BAB I

PENDAHULUAN

1.1 Latar Belakang

Di era globalisasi ini, teknologi informasi dan komunikasi meningkat semakin pesat, sehingga hal ini membawa perubahan di segala aspek kehidupan manusia, ini termasuk cara manusia saling berkomunikasi. Untuk saling berkomunikasi, seorang manusia harus bisa memahami bahasa yang digunakan oleh lawan bicara. Jika lawan bicara kita menggunakan bahasa asing dan kita tidak menguasai bahasa asing tersebut, salah satu contohnya adalah jika lawan bicara kita menggunakan bahasa Jepang. Seperti yang dijelaskan pada penelitian (Istiqomah et al., 2015) bahwa bahasa Jepang sangat berbeda dengan bahasa asing lainnya, bahasa Jepang memiliki struktur bahasa dan gaya bicara yang cukup rumit, sehingga ini cukup menyusahkan bagi individu yang tidak menguasai atau yang sedang mempelajari bahasa Jepang. Untuk itu, individu membutuhkan penerjemah bahasa yang akurat agar dapat memahami konteks yang disampaikan.

Untuk menerjemahkan suatu bahasa, individu biasanya menggunakan kamus untuk menerjemahkannya secara kata per kata. Namun seiring perkembangan teknologi, banyak kamus digital mulai bermunculan. Kamus digital dapat dibuat dengan berbagai bahasa pemrograman tergantung platform yang digunakan, jika untuk *Website* bisa menggunakan bahasa pemrograman *PHP, JavaScript,* dan *Python,* jika untuk *mobile (Android/IOS)* bisa menggunakan bahasa pemrograman *kotlin, java,* dan *swift.* Tidak seperti kamus tradisional yang

masih menerjemahkan kata per kata, kamus digital ini dapat menerjemahkan dalam bentuk kalimat ataupun dalam paragraf. Namun, ada beberapa kamus digital yang masih memiliki kekurangan, kekurangannya yaitu individu harus memasukkan teks terlebih dahulu ke aplikasi kamus digital untuk melakukan penerjemahan dan juga ada kemungkinan individu tersebut tidak bisa mengucapkan hasil terjemahan dalam bahasa asing dengan benar, sehingga bisa terjadi miskomunikasi antar individu.

Salah satu solusi yang bisa diberikan yaitu dengan menggunakan teknologi *Speech to Text* dan *Text To Speech. Speech to Text* (*STT*) merupakan suatu proses konversi ucapan manusia secara otomatis menjadi teks (Lubis et al., 2024). Dengan menggunakan *STT,* pengguna tidak perlu lagi memasukkan teks untuk menerjemahkannya, pengguna hanya perlu berbicara dan secara otomatis sistem akan mengonversi ucapan menjadi teks. Sedangkan *Text To Speech* (*TTS*) merupakan teknologi yang mengizinkan komputer untuk berbicara seperti layaknya pembicara asli atau mendekati pembicara asli berdasarkan teks masukan (Prastyo, 2022). Dengan menggunakan *TTS,* pengguna tidak akan kesusahan saat membaca hasil terjemahan. *Speech to Text* dan *Text To Speech* dapat diimplementasikan menggunakan *Web Speech API* dalam bahasa pemrograman *JavaScript.*

Web Speech API merupakan *Application Programming Interface (API)* yang dikembangkan oleh *Google* dan dapat diakses secara gratis oleh publik. *API* ini memiliki kemampuan untuk mengenali suara pengguna dan juga dapat mengonversi suatu teks menjadi ucapan manusia. *Web Speech API* dapat diimplementasikan menggunakan *Deep Learning* berbasis *Recurrent Neural Network* (*RNN*). *Deep Learning* memiliki kemampuan untuk mempelajari pola

suara yang rumit dengan memberikan model rekaman suara ke sistem. Salah satu jenis dari *Deep Learning* adalah *Recurrent Neural Network (RNN)*. *RNN* memiliki kemampuan untuk mempertahankan informasi dalam "memori" untuk waktu yang lama dan memproses *input* dalam urutan, sehingga cocok untuk menangani data seperti audio.

Selain menggunakan *Speech to Text* dan *Text To Speech*, *Machine Translation* (*MT*) diperlukan untuk melakukan penerjemahan. *Machine Translation* merupakan proses penerjemahan dari satu bahasa ke bahasa lainnya secara otomatis yang dilakukan oleh sistem dengan cara memasukkan *dataset* yang besar untuk melatih model penerjemahan. *Dataset* ini berupa pasangan kalimat dalam dua bahasa yang berbeda dan keduanya sudah diterjemahkan. *Dataset* ini memiliki fungsi untuk melatih model penerjemah agar dapat memahami teks dari satu bahasa ke bahasa lain. Dengan menggunakan *Google Translate API,* pengembang tidak perlu melatih model lagi dari awal, karena *Google Translate API* sudah dilengkapi dengan *Machine Translation* (MT). Pada tahun 2020, Google mengumumkan model generatif baru untuk *API* terjemahan *Google Cloud*, yang berbasis *Transformer* dan telah meningkatkan kualitas secara signifikan (*Google Cloud Translation AI*, 2025). *Google Translate API* dilengkapi dengan *dataset* dalam jumlah yang sangat besar sehingga dapat menerjemahkan ke berbagai bahasa, salah satunya adalah bahasa Indonesia-Jepang. *Transformer* merupakan salah satu bagian dari *Deep Learning.* Dalam konteks *Machine Translation, Transformer* memiliki kemampuan untuk memproses seluruh data seperti kalimat sekaligus, bukan kata per kata. Maksudnya *Transformer* dapat memahami hubungan antar kata dalam suatu kalimat, meskipun kata tersebut saling berjauhan, sehingga menjadikan model

ini lebih cepat dan tepat dalam memahami konteks kalimat secara keseluruhan (AWS, 2025). Hal ini menjadikan *Transformer* sangat bagus digunakan dalam berbagai tugas, salah satunya adalah penerjemahan bahasa.

Sesuai dengan latar belakang di atas, penulis membuat skripsi tentang pembuatan aplikasi berbasis *website* dengan judul "IMPLEMENTASI ALGORITMA *DEEP LEARNING* PADA APLIKASI PENERJEMAH SUARA OTOMATIS INDONESIA-JEPANG".

