UNIVERSITAS INDONESIA

Rancang Bangun Sistem Automatic Speech Recognition untuk Bahasa Indonesia Berbasis Wav2Vec 2.0

SKRIPSI

Mohammad Salman Alfarisi

1806200381

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JUNI 2022

UNIVERSITAS INDONESIA

Rancang Bangun Sistem Automatic Speech Recognition untuk Bahasa Indonesia Berbasis Wav2Vec 2.0

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

Mohammad Salman Alfarisi

1806200381

FAKULTAS TEKNIK

PROGRAM STUDI TEKNIK KOMPUTER

DEPOK

JUNI 2022

HALAMAN PERNYATAAN ORISINALITAS

Skirpsi ini adalah hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun dirujuk
telah saya nyatakan dengan benar.

Nama **: Mohammad Salman Alfarisi**
NPM **: 180628831**

Tanda Tangan **: _____**
Tanggal **: 03 Juni 2022**

LEMBAR PENGESAHAN

Skripsi ini diajukan oleh:
Nama : Mohammad Salman Alfarisi
NPM : 1806200381
Program Studi : Teknik Komputer
Judul Skripsi : RANCANG BANGUN SISTEM AUTOMATIC SPEECH RECOGNITION UNTUK BAHASA INDONESIA BERBASIS WAV2VEC 2.0

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana mata kuliah Skripsi pada Program Studi Teknik Komputer Fakultas Teknik Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Prima Dewi Purnamasari, M.Sc., M.T. (……)

Penguji 1 : Dr. Eng. Mia Rizkinia S.T., M.T. (……)

Penguji 2 : Yan Maraden, S.T., M.T., M.Sc. (……)

Ditetapkan di : Depok
Tanggal : 13 Juli 2022

KATA PENGANTAR

Puji dan syukur dipanjatkan kepada Allah SWT atas berkat, rahmat, dan nikmat-Nya, sehingga dapat diselesaikan laporan Skripsi dengan judul "Rancang Bangun Sistem *Automatic Speech Recognition* untuk Bahasa Indonesia Berbasis Wav2Vec 2.0". Laporan Skripsi disusun untuk memenuhi salah satu syarat kelulusan Fakultas Teknik Universitas Indonesia, khususnya bagi Program Studi S1 Teknik Komputer dalam Departemen Teknik Elektro.

Dalam proses pembuatan Laporan Skripsi, tentunya tidak dapat terselesaikan tanpa adanya bantuan dan dukungan dari pihak lain. Pada kesempatan ini, terima kasih dan apresiasi sebesar-besarnya diucapkan kepada pihak-pihak terkait, yang terdiri dari namun tidak terbatas pada:

1. Dr. Prima Dewi Purnamasari S.T., M.Sc selaku dosen pembimbing yang selalu membantu mengarahkan dan memberikan bantuan, masukan, kritik, serta saran dalam pelaksanaan Skripsi.
2. Orang Tua, kakak, adik, dan keluarga saya lainnya yang selalu mendukung dan memberikan doa selama pelaksanaan dan penulisan Laporan Skripsi.
3. Arief Saferman dan Qisas Tazkia Hasanuddin selaku rekan kerja yang selalu sedia untuk diajak berdiskusi, bertukar wawasan, dan memberikan dukungan serta bantuan kapan pun waktunya.
4. Teman-teman Program Studi S1 Teknik Komputer Universitas Indonesia dan Departemen Teknik Elektro Universitas Indonesia angkatan 2018 lainnya yang juga memberikan dukungan selama dilaksanakannya kegiatan Skripsi.

