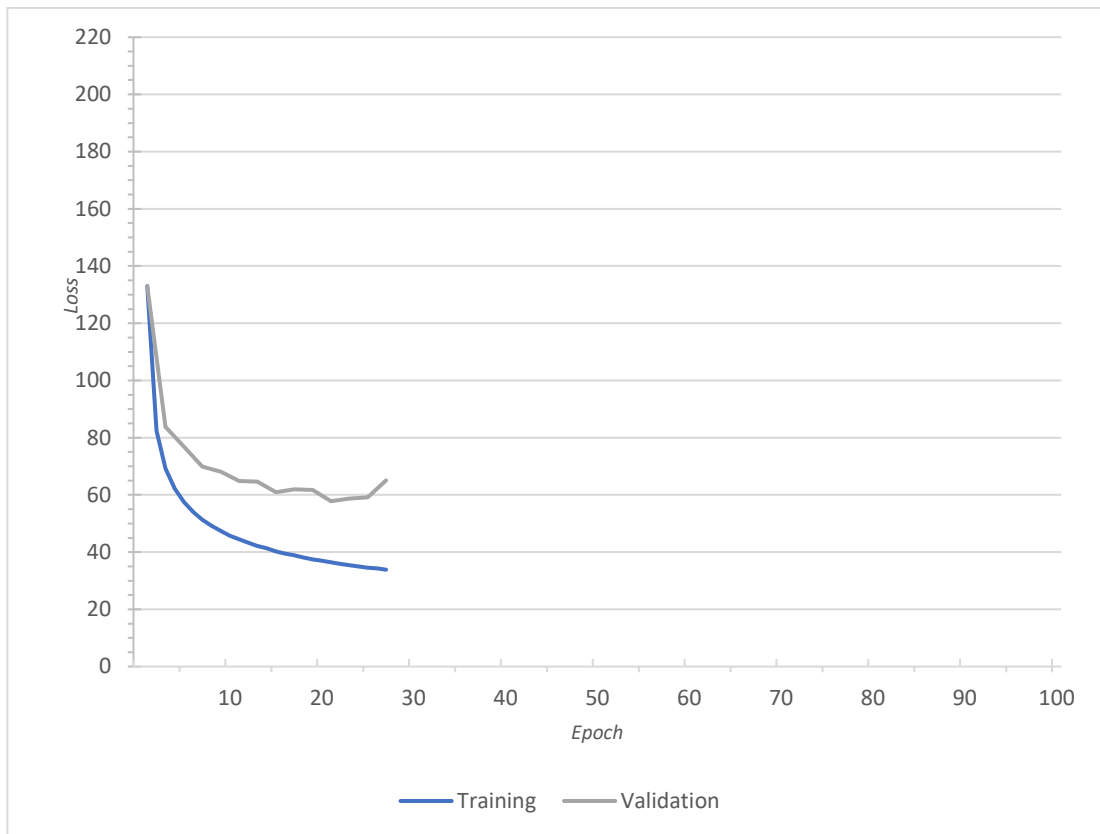
¹ Banyak epoch yang berjalan sebelum *early stopping*

Gambar 4.6 menunjukkan grafik perbandingan nilai *training loss* yang diperoleh pada setiap percobaan.