1.2 Rumusan Masalah

Berdasarkan uraian latar belakang di atas, rumusan masalah dalam penelitian ini adalah:

1. Bagaimana Algoritma *Deep Learning* dengan arsitektur *Recurrent Neural Network* (RNN) dan *Transformer* diimplementasikan pada pengembangan aplikasi penerjemah suara otomatis Indonesia-Jepang?
2. Bagaimana pembuatan aplikasi penerjemah suara otomatis Indonesia-Jepang?

1.3 Batasan Masalah

Berikut adalah batasan dari penelitian ini:

1. Pembuatan Aplikasi Penerjemah Suara Otomatis Indonesia-Jepang *Online* hanya menggunakan bahasa pemrograman JavaScript, serta memanfaatkan arsitektur *Recurrent Neural Network* (RNN) dan *Transformer*.
2. *Speech to Text,* penerjemahan, dan *Text to Speech* mendukung bahasa Indonesia dan bahasa Jepang, dengan memanfaatkan layanan *Web Speech API* untuk *Speech to Text* dan *Text to Speech* serta *Google Translate API*

untuk proses penerjemahan yang dapat dilakukan dalam dua arah, dari bahasa Indonesia ke Bahasa Jepang dan bahasa Jepang ke bahasa Indonesia.

3. Aplikasi Penerjemah Suara Otomatis Indonesia-Jepang *Online* mendukung dua metode *input* suara, yaitu melalui *microphone* dan unggahan file suara berformat mp3 atau wav.

1.4 Tujuan Penelitian

Berikut adalah tujuan yang ingin dicapai oleh penulis dalam penelitian ini adalah:

1. Mengimplementasikan algoritma *Deep Learning* dengan arsitektur *Recurrent Neural Network* (RNN) pada aplikasi penerjemah suara otomatis Indonesia-Jepang *Online*, untuk meningkatkan kinerja dan akurasi pada *Speech to Text* dan *Text To Speech.*
2. Mengimplementasikan salah arsitektur dari algoritma *Deep Learning,* yaitu *Transformer* pada *Machine Translation* guna memberikan hasil terjemahan yang akurat dan sesuai dengan konteks.

1.5 Manfaat Penelitian

Berikut adalah manfaat penelitian ini:

1. Untuk menghasilkan aplikasi yang dapat menerjemahkan secara otomatis dari bahasa Indonesia ke bahasa Jepang dan bahasa Jepang ke Indonesia menggunakan suara.
2. Untuk menghasilkan aplikasi yang dapat mengucapkan hasil terjemahan secara otomatis.

1.6 Metodologi Penelitian

Metode penelitian yang dilakukan pada penelitian ini adalah :

1. Studi Kepustakaan

   Penulis melakukan studi kepustakaan dengan mengumpulkan informasi dari berbagai bahan referensi yang berkaitan dengan penelitian.

2. Analisis dan Perancangan

   Penulis akan menganalisis spesifikasi aplikasi dan melakukan perancangan aplikasi, mulai dari perancangan proses, hingga perancangan *interface* (antarmuka).

3. Pengkodean

   Penulis akan melakukan pengkodean aplikasi berdasarkan analisis spesifikasi dan *interface* yang sudah dirancang sebelumnya.

4. Pengujian Aplikasi

   Pada tahap ini akan dilakukan pengujian terhadap aplikasi yang sudah dikembangkan.

5. Penyusunan Laporan

   Penulis akan melakukan proses dokumentasi dan laporan aplikasi yang sudah dikembangkan.

1.7 Sistematika Penulisan

Sistematika penulisan Skripsi ini dibagi menjadi beberapa bab, dimana masing-masing bab dibagi menjadi beberapa sub agar mempermudah penjelasan mengenai penelitian yang dilakukan dan mempermudah pembaca dalam

memahami isi penelitian. Adapun sistematika dari penulisan skripsi ini adalah sebagai berikut :

BAB 1 PENDAHULUAN

Pendahuluan berisi tentang Latar Belakang Masalah, Rumusan Masalah, Tujuan, Manfaat, Batasan Masalah, Metodologi Penelitian, dan Sistematika Penulisan dalam pembuatan tugas skripsi

BAB 2 TINJAUAN PUSTAKA

Bab ini berisi teori-teori pengetahuan dasar yang diperoleh dari studi kepustakaan atau literatur dan dokumentasi *internet* yang digunakan untuk memahami permasalahan yang dibahas pada penelitian ini. Teori-teori pengetahuan dasar yang disajikan antara lain tentang aplikasi, *Speech to Text, Text To Speech, Machine Translation, Deep Learning.*

BAB 3 METODE PENELITIAN

Bab ini menguraikan tahapan yang digunakan dalam melakukan kajian penelitian. Tahapan tersebut merupakan kerangka yang dijadikan pedoman penelitian untuk mencapai tujuan yang sudah ditetapkan. Tahapan tersebut berisi waktu dan tempat penelitian untuk membuat aplikasi penerjemah suara otomatis Indonesia-Jepang berbasis *website* dengan algoritma *Deep Learning*

BAB 4 HASIL DAN PEMBAHASAN

Bab ini berisi hasil dan pembahasan dari aplikasi penerjemah suara otomatis Indonesia-Jepang berbasis *website* dengan algoritma *Deep Learning.*

BAB 5 KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari keseluruhan bab penulisan skripsi dan saran yang diajukan untuk pengembangan yang lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1 Aplikasi

2.1.1 Pengertian Aplikasi

Aplikasi adalah suatu perangkat lunak yang dikembangkan untuk melakukan tugas dari sistem yang memiliki berbagai kegunaan yang dapat membantu kegiatan manusia. Aplikasi adalah suatu program yang siap digunakan untuk melakukan suatu tugas bagi pengguna aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju (Parjito et al., 2022). Selain pengertian diatas, banyak pengertian dari aplikasi yang dikemukakan oleh para ahli. Berikut adalah beberapa definisi dari aplikasi menurut para ahli (Hawari Nasution et al., 2023):

1. Hengky W. Pramana

   Menurut Hengky W. Pramana, aplikasi merupakan suatu perangkat lunak yang dibuat khusus untuk memenuhi kebutuhan berbagai aktivitas dan pekerjaan.