Adapun segala kekurangan yang terdapat selama pelaksanaan Skripsi harap disampaikan dalam bentuk kritik dan saran konstruktif, karena penulis sadar mengenai kelalaiannya dalam menjalankan proses Skripsi serta pembuatan laporan yang tentunya tidak sempurna. Selain itu, penulis juga ingin memohon maaf sebesar-besarnya kepada segala pihak yang masih merasakan kekurangan baik dari penulis, pelaksanaan Skripsi, ataupun laporan yang dibuatnya. Harapan penulis

untuk ke depannya adalah agar penulis mendapatkan kelancaran dalam menjalankan kehidupan pasca-kampus, serta agar penulis dapat menjadi insan yang berguna bagi agama, bangsa, dan keluarga

Depok, Juni 2022

Mohammad Salman Alfarisi

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertandatangan di bawah ini:

Nama	: Mohammad Salman Alfarisi
NPM	: 1806200381
Program Studi	: Teknik Komputer
Fakultas	: Fakultas Teknik
Jenis Karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universtas Indonesia **Hak Bebas Royalti Noneksklusif** (**Non-exclusive Royalty-Free Right**) atas karya ilmiah saya yang berjudul:

RANCANG BANGUN SISTEM AUTOMATIC SPEECH RECOGNITION UNTUK BAHASA INDONESIA BERBASIS WAV2VEC 2.0

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok
Pada tanggal : 03 Juni 2022
Yang menyatakan

Mohammad Salman Alfarisi

ABSTRAK

Nama : Mohammad Salman Alfarisi
Program Studi : Teknik Komputer
Judul : Rancang Bangun Sistem Automatic Speech Recognition untuk Bahasa Indonesia Berbasis Wav2Vec 2.0

Salah satu permasalahan yang terdapat pada sistem *Automatic Speech Recognition* (ASR) yang sudah ada adalah kurangnya transparansi dalam penanganan data suara, yang tentunya membuat adanya keraguan terhadap privasi data tersebut. Di sisi lainnya, untuk mengembangkan sebuah sistem ASR yang memiliki akurasi memadai dan dapat bekerja secara luring membutuhkan jumlah data yang banyak, khususnya data suara yang sudah diiringi dengan transkripnya. Hal ini menjadi salah satu hambatan utama pengembangan sistem pengenalan suara, terutama pada yang memiliki sumber daya minim seperti Bahasa Indonesia. Oleh karena itu, dalam penelitian ini dilakukan perancangan sistem pengenalan suara otomatis berbasis model wav2vec 2.0, sebuah model kecerdasan buatan yang dapat mengenal sinyal suara dan mengubahnya menjadi teks dengan akurasi yang baik, meskipun hanya dilatih data dengan label yang berjumlah sedikit. Dari pengujian yang dilakukan dengan *dataset* Common Voice 8.0, model wav2vec 2.0 menghasilkan WER sebesar 25,96%, dua kali lebih baik dibandingkan dengan model Bidirectional LSTM biasa yang menghasilkan 50% namun membutuhkan jumlah data dengan label 5 kali lipat lebih banyak dalam proses pelatihan. Namun, model wav2vec membutuhkan sumber daya komputasi menggunakan 2 kali lebih banyak RAM dan 10 kali lebih banyak memori dibandingkan model LSTM.

Kata Kunci: *deep learning, self-supervised learning, wav2vec 2.0, transformer networks, speech recognition*

ABSTRACT

Name : Mohammad Salman Alfarisi
Study Program : Computer Engineering
Title : *Development of Automatic Speech Recognition System for Indonesian Language Based on Wav2Vec 2.0*

One of the main problems that have plagued ready-to-use Automatic Speech Recognition (ASR) Systems is that there is less transparency in handling the user's voice data, that has raised concerns regarding the privacy of said data. On the other hand, developing an ASR system from scratch with good accuracy and can work offline requires a large amount of data, more specifically labeled voice data that has been transcribed. This becomes one of the main obstacles in speech recognition system development, especially in low-resourced languages where there is minimal data, such as Bahasa Indonesia. Based on that fact, this research conducts development of an automatic speech recognition system that is based on wav2vec 2.0, an Artificial Model that is known to recognize speech signals and convert it to text with great accuracy, even though it has only been trained with small amounts of labeled data. From the testing that was done using the Common Voice 8.0 dataset, the wav2vec 2.0 model produced a WER of 25,96%, which is twice as low as a traditional Bidirectional LSTM model that gave 50% WER, but required 5 times more labeled data in the training process. However, the wav2vec model requires more computational resource, which are 2 times more RAM and 10 times more storage than the LSTM model.