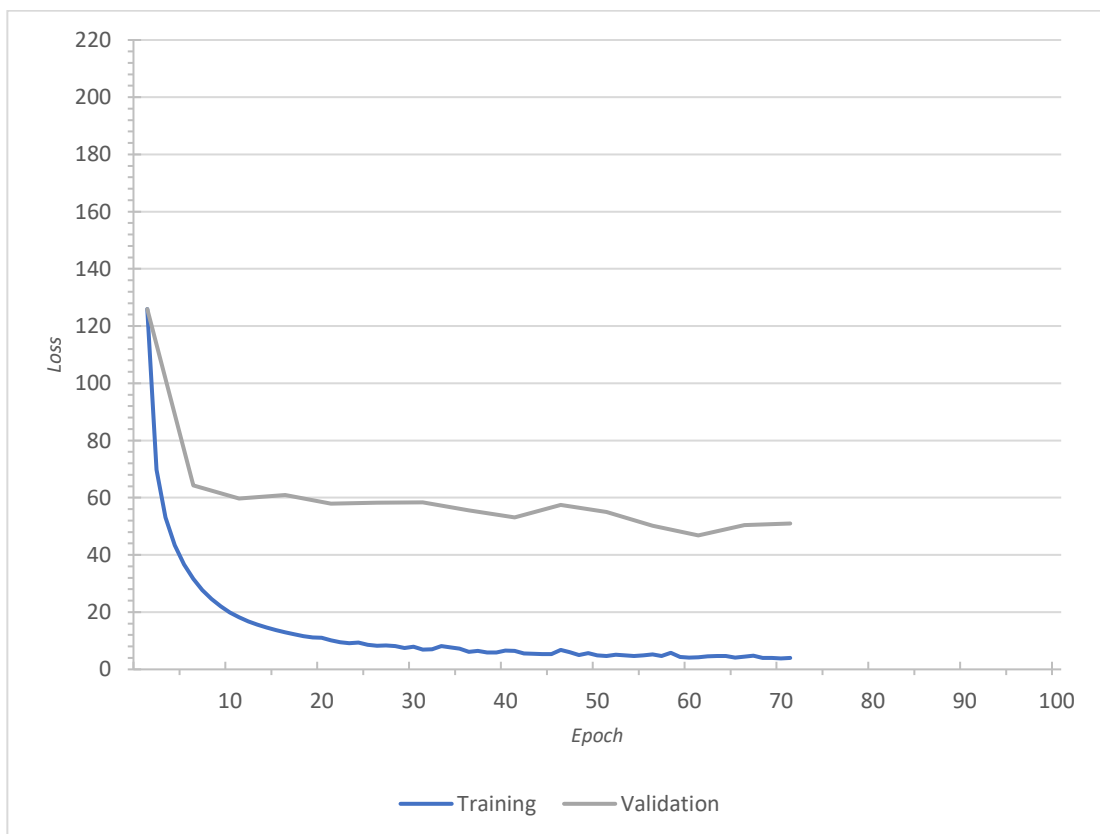


Gambar 4.6 Perbandingan *loss* pada percobaan

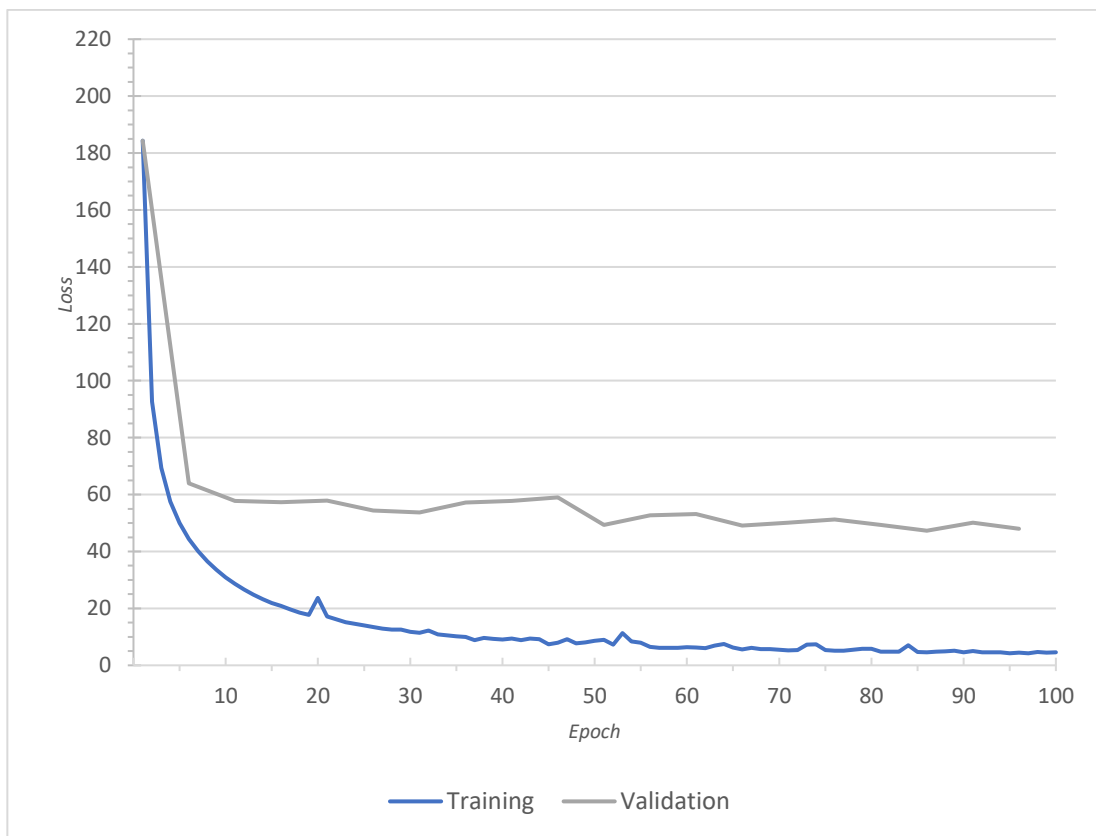
Gambar 4.7 merupakan grafik *loss* proses *training* dan *validation* pada setiap percobaan. Pengaruh *learning rate* terhadap kecepatan generalisasi dapat dilihat melalui percobaan A dengan *learning rate* 10^{-3} yang beradaptasi lebih cepat dan berhenti pada *epoch* 27 dengan nilai *training loss* 33,8531 dibandingkan dengan percobaan E dengan *learning rate* 10^{-4} , *actual epoch* 91, dan *training loss* 35,3542. *Learning rate* dapat digunakan untuk mencegah suatu jaringan *bidrectional LSTM* menyimpulkan hasil klasifikasi untuk suara yang tidak dibutuhkan pada *dataset training* sehingga memberikan hasil yang buruk pada *dataset* lainnya (*overfitting*). Hal ini dapat dilihat pada percobaan E yang memiliki nilai *validation loss* yang lebih baik dibandingkan dengan percobaan A dengan perbedaan sebesar 9,8105.