2. Harip Santoso

   Menurut Harip Santoso, aplikasi adalah suatu kelompok file (*report, class, form*) yang dibuat untuk mengeksekusi kegiatan tertentu yang saling berhubungan.

3. Sri Widianti

Menurut Sri Widianti, aplikasi adalah suatu perangkat lunak yang dibuat sebagai *front end* sebuah sistem yang dipakai untuk mengelola data sehingga menjadi suatu informasi yang bermanfaat bagi pengguna.

4. Rachmad Hakim S.

Menurut Rachmad Hakim S., aplikasi merupakan sebuah *software* yang dibuat untuk tujuan tertentu, contohnya untuk mengelola dokumen, bermain *game*, dan lain sebagainya.

2.1.2 Jenis-Jenis Aplikasi

Aplikasi dapat dikategorikan dalam tiga kelompok, yaitu (Handayani et al., 2022):

1. Aplikasi *Desktop*, yaitu aplikasi yang dijalankan di perangkat PC atau laptop.
2. Aplikasi *Web*, yaitu aplikasi yang dijalankan menggunakan komputer dan membutuhkan koneksi *internet*.
3. Aplikasi *Mobile*, yaitu aplikasi yang dijalankan di perangkat *mobile*.

2.2 *Website*

2.2.1 Pengertian *Website*

Website merupakan kumpulan dari beberapa halaman yang dapat diakses menggunakan jaringan *internet*, halaman-halaman ini berisi informasi digital yang dapat dibaca dan dilihat oleh pengguna. Informasi ini dapat berupa gambar, teks, animasi, suara dan video (Fitriani et al., 2022).

2.2.2 Jenis-Jenis *Website*

Secara umum, *website* dibagi menjadi 2 jenis, yaitu:

1. *Website Statis*

   Website statis merupakan salah satu dari jenis *website* yang memiliki ciri khas yang memiliki isi yang jarang diperbarui secara berkala, kecuali ada perbaruan informasi yang dilakukan oleh pemilik *website.* Jenis *website* ini biasanya dikembangkan menggunakan HTML dan CSS tanpa adanya koneksi dengan *database.* Jenis *website* ini biasanya dipakai untuk mengembangkan profil perusahaan, dan portofolio pribadi. *Website statis* biasanya memiliki struktur *landing page* yang dimana hanya memiliki satu halaman *website* saja.

2. *Website Dinamis*

   Website dinamis merupakan salah satu dari jenis *website* yang memiliki isi yang isinya akan selalu diperbaharui secara berkala oleh pengembang *website.* Jenis *website* ini biasanya tidak hanya menggunakan HTML dan CSS saja dalam pengembangannya, tetapi juga menggunakan bahasa *PHP, JavaScript,* atau *Python* dan juga *website* ini juga biasanya memiliki koneksi dengan *database* seperti *MySql.* Contoh dari jenis *website* ini adalah *website media social, e-commerce,* dan *website* berita.

2.3 Algoritma

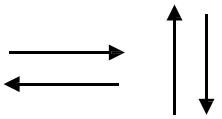
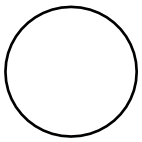
2.3.1 Pengertian Algoritma

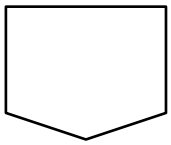
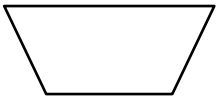
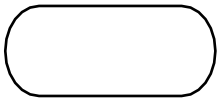
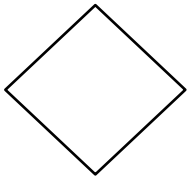
Algoritma merupakan suatu langkah-langkah yang disusun secara sistematis untuk menyelesaikan suatu permasalahan. Sedangkan algoritma pemrograman adalah langkah-langkah yang ditulis secara berurutan guna untuk menyelesaikan masalah yang berkaitan dengan komputer (Khairiah, 2023). Pada setiap langkah yang digambarkan pada flowchart dihubungkan dengan garis atau tanda panah. *Flowchart* sendiri memiliki fungsi sebagai gambar bagaimana suatu program berjalan dari satu proses ke proses lainnya sehingga alur berjalannya suatu program dapat dipahami oleh semua orang.

2.3.2 Simbol *Flowchart*

Setiap simbol pada *flowchart* memiliki arti yang berbeda. Berikut adalah simbol-simbol yang digunakan dalam proses pembuatan *flowchart* (Hosting, 2022).

Tabel 2. 1 Simbol-simbol *flowchart*

No.	Simbol	Nama Simbol	Fungsi
1.		Arus (*Flow Direction*)	Untuk menunjukkan garis alir dari proses.
2.		Penghubung (*On-Page Connector*)	Sebagai penyambungan proses dalam halaman yang sama.

3.		Penghubung (*Off-Page Connector*)	Sebagai penyambungan proses dalam halaman yang berbeda.
4.		Proses (*Processing*)	Untuk menunjukkan proses pengolahan data yang dilakukan oleh komputer.
5.		Operasi Manual (*Manual Operation*)	Untuk menunjukkan proses pengolahan yang tidak dilakukan oleh komputer.
6.		Titik Terminal (*Terminal Point*)	Untuk menunjukkan suatu permulaan (*start*) atau akhir (*end*) dalam suatu alur proses.
7.		Keputusan (*Decision*)	Untuk memilih proses berdasarkan kondisi yang ada.
8		Proses terdefenisi (Predefined Process)	Sebagai kumpulan langkah-langkah.
9.		Persiapan (*Preparation*)	Untuk pelaksanaan suatu bagian (sub-program) atau prosedur.
10.		Keluar-Masuk (*Input-Output*)	Untuk menyatakan proses *input* dan *output* pada alur proses.

11.		Dokumen (*Document*)	Untuk menyatakan *input* yang berasal dari dokumen dalam bentuk kertas atau *output* dicetak ke kertas.