Key words: *deep learning, self-supervised learning, wav2vec 2.0, transformer networks, speech recognition*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1. Latar Belakang

Sistem pengenalan suara otomatis, atau *Automatic Speech Recognition* (ASR) *System*, merupakan gabungan komponen perangkat lunak dan keras yang dapat menerjemahkan suara percakapan seseorang menjadi bentuk teks tanpa perlu dituliskan secara manual satu huruf pun dari pembicara tersebut. Sistem ASR sudah dapat ditemui di kehidupan sehari-hari pada perangkat *voice assistant* seperti Google Home, Amazon Alexa ataupun fitur *voice typing* yang ditemukan dalam perangkat seperti pada fitur *Dictation* pada aplikasi Microsoft Word, yang memungkinkan penggunanya untuk mencatat teks hanya dengan menyuarakan percakapan yang ingin dicatat. Meskipun sudah mengalami perkembangan sejak lama, sistem ASR mengalami perkembangan yang pesat selama beberapa tahun terakhir yang disebabkan oleh perkembangan teknologi kecerdasan buatan, khususnya dalam bidang *Deep Neural Networks* (DNN) [1]. Penggunaan DNN yang menggantikan pendekatan *Hidden Markov Model* sebelumnya membuat sistem ASR menjadi lebih akurat terutama dalam menerjemahkan suara percakapan dengan keadaan terganggu, atau *noisy* [2].

Namun, terdapat beberapa permasalahan umum yang terdapat pada sistem ASR yang sudah banyak digunakan dalam kehidupan sehari-hari. Salah satu perhatian utama terdapat pada permasalahan privasi dan seberapa banyak percakapan yang disuarakan oleh masyarakat di seluruh dunia didengar atau disimpan oleh pembuat sistem ASR tersebut. Hal ini dikarenakan mayoritas sistem ASR yang terdapat pada kehidupan sehari-hari dilakukan secara daring (*online*), sehingga memerlukan transmisi data suara antara perangkat tersebut dengan sebuah *server* yang melakukan pengenalan suara, tanpa adanya transparansi terhadap alur dan proses dari pengiriman data melalui Internet. Berdasarkan riset, terdapat banyak informasi mengenai pengguna sistem ASR *online* seperti pada perangkat Amazon Echo hanya dengan percakapan, contohnya lokasi pengguna, kegiatan yang sedang dilakukan, data riwayat medis, waktu dan tanggal [3]. Selain itu, data

yang dikirimkan juga lebih rentan terhadap pencurian atau kebocoran data, seperti pada 283 kasus yang terjadi di wilayah Asia Pasifik pada tahun 2021 [4].

Salah satu solusi bagi masalah privasi yang menghantui sistem ASR adalah menggunakan sistem ASR secara luring melalui sebuah mikrofon yang terhubung dengan perangkat yang tidak terhubung ke jaringan luar yang menjalankan sistem ASR tersebut. Hal ini mengatasi masalah privasi yang terdapat pada sistem ASR yang bekerja secara daring, karena tidak adanya pengiriman data melalui Internet di dalamnya. Pada perangkat ini biasanya diterapkan teknologi DNN agar dapat menerjemahkan suara menjadi tulisan. Namun, solusi ini juga memiliki masalah yakni sistem ASR berbasis DNN yang harus dilatih terlebih dahulu dengan data yang sangat banyak. Sedangkan pengumpulan data yang sangat banyak ini sulit dilakukan, terlebih pada negara yang menggunakan bahasa dengan sumber daya minim atau *low-resource language*, seperti Bahasa Indonesia [2].