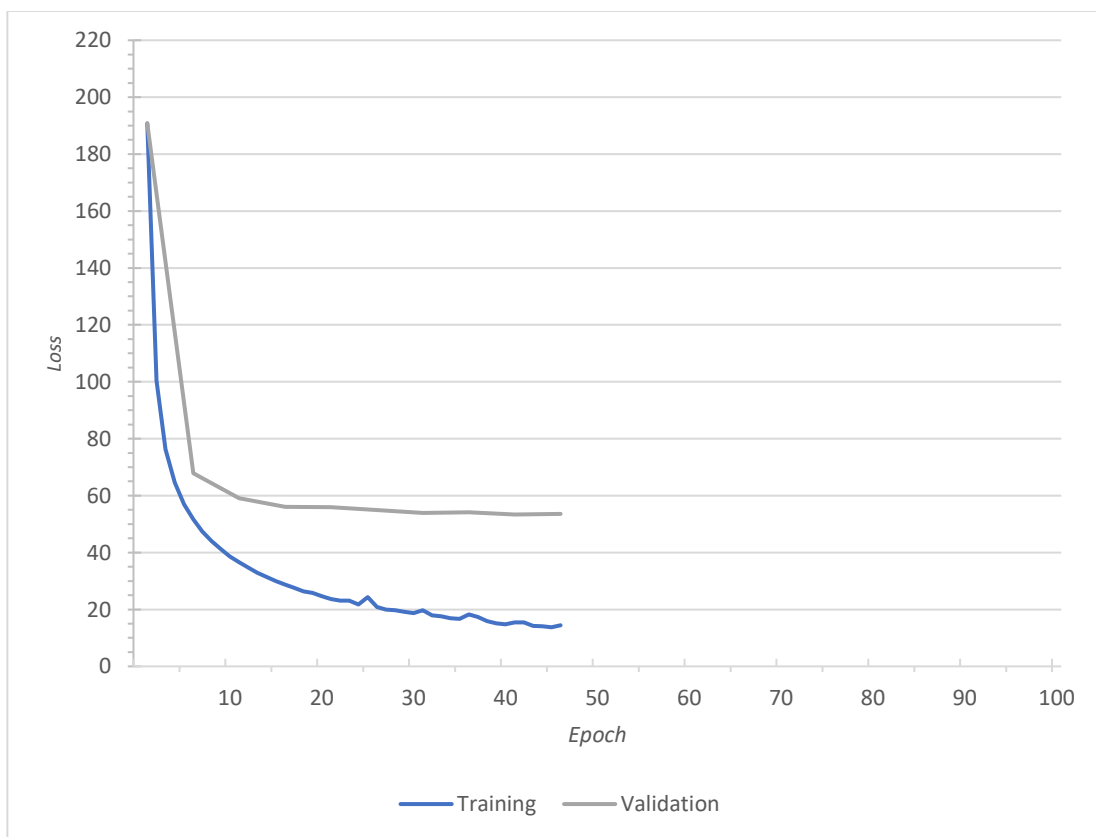
(A)



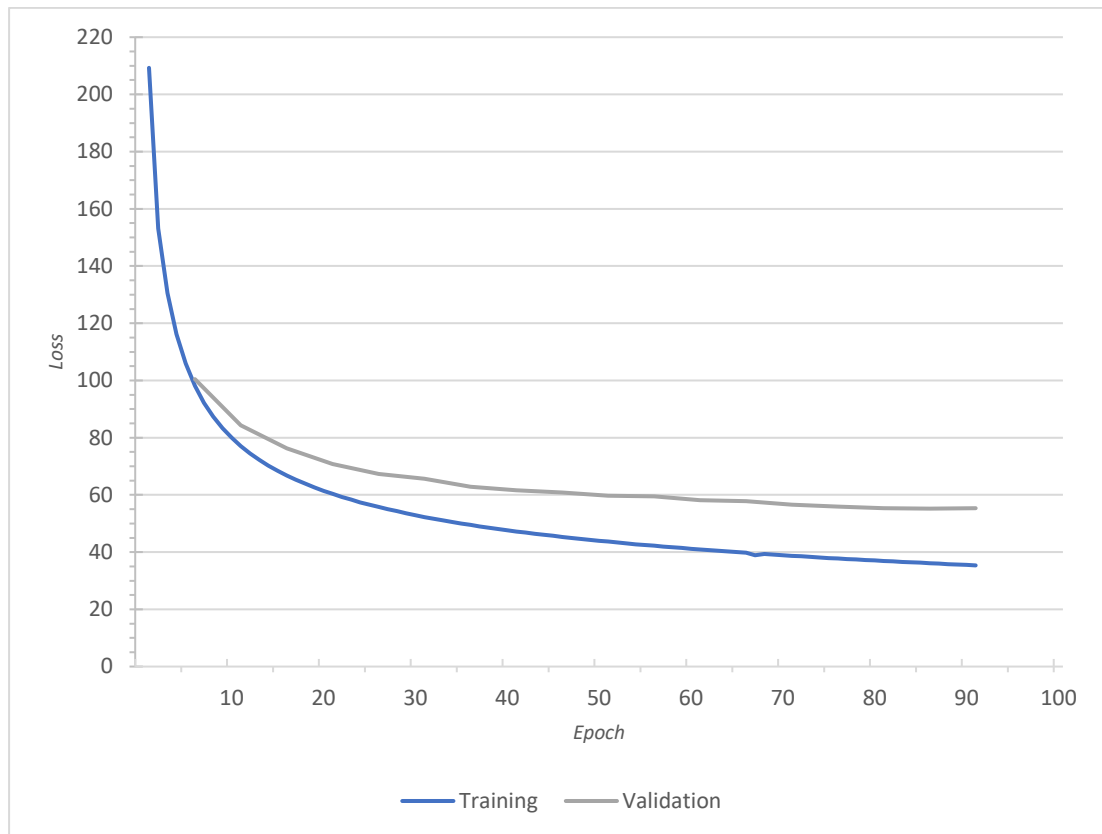
(B)