2.4 Bahasa Pemrograman *Web*

2.4.1 HTML (*Hypertext Markup Language*)

HTML berasal dari singkatan dari *Hypertext Markup Language* merupakan suatu bahasa yang menggunakan *tag* tertentu untuk menyatakan kode-kode yang harus ditafsirkan oleh *browser* agar halaman dapat ditampilkan secara benar. HTML memiliki peran penting dalam pemrograman *web* karena HTML merupakan penyusun struktur dari halaman *website* dengan menempatkan setiap *tag* sesuai dengan *layout* yang diinginkan. Untuk membuat skrip HTML, kita dapat menggunakan *text editor* seperti *sublime text,* dan *visual studio code, notepad,* dan *text editor* lainnya. Skrip HTML biasanya disimpan dengan file yang berekstensi .html.

2.4.2 CSS (*Cascading Style Sheet*)

Cascading Style Sheet atau yang disingkat CSS merupakan bahasa yang digunakan untuk mengatur tampilan dan desain dari sebuah halaman *website*. CSS memiliki peran sebagai pengatur desain suatu *website* menjadi tampil lebih menarik dan responsif dari sebelumnya, dengan cara mengatur warna, ukuran *font,* jenis *font, dan* efek visual lainnya dari elemen-elemen HTML. Hal ini dapat meningkatkan

*User Experience (*pengalaman pengguna) saat berinteraksi dengan halaman *website* tersebut (Silverwood Summit, 2023).

Cara kerja dari CSS ini cukup sederhana, yaitu dengan memilih *class/id* atau *tag* dari file HTML yang sudah dibuat, kemudian berikan property yang sesuai dengan tampilan yang diinginkan. Sama hal nya dengan HTML, CSS harus dibuat di *text editor*, seperti *sublime text, visual studio code,* dan *text editor* lainnya. Kemudian disimpan dengan ekstensi .css. Untuk mengintegrasikan file CSS dengan file HTML, cukup dengan mengetik perintah "<link rel="stylesheet" href="style.css"> didalam *tag* <head></head> pada file HTML. CSS dan HTML juga bisa dibuat dalam satu file dengan cara mengetik tag <style></style> pada file HTML, kemudian diisi dengan class/id atau *tag* yang akan diatur tampilannya. Jika hanya ingin mengatur sedikit saja tampilan dari *website,* kita juga bisa menggunakan CSS langsung pada *tag* HTML, contohnya <h1 style="color: red; font-size: 12px;">Aku kaya</head>.

2.4.3 JavaScript

JavaScript merupakan salah satu bahasa pemrograman yang cukup popular saat ini. JavaScript biasanya digunakan untuk membangun suatu *website,* namun seiring perkembangan jaman, JavaScript bisa diimplementasikan dalam pengembangan aplikasi *mobile,* aplikasi *dekstop, backend (server), game development, Internet of Things (IOT),* dan *Machine Learning*, tapi untuk melakukan ini semua diperlukannya integrasi dengan *framework* dan *library.*

Sama halnya dengan HTML dan CSS, JavaScript memerlukan *text editor* dalam pembuatannya, misalnya *Visual Studio Code.* JavaScript dapat ditulis dengan

dua cara, yang pertama JavaScript dapat ditulis langsung dalam HTML, dan yang kedua dapat ditulis terpisah dengan file HTML. JavaScript yang ditulis terpisah dari HTML akan disimpan dengan ekstensi .js, dan untuk mengintegrasikan file .js tersebut dengan file .html, diperlukan mengetik *tag* <script src=”main.js”></script>. Jika ingin menulis JavaScript secara langsung didalam file .html, diperlukan mengetik *tag* <*script*> kemudian dilanjutkan dengan *function* dan memilih salah satu *function* yang ingin diatur interaksinya (Mallisa, 2021). Contohnya adalah sebagai berikut :

```
<body>
<button onclick=”tampilPesan()”>Ayo klik aku klik aku</button>
<script>
function tampilPesan(){
alert(“Aku kaya”);
}
</script>
```

Skrip ini akan memiliki *output* “aku kaya” jika pengguna menekan tombol “ayo klik aku klik aku”. Jika HTML berperan sebagai struktur halaman *website*, CSS sebagai pembuat tampilan halaman *website,* JavaScript memiliki peran sebagai meningkatkan interaksi dan perilaku pada halaman *website* seperti pada contoh diatas. JavaScript tidak hanya untuk menampilkan teks seperti pada contoh diatas, JavaScript bisa melakukan banyak hal, seperti mengambil data dari server, membuat animasi, dan membuat halaman *website* menjadi lebih hidup dan lebih

responsif. Sehingga JavaScript memiliki peran penting dalam pengembangan *website* karena dapat membuat *website* menjadi lebih modern, interaktif, dan tidak menjadikannya *website statis.*

2.5 *Speech to Text*

2.5.1 Pengertian *Speech to Text*

Speech to Text merupakan perangkat lunak yang dapat mengenali dan mengonversi ucapan manusia secara otomatis menjadi format teks. *Speech to Text* ini bekerja dengan mengenali gelombang suara, menganalisis tata bahasa, kemudian mengubahnya menjadi teks sesuai dengan apa yang disampaikan oleh pengguna. Sebelum adanya *Speech to Text,* manusia biasanya perlu memasukkan teks terlebih dahulu untuk melakukan sesuatu seperti ingin melakukan penerjemahan ke bahasa asing. Namun dengan menggunakan *Speech to Text,* manusia hanya perlu berbicara ke kamus digital, dan kamus digital tersebut dapat mendeteksi dan mengenali ucapan pengguna, kemudian akan menerjemahkannya secara otomatis ke bahasa asing. Selain untuk aplikasi penerjemah, *Speech to Text* semakin banyak diimplementasikan di berbagai hal, contohnya pada alat bantu komunikasi bagi tuna rungu (tuli).