Untuk mengatasi permasalahan kebutuhan sumber daya bahasa tersebut, dibentuklah sebuah model pengenalan suara bernama wav2vec 2.0 yang memanfaatkan konsep *self-supervised learning* yang membantu model tersebut mempelajari data yang tidak diberikan label (*unlabeled data*) terlebih dahulu, sehingga tidak membutuhkan banyak data dengan label (*labeled data*) [5]. Berdasarkan penelitian terdahulu, proses pembelajaran *self-supervised learning* memungkinkan model wav2vec 2.0 mendapatkan hasil yang menyaingi dan melebihi model-model sebelumnya seperti Discrete BERT dan Noisy Student dalam kondisi data pelatihan dengan label yang minim untuk Bahasa Inggris [5]. Untuk mengoptimalkan sistem pengenalan suara, dibutuhkan pengujian variasi *learning rate* model wav2vec 2.0 yang memiliki performa terbaik dan kesalahan terendah, khususnya untuk melakukan pengenalan dan penerjemahan suara percakapan dalam Bahasa Indonesia. Model tersebut kemudian dapat diintegrasikan ke dalam sebuah sistem ASR yang bekerja secara luring, sehingga membentuk sistem penganalan suara otomatis yang menghormati privasi penggunanya.

1.2. Rumusan Masalah

Rumusan masalah dari penelitian ini adalah sebagai berikut:

1. Bagaimana cara meningkatkan dan mengoptimalkan performa model wav2vec 2.0 untuk pengenalan suara Bahasa Indonesia?
2. Bagaimana performa model wav2vec 2.0 jika dibandingkan dengan pendekatan jaringan saraf tiruan yang biasa digunakan dalam sistem pengenalan suara?

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Merancang sistem pengenalan suara (ASR) berbasis wav2vec 2.0 yang dapat bekerja secara luring.
2. Mengevaluasi model wav2vec 2.0 yang dibangun dengan model lain yang lebih standar.
3. Mengevaluasi *learning rate* terbaik dari model wav2vec agar mendapatkan akurasi tertinggi.

1.4. Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sistem pengenalan suara harus dapat bekerja secara luring tanpa adanya data yang dikirimkan melalui Internet.
2. Sistem pengenalan suara harus dapat menerjemahkan suara percakapan pada mayoritas kata dalam Bahasa Indonesia, berdasarkan himpunan data Common Voice Bahasa Indonesia.
3. Varian model wav2vec 2.0 yang dibandingkan berupa varian wav2vec 2.0 XLSR-53, dengan model berbasis LSTM.
4. Pengujian varian model wav2vec 2.0 dibatasi hanya kepada parameter yang ditentukan, yaitu akurasi yang diukur dalam WER dan penggunaan sumber daya komputasi memori dan penyimpanan pada komputer.

1.5. Metodologi Penelitian

Metode yang digunakan selama penelitian ini adalah sebagai berikut:

1. Studi literatur

   Dalam tahapan studi literatur, dilakukan pencarian, pengumpulan, dan pemahaman terhadap sumber lain yang berhubungan dengan DNN, sistem pengenalan suara atau ASR, model wav2vec 2.0, dan segala himpunan bagian teknologi lainnya yang berkaitan dengan penelitian ini.
2. Konsultasi dengan dosen pembimbing

   Konsultasi yang dilakukan dengan dosen pembimbing berupa pertemuan secara rutin setiap pekan untuk menambah wawasan, menerima masukan, dan membahas segala hambatan yang dialami mengenai topik kecerdasan buatan, *Deep Neural Networks*, ataupun topik-topik lainnya yang berkaitan dengan penelitian ini.
3. Pengumpulan Data

   Pengumpulan Data dilakukan dengan mencari sumber daya suara percakapan dalam Bahasa Indonesia yang dapat digunakan dalam penelitian.
4. Desain Sistem

   Desain sistem pengenalan suara dilakukan dengan membuat rancang bangun dari sistem yang diimplementasikan berdasarkan batasan masalah yang sudah ditentukan sebelumnya.
5. Rancangan Proses Pelatihan Model

   Rancangan proses pelatihan model wav2vec 2.0 dibentuk berdasarkan studi literatur yang telah ditentukan dan sumber-sumber lainnya sehingga dapat menghasilkan model dengan minim kesalahan.
6. Rancangan Proses Pengujian dan Perbandingan Model

   Rancangan proses pengujian dan perbandingan model wav2vec 2.0 XLSR-53 dibentuk sehingga dapat membandingkan kedua varian dengan sama rata, tanpa mengunggulkan varian mana pun.