(C)



(D)



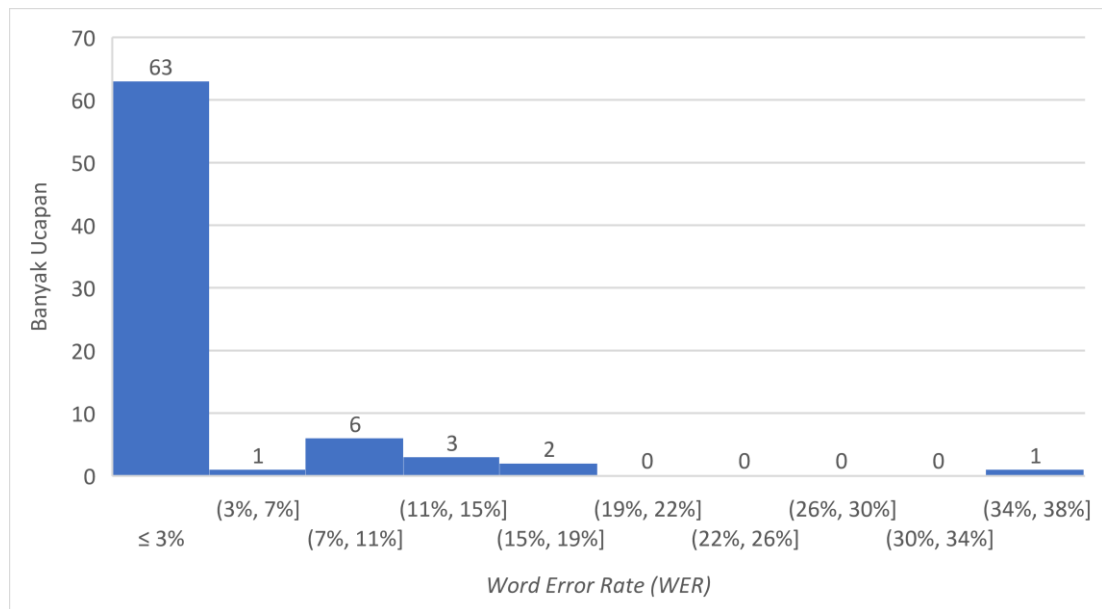
(E)