2.5.2 Cara Kerja *Speech to Text*

Speech to Text bekerja dengan menggunakan *Machine Learning (ML)* dan *Artificial Intelegent (AI). Machine Learning (ML)* merupakan teknologi yang berfungsi untuk melatih komputer dengan memberikan *dataset* dalam jumlah yang besar. Dalam konteks pengenalan ucapan, *dataset* ini berupa data rekaman suara dan diberikan kepada *Machine Learning* untuk melatih komputer. Dengan melatih

komputer menggunakan *dataset* dalam jumlah yang besar, komputer mampu untuk mengonversi suara menjadi teks dengan akurat, karena dapat mencocokkan pola suara yang direkman dengan pola suara dari data yang sudah disimpan sebelumnya (AWS, 2025). Berikut adalah alur kerja dari *Speech to Text:*

Gambar 2. 1 Alur kerja *Speech to Text*

2.6 *Text To Speech*

2.6.1 Pengertian *Text To Speech*

Text To Speech merupakan sistem yang memiliki kegunaan untuk mengubah teks tertulis menjadi ucapan yang mirip dengan ucapan manusia. Hanya dengan memasukkan teks dalam bentuk kata, kalimat ataupun paragraf, sistem akan memberikan output berupa suara yang mirip dengan ucapan manusia yang sesuai dengan teks yang sudah dimasukkan sebelumnya. Dengan adanya teknologi ini, orang-orang yang memiliki keterbatasaan dalam penglihatan (tuna netra) atau orang yang kesulitan membaca (disleksia) akan terbantu saat melakukan komunikasi dan saat mengakses informasi (Handayani et al., 2022). Dengan mengimplementasikan *Text To Speech* ke kamus digital, pengguna tidak perlu membaca hasil terjemahan yang sulit dibaca, sehingga dapat meminimalisir miskomunikasi karena kesalahan saat pembacaan hasil terjemahan.

2.6.2 Cara Kerja *Text To Speech*

Secara umum, proses kerja *Text To Speech* terdiri dari dua tahap utama, yaitu analisis teks (*Text Analysis*) dan sintesis suara (*Speech Synthesis*). Pada tahap analisis teks, sistem akan membaca dan memahami teks yang dimasukkan pengguna, kemudian teks tersebut akan dianalisis secara linguistik untuk menentukan struktur kalimat, pengucapan kata, dan pemilihan intonasi yang tepat. Sistem *Text To Speech* akan mengubah teks yang ada menjadi fonem, fonem merupakan satuan bunyi terkecil dalam bahasa yang mewakili pengucapan sebuah kata. Fonem berperan penting sebagai dasar bagi komputer untuk menghasilkan suara yang menyerupai ucapan manusia (admin, 2021).

Setelah tahap analisis teks selesai, selanjutnya adalah sintesis suara. Sistem akan menyatukan fonem-fonem dari teks yang sudah dimasukkan pengguna, menjadi gelombang suara yang menyerupai ucapan manusia. Berikut adalah alur kerja dari *Text to Speech:*

Gambar 2. 2 Alur kerja *Text to Speech*

2.7 *Machine Translation*

2.7.1 Pengertian *Machine Translation*

Machine Translation merupakan proses penerjemahan dari satu bahasa ke bahasa lainnya secara otomatis yang dilakukan oleh sistem dengan cara memasukkan *dataset* yang besar untuk melatih model penerjemahan. *Dataset* ini berupa pasangan kalimat dalam dua bahasa yang berbeda dan keduanya sudah diterjemahkan. *Dataset* ini memiliki fungsi untuk melatih model penerjemah agar dapat memahami teks dari satu bahasa ke bahasa lain. Dengan menggunakan *Machine Translation* (*MT*), pengguna tidak perlu memasukkan kata satu per satu seperti kamus tradisional, pengguna bisa menerjemahkan dalam bentuk kalimat ataupun paragraf secara langsung. *Machine Translation* akan memeriksa hubungan antar kata dalam sebuah kalimat, sehingga konteks dan makna dari teks asli tetap terjaga dalam bahasa sasaran (AWS, 2025).

2.7.2 Cara Kerja *Machine Translation*

Sebelum menggunakan *Machine Translation,* pengembang harus melatih modelnya terlebih dahulu menggunakan *dataset* dalam jumlah yang sangat besar. *Dataset* ini berupa bahasa sumber yang sudah diterjemahkan ke bahasa asing yang sudah disusun secara berpasangan. *Dataset* ini akan menjadi bekal bagi *Machine Translation* untuk memahami pola dan hubungan antar kata pada teks, sehingga *Machine Translation* dapat memahami kata-kata yang harus dipilih dan menyusun struktur bahasa secara lebih optimal (Habash, 2023). Berikut adalah alur kerja dari *Machine Translation:*

Gambar 2. 3 Alur kerja *Machine Translation*

2.8 *Speech Recognition*

Speech Recognition merupakan suatu teknologi yang diterapkan pada *Speech to Text.* Teknologi ini memberikan kemampuan pada *Speech to Text* untuk mendeteksi masukan dari ucapan manusia. Teknologi ini bekerja dengan memahami kata yang diucapkan oleh pengguna dengan mengubah suara tersebut menjadi format digital dan mencocokkan format digital tersebut dengan pola tertentu yang sudah disimpan sebelumnya. Format digital ini akan dikonversi menjadi gelombang suara dan akan dikonversi lagi menjadi sekumpulan data

numerik, dan selanjutnya akan diterjemahkan dengan kode-kode tertentu untuk mengidentifikasi kata-kata tersebut (Jaman & Fergina, 2021).

2.9 *Speech Synthesis*

Speech Synthesis merupakan proses konversi teks tertulis menjadi ucapan yang dapat didengar. Teknologi ini menggunakan algoritma dan model komputer untuk menghasilkan suara yang menyerupai cara manusia berbicara. Dalam prosesnya, *Speech Synthesis* melibatkan analisis fonetik dan linguistik, kemudian mengkonversikannya menjadi gelombang suara yang dapat dipahami pengguna.

Pada dasarnya, *Speech Synthesis* bertujuan untuk memberikan akses komunikasi yang lebih mudah bagi beberapa kelompok, seperti individu yang memiliki keterbatasan penglihatan atau kesulitan membaca. *Speech Synthesis* juga diterapkan diberbagai bidang, seperti asisten virtual, navigasi GPS, edukasi, dan layanan pelanggan (UMA, 2024).

2.10 *Web Speech API (Web Speech Application Programming Interface)*

Web Speech API merupakan *API* yang memungkinkan *developer* untuk mengintegrasikan fitur pengenalan suara dan sintesis suara ke dalam aplikasi *web. Web Speech API* memiliki dua bagian, yaitu (Swapurba & Pratama, 2023):

1. *Speech Recognition,* merupakan *interface* dari *Web Speech API* yang mengatur proses pengenalan suara (*Speech to Text*). *API* ini akan menerima masukkan berupa ucapan manusia, kemudian mengubahnya menjadi teks.
2. *Speech Synthesis,* merupakan salah satu *interface* yang tersedia dalam *Web Speech API* yang bertugas untuk mengatur proses penerjemahan teks menjadi suara (*Text To Speech*).

2.11 *Google Translate API (Google Translate Application Programming Interface)*

Google Translate API merupakan *API* dikembangkan oleh *Google. API* ini memiliki kemampuan penerjemahan secara otomatis yang mendukung lebih dari 100 bahasa. Layanan ini didukung oleh *Neural Machine Translation* (*NMT*) yang menggunakan model *Transformer*. Dengan memanfaatkan *Google Translate API,* pengguna tidak perlu menerjemahkan kata satu per satu, tetapi bisa langsung menerjemahkan suatu teks dalam bentuk kalimat ataupun paragraf, sehingga hasil terjemahan akan lebih alami dan sesuai dengan konteks yang ingin disampaikan. Google Translate adalah salah satu *translate* bahasa *online* yang paling terkenal dan paling banyak digunakan di seluruh dunia saat ini (Saientisna et al.,2025).

2.12 Algoritma *Deep Learning*

Deep Learning merupakan subbidang dari *Machine Learning* yang terinspirasi dari otak manusia. Struktur ini disebut *Artificial Neural Network (ANN)*. Pada dasarnya, *Deep Learning* merupakan metode pembelajaran menggunakan jaringan saraf tiruan (*Artificial Neural Network*) yang berlapis-lapis (*multi layer*). Jaringan saraf tiruan ini dibuat dengan meniru otak manusia, yang dimana terdapat neuron-neuron yang saling terkoneksi satu sama lain sehingga membentuk sebuah jaringan yang kompleks untuk mengolah informasi. Neuron-neuron tersebut memiliki tugas untuk menerima masukan, mengolah, dan mengirim hasil keluaran ke neuron lain yang berada di layer selanjutnya, sehingga terjadi proses pengolahan data bertingkat. (Nugroho et al., 2020).

Dengan banyaknya lapisan neuron, maka sistem yang menggunakan *Deep Learning* mampu untuk mengekstraksi fitur-fitur dari data yang kompleks, mulai dari pola yang sederhana hingga ke pola yang kompleks. Hal ini menjadikan *Deep Learning* sukses dalam memahami pola yang kompleks di berbagai bidang , seperti cara menyusun gambar, teks, pengenalan suara, pemrosesan bahasa alami (Nugroho et al., 2020). Berikut adalah prinsip kerja dari algoritma *Deep Learning* (Lubis et al., 2024):

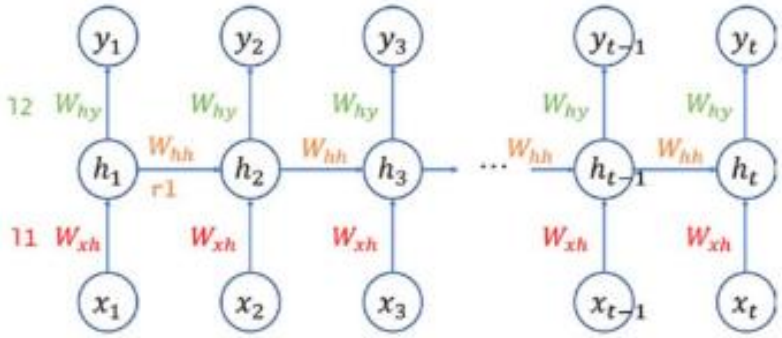
1. Buat arsitektur yang terdiri dari *layer input, hidden,* dan *output*. Gunakan jenis metode *Deep Learning* yang sesuai dan berdasarkan kebutuhan dari kompleksitas kasus yang akan diselesaikan.
2. Siapkan data yang dibutuhkan saat *training* dan *testing Deep Leaning*
3. Pertimbangkan untuk tempat implementasinya (misal di *local* atau menggunakan *cloud*) dan kode programnya (membuat kode program *from scratch* menggabungkan dengan membuat sebagian secara scratch dan menggunakan sebagian kode dari *library*, atau menggunakan *full library*).
4. Lakukan pengujian yang sesuai dengan strandar dari algoritma *Deep Learning*, mulai dari parameter, sampai ke pengujian bentuk arsitekturnya. Misal pengujiannya secara *waterfall* (sekuensial tanpa *loop*) atau secara *recycle* (sekuensial dengan *loop* tertentu). Di mana pengujian *recycle* ini mengkondisikan pengujiannya diulang-ulang sampai beberapa kali siklus (perulangan), misal terdapat 3 macam pengujian, pertama menguji parameter A, lalu hasil nilai dari parameter A yang terbaik akan digunakan untuk pengujian parameter B, lalu hasil pegujian nilai dari parameter A dan B yang terbaik akan digunakan untuk pengujian parameter C, lalu hasil

pengujian nilai dari parameter B dan C yang terbaik akan digunakan untuk pengujian parameter A (*loop* ke-1 dan seterusnya sampai iterasi tertentu), dan sebaiknya iterasi tersebut dihentikan ketika nilai paramter A, B dan C sudah stabil serta nilai akurasinya sudah stabil pada nilai yang paling tinggi.