Gambar 4.7 Grafik perbandingan *loss training* dan *validation* setiap percobaan

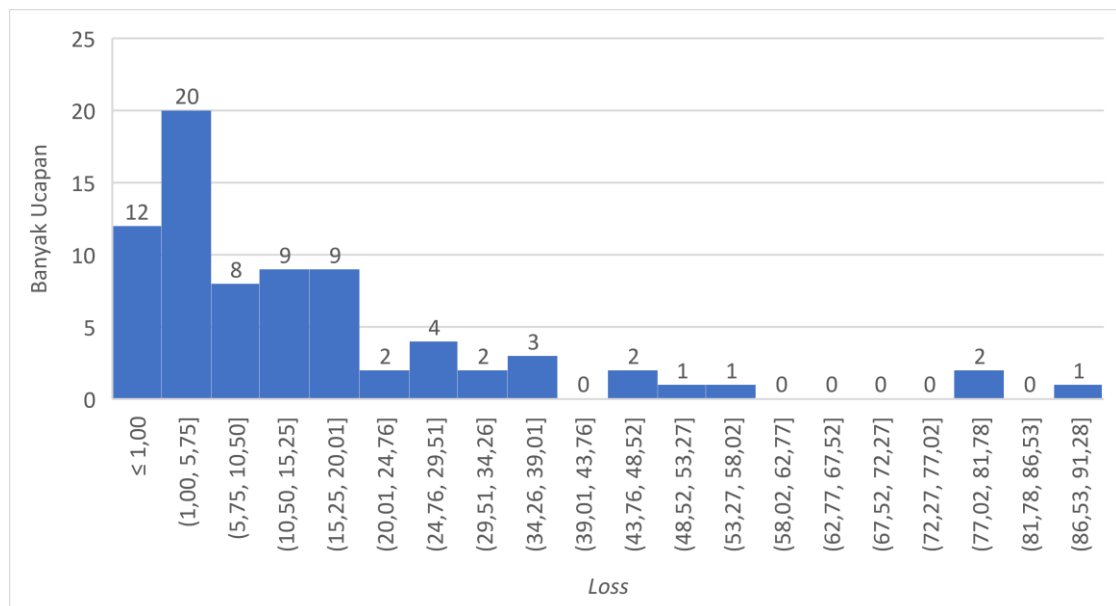
4.5. Pengujian Sistem

Evaluasi kinerja pengenalan ucapan menggunakan jaringan *bidirectional long short term memory (LSTM)* dan *connectionist temporal classification (CTC)* dilakukan setelah proses *training* selesai dilakukan atau setelah terjadinya *early stopping* pada proses *training*. Proses *testing* dilakukan dengan menggunakan *dataset testing* dari *speech corpus TITML* yang telah ditentukan pada bagian 3.1. Proses *testing* melakukan *decoding* dengan menggunakan metode *prefix beam search* yang memanfaatkan *language model* yang telah dihasilkan pada bagian 4.3.

Berdasarkan hasil pelatihan sistem dengan menggunakan parameter pada Tabel 4.1, pengujian sistem dilakukan dengan menggunakan jaringan *bidirectional LSTM* dengan parameter percobaan C yang memiliki nilai *validation loss* terbaik. Hasil pengujian menunjukkan bahwa jaringan *bidirectional LSTM* dapat melakukan pengenalan ucapan otomatis dengan nilai WER rata-rata sebesar 3,01 % dan nilai *loss* rata-rata sebesar 23,3904.



(A)



(B)

Gambar 4.8 Grafik WER (A) dan *testing loss* (B) pada percobaan C

Tabel 4.3 menunjukkan lima hasil terburuk pengenalan ucapan pada pengujian sistem. Hasil *testing* menunjukkan bahwa ucapan nama orang tidak dapat dikenal dengan baik, seperti Kafelnikov, Abdurrahman Wahid, Adi, dan Suadmojo. Jaringan *bidirectional LSTM* melakukan kesalahan dalam pengenalan *label* spasi sehingga menyebabkan WER pada hasil ketiga pada Tabel 4.3 bernilai 50 %.

Tabel 4.3 Lima Hasil Pengenalan Ucapan Terburuk pada Percobaan C

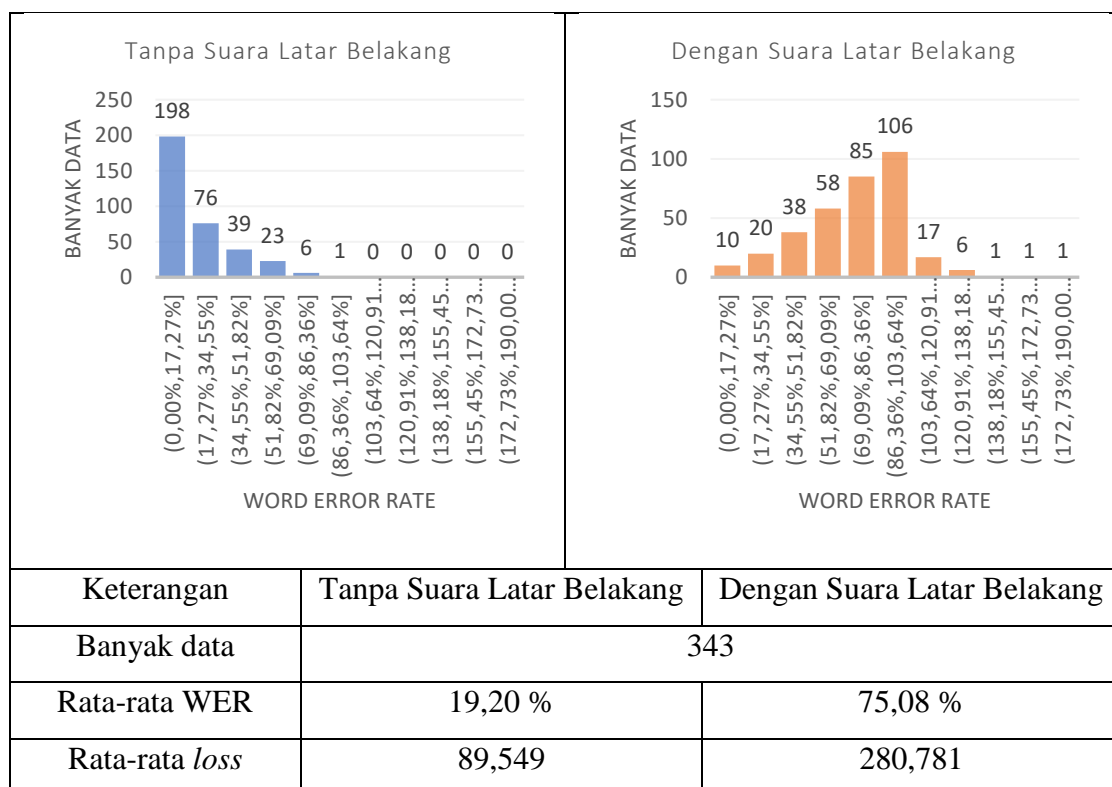
Nomor	Transkrip	Hasil	WER	Loss
1	tahun lalu di final dia mengalahkan petenis rusia yevgeny kafelnikov	palu ani di sidak dia mematahkan petenis rusia yevgeny reli ko	60 %	23,3905
2	kh idris marzuki menyosialisasikan kepada abdurrahman wahid tentang hasil muktamar nu	kh idris marzuki menyosialisasikan kepada aman itu kasi dan laren	54,54 %	181,0441
3	adi pandai bermain alat musik keyboard	hadi pad dai bermain alat musik keyboard	50 %	29,4604
4	dokter musni suadmodjo memaklumi akan terjadinya insiden tersebut	dokter musni lain meju mati di akan terjadinya insiden tersebut	50%	119,2341
5	disbursement atau pencairan pinjaman itu masih tergantung dari beberapa pekerjaan rumah yang harus dikerjakan oleh pemerintah	disbursement atau pencairan ida menit masih bagan pun dari beberapa pekerja dumai harus dikerjakan oleh pemerintah	43,75 %	178,1519