5. Pikirkan bagaimana mengoptimasi lagi algoritma *Deep Learning* dari hasil yang didapatkan saat ini.

2.12.1 *Recurrent Neural Network (RNN)*

Recurrent Neural Network (*RNN*) atau yang disebut juga jaringan saraf berulang merupakan salah satu metode dari *Deep Learning* yang cocok untuk mengenali pola dalam urutan data, seperti teks, video, ucapan, bahasa, genom, dan data deret waktu. Semua hal ini dapat dilakukan oleh *Recurrent Neural Network* (*RNN*) karena *RNN* memiliki kemampuan untuk menyimpan informasi dari aliran data sebelumnya dengan melakukan perulangan di dalam arstekturnya, sehingga *Recurrent Neural Network* (*RNN*) mampu untuk mengingat informasi dari masa lalu yang sudah tersimpan (Irawan, 2024).



https://penerbit.stekom.ac.id/index.php/yayasanpat/article/view/271

Gambar 2. 4 Arsitektur *RNN*

Keterangan :

- h_t adalah nilai neuron pada waktu t
- x_t adalah nilai *input* pada waktu t
- y_t adalah nilai *output* yang dihasillkan dari h_t
- W_{xh} adalah parameter bobot yang menghubungkan antara neuron hidden sebelumnya
- W_{hy} adalah parameter bobot yang menghubungkan antara *input* x_t dengan *output* y_t
- *b* adalah bias dari neuron hidden
- *tanh* adalah fungsi aktivasi
- *l1,r1,l2* adalah nama lain untuk parameter bobot, misalnya $l1(x_t) = W_{xh}h_t$

Persamaan *input* Neuron Hidden pada gambar 2.2 adalah (Dr. Budi Raharjo, S.Kom., M.Kom., MM., 2022):

$$h_t = tanh\ (l1(x_t) + r1(h_{t-1}) \quad \text{.....(1)}$$

Keterangan :

- h_t merupakan neuron tersembunyi pada waktu ke – *t,* yang menyimpan informasi dar *input* saat ini dan sebelumnya.
- x_t merupakan *input* pada waktu ke-*t,* misalnya kata ke-3 dalam kalimat.
- h_{t-1} adalah nilai *hidden state* dari waktu sebelumnya (t-1) yang berisi informasi masa lalu.
- $l1(x_t)$ merupakan transformasi linier dari *input* x_t.
- $r1(h_{t-1})$ merupakan transformasi linier dari *hidden state* sebelumnya.

- *tanh* digunakan sebagai pengubah hasil penjumlahan menjadi nilai non-linear antar -1 sampai 1.

Persamaan *output* dari gambar 2.1 ialah (Dr. Budi Raharjo, S.Kom., M.Kom., MM., n.d.) :

$$y_t = l2(h_t) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

Keterangan

- y_t adalah *output* dari *RNN* pada waktu le-*t,* misalnya prediksi label atau kata berikutnya.
- h_t adalah *hidden state* saat ini, hasil dari persamaan (1).
- $l2(h_t)$ adalah transformasi linier dari h_t.

2.12.2 *Transformer*

Transformer merupakan salah satu bagian dari *Deep Learning* yang dirancang untuk memproses data secara berurutan dan bahasa alami. Ciri utama dari arsitektur Transformer adalah penggunaan mekanisme "*self-attention*" yang memungkinkan model untuk secara efisien memahami keterkaitan dan pentingnya kata-kata dalam teks input. *Self-attention* memungkinkan model untuk memberikan bobot yang berbeda pada setiap kata dalam kalimat, sehingga kata-kata yang lebih relevan mendapatkan perhatian yang lebih tinggi (Khoiriyah, 2023).

Dalam konteks *Machine Translation, Transformer* memiliki kemampuan untuk memproses seluruh data seperti kalimat sekaligus, bukan kata per kata. Maksudnya *Transformer* dapat memahami hubungan antar kata dalam suatu kalimat, meskipun kata tersebut saling berjauhan, sehingga menjadikan model ini lebih cepat dan tepat dalam memahami konteks kalimat secara keseluruhan. Hal ini

menjadikan *Transformer* sangat bagus digunakan dalam berbagai tugas, salah satunya adalah penerjemahan bahasa.

Transformer memiliki dua struktur utama yaitu *encoder* dan *decoder. Encoder* memiliki tugas untuk mengolah teks input menjadi representasi vektor yang berisi informasi tentang teks tersebut. Sedangkan *decoder* memiliki tugas untuk menghasilkan keluaran berdasarkan vektor yang dibuat oleh *encoder.*

2.13 Penelitian Terdahulu

Berikut peneliti mencantumkan hasil-hasil penelitian terdahulu sebagai berikut:

1. Hasil Penelitian Nadhira Lubis (2024)

   Penelitian Nadhira Lubis (2024), yang berjudul "*Implementasi Algoritma Deep Learning pada Aplikasi Speech to Text Online"*. Penelitian ini merupakan penelitian yang menggunakan metode *Deep Learning*, khususnya algoritma *Recurrent Neural Network (RNN),* dalam pengembangan sistem *Speech to Text* berbasis *web*. Penelitian ini bertujuan untuk membangun suatu aplikasi berbasis *web* yang berkemampuan untuk mengonversi suara manusia menjadi teks secara otomatis dengan akurasi yang tinggi. Aplikasi ini dikembangkan menggunakan *Recurrent Neural Network (RNN)* untuk menangkap pola sekuensial pada data audio. Sistem ini dimulai dengan pengambilan suara dari *microphone*, kemudian suara akan diekstraksi menggunakan metode *Mel-Spectogram* atau MFCC, dan proses model *RNN* akan berjalan untuk menghasilkan output teks. Hasil dari penelitian ini menunjukkan bahwa model *RNN* dapat mengenali ucapan

sederhana dengan cukup akurat. Penelitian ini masih terbatas pada proses pengenalan suara saja dan belum mencakup tahap penerjemahan dan mengonversi teks menjadi suara.

2. Hasil Penelitian Raihan Fajar Fadhilah dan Rima Tamara Aldisa (2023)

Penelitian yang dilakukan oleh Raihan Fajar Fadhilah dan Rima Tamara Aldisa (2024), berjudul "Penggunaan Metode Speech Processing untuk Optimasi Performa Aplikasi NLP dalam Penerjemah Bahasa Minang dengan Indonesia Berbasis *Android*". Penelitian ini bertujuan untuk mengembangkan aplikasi penerjemah yang dapat menerjemahkan kalimat berbahasa Minang ke dalam bahasa Indonesia secara otomatis, dengan dukungan metode *Speech Processing* dan teknik *Natural Language Processing* (NLP). Aplikasi ini dibangun berbasis Android, dan ditujukan untuk membantu pelestarian bahasa Minang yang kini mulai jarang digunakan oleh generasi muda akibat pengaruh globalisasi dan rendahnya literasi bahasa daerah.

Metode yang digunakan mencakup proses pengenalan suara (speech input), lalu diteruskan dengan pemrosesan kalimat dalam bentuk teks melalui beberapa tahapan NLP seperti *case folding, tokenizing, stemming,* dan *cosine similarity* untuk mencocokkan kata-kata Minang dengan padanan bahasa Indonesianya. Hasil terjemahan disusun ulang menjadi kalimat yang utuh. Aplikasi ini memiliki tampilan antarmuka berupa *home screen*, pilihan input suara atau teks, pengaturan tukar bahasa, dan fitur hasil terjemahan.

Dalam pengujian, aplikasi ini dinilai cukup efektif dalam menerjemahkan kalimat-kalimat sederhana dalam bahasa Minang. Namun, masih terdapat kelemahan saat menangani frasa kompleks, dialek lokal yang khas, serta istilah budaya yang tidak memiliki padanan langsung dalam bahasa Indonesia. Evaluasi hasil dilakukan dengan membandingkan terjemahan aplikasi dengan hasil dari penerjemah manusia. Fokus utama dari penelitian ini adalah pada penerapan teknologi NLP dan peran penting aplikasi dalam membantu proses pelestarian bahasa lokal melalui solusi teknologi berbasis *mobile*.

3. Hasil Penelitian Mulyono Eko Prastyo (2022)

Penelitian Mulyono Eko Prastyo (2022), berjudul "*Aplikasi Text to Speech Berbasis Javascript.* Penelitian ini membahas perancangan dan pengembangan sebuah aplikasi *Text to Speech* berbasis JavaScript yang memanfaatkan fitur *Web Speech API* yang disediakan oleh browser modern seperti Google Chrome. Tujuan dari penelitian ini adalah menyediakan sebuah aplikasi berbasis *web* yang mampu mengonversi teks menjadi suara (TTS) tanpa memerlukan instalasi perangkat lunak tambahan. Pengembangan aplikasi dilakukan menggunakan pendekatan *Software Development Life Cycle* (SDLC) model *waterfall*, yang mencakup tahap analisis kebutuhan, desain sistem, implementasi, pengujian, dan pemeliharaan. Dalam implementasinya, aplikasi ini menggunakan API *speechSynthesis* dan *speechRecognition* untuk menangani proses pengubahan teks menjadi suara dan juga untuk mengenali suara pengguna menjadi teks secara langsung.

Aplikasi yang dikembangkan tidak menggunakan metode *Deep Learning*, melainkan memanfaatkan layanan bawaan yang telah disediakan browser melalui JavaScript, sehingga lebih ringan dan cepat diakses tanpa koneksi *internet* karena dilengkapi *service worker* yang memungkinkan penggunaan secara offline. Pengujian dilakukan menggunakan metode *blackbox testing* untuk menguji fungsionalitas, serta evaluasi pengalaman pengguna menggunakan *System Usability Scale* (SUS). Hasil pengujian menunjukkan nilai SUS sebesar 73, yang masuk dalam kategori baik hingga sangat baik (*good to excellent*), menandakan bahwa aplikasi ini memiliki tingkat kemudahan penggunaan yang tinggi dan dapat diterima oleh pengguna. Penelitian ini menunjukkan bahwa teknologi berbasis *web* yang ringan dapat dimanfaatkan secara optimal untuk mengembangkan aplikasi TTS yang efektif, efisien, dan mendukung pembelajaran serta kebutuhan komunikasi, terutama bagi pengguna dengan kebutuhan khusus.

2.14 Perbandingan Penelitian Terdahulu Dengan Sekarang

Berikut merupakan perbandingan antara penelitian terdahulu dengan penelitian yang dilakukan oleh penulis:

Tabel 2. 2 Tabel perbandingan terdahulu

Objek Pembahasan	Kelebihan	Kekurangan	Keterangan
Nadhira Lubis (2024), Implementasi Agoritma *Deep Learning* pada Aplikasi *Speech to Text Online*	Menggunakan algoritma *Deep Learning* dengan metode *Recurrent Neural Network* (RNN). Memanfaatkan *Web Speech API,* mendukung *input* suara dari *microphone* dan file rekaman secara *online.*	Tidak memiliki fitur penerjemah dan konversi teks ke suara	Fokus pada *Speech to Text,* bagian dari sistem dari penelitian ini.
Raihan Fajar Fadhilah dan Rima Tamara Aldisa (2024), Penggunaan Metode *Speech Processing* untuk Optimasi Performa Aplikasi *NLP* dalam	Menggabungkan *Speech Processing* dengan *Natural Language Processing* (NLP) untuk menerjemahkan bahasa Minang ke Indonesia.	Tidak memiliki fitur *Text to Speech* dan hanya tersedia di *Android*	Berfokus pada metode pengolahan suara dan NLP, dan fokus bahasa yang berbeda

Penerjemah Bahasa Minang dengan Indonesia Berbasis *Android*			
Mulyono Eko Prastyo (2022), Aplikasi *Text to Speech* berbasis *Javascript*	Menggunakan *Web Speech API* dan *Service Worker API* agar dapat diakses secara *offline.* Mendukung pengaturan suara (kecepatan, dan nada).	Tidak mendukung *Speech to Text* dan penerjemahan.	Hanya berfokus pada *Text to Speech.*
Penelitian ini, Implementasi Algoritma *Deep Learning* pada Aplikasi Penerjemah Suara Otomatis Indonesia-Jepang *Online*	Mengintegrasikan *Speech to Text, Machine Translation,* dan *Text to Speech* dalam satu aplikasi. Menggunan Algoritma *Deep Leaning* dengan metode *Recurrent Nueral Network* dan *Transformer,* memanfaatkan *Web Speech API,* serta mendukung *input* dari	Hanya dapat diakses dengan koneksi *internet.* Penerjemahan hanya tersedia dalam bahasa Indonesia dan Jepang. Tidak adanya pengaturan suara (kecepatan dan nada) pada *Text to Speech*	Penerjemah suara menggunakan *Speech to Text*, Machine Translation, dan *Text to Speech.*

	microphone dan file rekaman.		