4.5.1. Pengujian sistem pada dataset tambahan

Pengujian sistem dilakukan pada *dataset* tambahan yang terdiri dari suara ucapan 343 kalimat yang direkam pada dua kondisi, yaitu pada suasana hening dan pada ruangan dengan suara latar belakang. Dataset tambahan direkam menggunakan mikrofon yang berbeda dengan dataset *testing*, yaitu *Taffware MK-F100TL*. Hal ini dilakukan untuk menguji apakah jaringan *bidirectional LSTM* mampu melakukan pengenalan ucapan pada *speaker*, perangkat, dan suasana yang berbeda.

Tabel 4.4 menunjukkan grafik perbandingan *WER* pengenalan ucapan antara ucapan pada suasana hening dan ucapan dengan suara latar belakang berupa percakapan pada televisi untuk setiap percobaan. Hasil menunjukkan bahwa jaringan *bidirectional LSTM* dengan *CTC* dapat memberikan hasil pengenalan ucapan yang cukup baik pada suasana hening, yaitu dengan nilai rata-rata *WER* sebesar 19,20 % untuk ucapan dengan tanpa suara latar belakang, namun memberikan hasil yang kurang maksimal pada ucapan dengan suara latar belakang dengan nilai rata-rata *WER* sebesar 75,08 %. Hasil pengujian dengan perangkat mikrofon yang berbeda memberikan hasil yang lebih buruk dibandingkan dengan hasil pengujian dengan menggunakan *dataset testing* yang memberikan nilai rata-rata *WER* sebesar 3,01 %. Hal ini menunjukkan bahwa jaringan *bidirectional LSTM* mengalami *mismatch problem*, yaitu ketidakmampuan model *automatic speech recognition* dalam melakukan pengenalan ucapan pada suara yang direkam pada lingkungan yang berbeda dan suara yang direkam menggunakan perangkat yang berbeda (O'Shaughnessy, 2008).

Tabel 4.4 Hasil Pengujian pada Lingkungan yang Berbeda



BAB 5

KESIMPULAN

Bab ini membahas kesimpulan dari penerapan metode yang diajukan untuk melakukan pengenalan ucapan dan saran yang dapat dipertimbangkan untuk penelitian selanjutnya.

5.1. Kesimpulan

Berdasarkan hasil penelitian dengan metode *bidirectional long short-term memory* dan *connectionist temporal classification* didapatkan kesimpulan, yaitu:

1. Metode *bidirectional Long Short-Term Memory* dan *Connectionist Temporal Classification* dapat digunakan untuk melakukan pengenalan suara tanpa membutuhkan proses segmentasi dan *tuning* suara dengan nilai *WER* terbaik sebesar 3,01 % untuk ucapan pada *dataset testing*.
2. Nilai *loss* yang baik pada *training* tidak selalu memberikan hasil yang baik dengan ucapan pada *dataset validation, testing*, dan ucapan baru.
3. Penerapan *early stopping* pada proses *training* mencegah jaringan *bidirectional long short-term memory* untuk mengalami *overfitting*.
4. Jumlah *neuron* dan nilai *learning rate* yang digunakan pada jaringan *bidirectional Long Short-Term Memory* dapat memengaruhi tingkat kesalahan pada pengenalan ucapan
5. *Learning rate* dapat memengaruhi kecepatan perubahan nilai *loss* setiap *epoch* pada jaringan *bidirectional Long Short-Term Memory* dan mencegah jaringan *bidirectional Long Short-Term Memory* untuk melakukan generalisasi terhadap *noise* yang tidak dibutuhkan.
6. Metode *bidirectional Long Short-Term Memory* dan *Connectionist Temporal Classification* mengalami kesulitan dalam melakukan pengenalan ucapan terhadap kalimat yang mengandung nama orang dan kata asing.

7. Jaringan *bidirectional Long Short-Term Memory* memberikan hasil dengan *WER* yang lebih buruk pada ucapan dengan suara latar belakang dan pada ucapan yang direkam menggunakan perangkat mikrofon yang berbeda. Jaringan *bidirectional LSTM* memberikan nilai *WER* sebesar 19,20 % pada ucapan tanpa suara latar belakang dengan perangkat yang berbeda dan 75,08 % pada ucapan dengan suara latar belakang.

5.2. Saran

Saran yang dapat diberikan oleh penulis untuk penelitian selanjutnya adalah sebagai berikut:

1. Pembuatan dan penggunaan *speech corpus* dengan suara latar belakang, *noise*, dan transkrip yang lebih variatif sehingga jaringan *bidirectional Long Short-Term Memory* dapat memberikan *WER* yang lebih baik. *Speech corpus* lain yang digunakan dapat memiliki *speaker* dengan logat yang berbeda, alat perekaman yang berbeda, atau direkam pada lingkungan yang berbeda.
2. Peneliti selanjutnya dapat menggunakan teknik *noise reduction* pada suara sebelum melakukan pengenalan ucapan otomatis, seperti penggunaan filter *Kalman* dan *Wiener*.
3. Penerapan arsitektur *bidirectional recurrent neural network* lainnya untuk pengenalan ucapan, seperti *Gated Recurrent Unit (GRU)*.
4. Peneliti selanjutnya dapat menerapkan *distributed computing* untuk mempercepat proses *training*.

DAFTAR PUSTAKA

- Castillo, D. (2017). The Fast Fourier Transform Algorithm. Diambil 5 Mei 2018, dari <https://medium.com/@diegocasma/the-fast-fourier-transform-algorithm-6f06900c565b>
- Dahl, G. E., Yu, D., Deng, L., & Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech and Language Processing*, 20(1), 30–42. <https://doi.org/10.1109/TASL.2011.2134090>
- Davis, K. H., Biddulph, R., & Balashek, S. (1952). Automatic Recognition of Spoken Digits. *The Journal of the Acoustical Society of America*, 24(6), 637–642. <https://doi.org/10.1121/1.1906946>
- Davis, S. B., & Mermelstein, P. (1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 357–366. <https://doi.org/10.1109/TASSP.1980.1163420>
- Goldhahn, D., Eckart, T., & Quasthoff, U. (2012). Building Large Monolingual Dictionaries at the Leipzig Corpora Collection: From 100 to 200 Languages. *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, 759–765. <https://doi.org/10.1007/s00127-009-0084-7>
- Graves, A. (2013). *Supervised Sequence Labeling with Recurrent Neural Networks*. *arXiv preprint arXiv:1308.0850* (Vol. 12). <https://doi.org/10.1145/2661829.2661935>
- Graves, A., Fernandez, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks. *Proceedings of the 23rd international conference on Machine Learning*, 369–376. <https://doi.org/10.1145/1143844.1143891>
- Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005*. (Vol. 4, hal. 2047–2052). IEEE. <https://doi.org/10.1109/IJCNN.2005.1556215>
- Gupta, S., Jaafar, J., Ahmad, W. wan, & Bansal, A. (2013). Feature Extraction Using Mfcc. *Signal & Image Processing: An International Journal (SIPIJ)*, 4(4), 101–108. <https://doi.org/10.5121/sipij.2013.4408>
- Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., ... Ng, A. Y. (2014). Deep Speech: Scaling up end-to-end speech recognition, 1–12. <https://doi.org/arXiv:1412.5567v2>
- Hannun, A. Y., Maas, A. L., Jurafsky, D., & Ng, A. Y. (2014). First-Pass Large

- Vocabulary Continuous Speech Recognition using Bi-Directional Recurrent DNNs, 1–7. Diambil dari <http://arxiv.org/abs/1408.2873>
- Haykin, S. (2008). *Neural Networks and Learning Machines*. New Jersey: Pearson Prentice Hall. <https://doi.org/978-0131471399>
- Heafield, K. (2011). KenLM : Faster and Smaller Language Model Queries. *Wmt-2011*, (2009), 187–197. Diambil dari <http://www.aclweb.org/anthology/W11-2123%5Cnhttp://kheafield.com/code/kenlm>
- Heaton, J. (2008). *Programming Neural Networks with Encog3 in Java*. Diambil dari <http://www.heatonresearch.com/book/programming-neural-networks-encog3-java.html>
- Katz, S. M. (1987). Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 35(3), 400–401. <https://doi.org/10.1109/TASSP.1987.1165125>
- Kim, Y.-G., & Huynh, X.-P. (2017). Discrimination Between Genuine Versus Fake Emotion Using Long-Short Term Memory with Parametric Bias and Facial Landmarks. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 3065–3072. <https://doi.org/10.1109/ICCVW.2017.362>
- Ko, T., Peddinti, V., Povey, D., & Khudanpur, S. (2015). Audio augmentation for speech recognition. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH* (Vol. 2015–Janua, hal. 3586–3589). Sixteenth Annual Conference of the International Speech Communication Association. <https://doi.org/10.1016/j.neucom.2011.09.037>
- Krizhevsky, A., Sutskever, I., & Geoffrey E., H. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, 1–9. <https://doi.org/10.1109/5.726791>
- Lestari, D. P., Shinoda, K., & Furui, S. (2011). TITML-IDN Tokyo Institute of Technology Multilingual Speech Corpus - Indonesian -. Tokyo: Tokyo Institute of Technology.
- Miao, Y., Gowayyed, M., & Metze, F. (2016). EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015 - Proceedings*, 167–174. <https://doi.org/10.1109/ASRU.2015.7404790>
- Morais, R. (2017). A Journey to <10% Word Error Rate. Diambil 12 April 2018, dari <https://hacks.mozilla.org/2017/11/a-journey-to-10-word-error-rate/>
- Naveen Kumar, M., Linga Chandar, P. C., Venkatesh Prasad, A., & Sumangali, K. (2017). Android based educational Chatbot for visually impaired people. *2016 IEEE International Conference on Computational Intelligence and Computing Research, ICCIC 2016*, 0–3. <https://doi.org/10.1109/ICCIC.2016.7919664>
- Negnevitsky, M. (2005). *Artificial Intelligence* (3rd Editio). Harlow: Pearson Education Limited.
- O'Shaughnessy, D. (2008). Invited paper: Automatic speech recognition: History,

- methods and challenges. *Pattern Recognition*, 41(10), 2965–2979. <https://doi.org/10.1016/j.patcog.2008.05.008>
- Olah, C. (2015). Understanding LSTM Networks. Diambil 13 Desember 2018, dari <http://colah.github.io/posts/2015-08-Understanding-LSTMs>
- Oshana, R. (2012). Introduction to Digital Signal Processing. *DSP for Embedded and Real-Time Systems*, 1–14. <https://doi.org/10.1016/B978-0-12-386535-9.00001-9>
- Prechelt, L. (2012). Early stopping - But when? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7700 LECTU, 53–67. <https://doi.org/10.1007/978-3-642-35289-8-5>
- Russel, S. J., & Norvig, P. (2010). *Artificial Intelligence a Modern Approach*. <https://doi.org/10.1017/S0269888900007724>
- Sajjan, S. C., & Vijaya, C. (2012). Comparison of DTW and HMM for isolated word recognition. *International Conference on Pattern Recognition, Informatics and Medical Engineering, PRIME 2012*, (1), 466–470. <https://doi.org/10.1109/ICPRIME.2012.6208391>
- Sak, H., Senior, A., Rao, K., & Beaufays, F. (2015). Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition. Diambil dari <http://arxiv.org/abs/1507.06947>
- Serrano, J., Gonzalez, F., & Zalewski, J. (2015). CleverNAO: The intelligent conversational humanoid robot. *Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2015*, 2(September), 887–892. <https://doi.org/10.1109/IDAACS.2015.7341431>
- Servan-Schreiber, D., Cleeremans, A., & McClelland, J. L. (1991). Graded State Machines: The Representation of Temporal Contingencies in Simple Recurrent Networks, 193, 161–193.
- Sigurdson, S., Petersen, K., & Larsen, J. (2006). Mel Frequency Cepstral Coefficients: An Evaluation of Robustness of MP3 Encoded Music. *Victoria*, (m), 3–6. <https://doi.org/10.1.1.83.7049>
- Smith, S. W. (1999). *Digital signal processing*. California Technical Publishing. <https://doi.org/10.1109/79.826412>
- Soares, F., & Souza, A. M. F. (2016). *Neural network programming with Java : unleash the power of neural networks by implementing professional Java code*. Birmingham: Packt Publishing Ltd.
- Song, F., & Croft, W. B. (1999). A general language model for information retrieval. *Proceedings of the eighth international conference on Information and knowledge management - CIKM '99*, 316–321. <https://doi.org/10.1145/319950.320022>
- Yi, J., Wen, Z., Tao, J., Ni, H., & Liu, B. (2017). CTC Regularized Model Adaptation for Improving LSTM RNN Based Multi-Accent Mandarin Speech Recognition. *Journal of Signal Processing Systems*, 1–13. <https://doi.org/10.1007/s11265-017-1291-1>

Yucesoy, E., Nabiyev, V. V., Yücesoy, E., & Nabiyev, V. V. (2013). Gender Identification Of A Speaker Using MFCC And GMM. *International Conference on Electrical and Electronics Engineering (ELECO)*, (April), 626–629. <https://doi.org/10.1109/ELECO.2013.6713922>