7. Analisis

   Analisis diberikan terhadap hasil rancang bangun sistem pengenalan suara yang sudah dibentuk beserta terhadap kedua varian model wav2vec 2.0 yang dibandingkan.

8. Kesimpulan

   Pemberian kesimpulan secara keseluruhan terhadap penelitian yang telah dilakukan berdasarkan hasil yang ditemukan beserta dengan analisis yang didapatkan dari pembuatan rancang bangun sistem pengenalan suara berbasis wav2vec 2.0 dan perbandingan varian model wav2vec 2.0 agar dapat digunakan dalam sistem tersebut.

1.6. Sistematika Penulisan

Sistematika Penulisan pada penelitian ini dibagi ke dalam 4 bab, yaitu:

BAB I PENDAHULUAN

Bab ini menjelaskan latar belakang, tujuan, batasan masalah, serta metodologi yang digunakan sebagai acuan utama dari pelaksanaan penelitian.

BAB II DASAR TEORI

Bab ini berisi dasar teori yang telah digunakan dalam pembuatan rancang bangun sistem ASR beserta dengan landasan dari penggunaan dan perbandingan model wav2vec 2.0 yang dilakukan.

BAB III PERANCANGAN SISTEM AUTOMATIC SPEECH RECOGNITION

Bab ini menjelaskan detail dari rancang bangun sistem pengenalan suara yang telah dibentuk, informasi mengenai data dan pemrosesan data yang diterapkan dalam pelatihan dan pengujian data, serta rancangan proses pelatihan dan pengujian model wav2vec 2.0.

BAB IV IMPLEMENTASI DAN ANALISIS SISTEM AUTOMATIC SPEECH RECOGNITION

Bab ini membandingkan dan menganalisa performa dari implementasi sistem *automatic speech recognition* berbasis model wav2vec 2.0 dalam pengujian dengan model lain dan model wav2vec 2.0 dengan konfigurasi berbeda.

BAB V PENUTUP

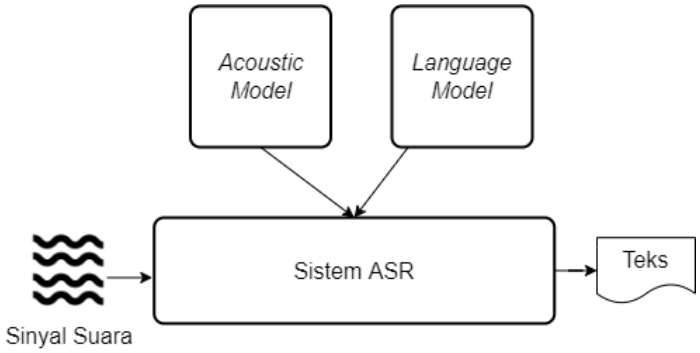
Bab ini berisi kesimpulan, yang terdiri dari hasil penelitian yang telah dilakukan beserta dengan poin-poin penting dari wawasan yang didapatkan selama penelitian berlangsung

BAB 2
SISTEM AUTOMATIC SPEECH RECOGNITION BERBASIS MODEL SELF-SUPERVISED LEARNING WAV2VEC 2.0

2.1. Automatic Speech Recognition

Automatic Speech Recognition (ASR) *Systems*, atau Sistem Pengenalan Suara Otomatis, merupakan sebuah sistem yang dapat mengenali dan memahami makna dari percakapan berbentuk sinyal suara dalam suatu bahasa [6]. Sistem ASR pada umumnya menerima masukan dalam bentuk sinyal suara, kemudian memrosesnya dan memberikan keluaran berupa makna yang ditemukan dalam sinyal suara tersebut dalam bentuk teks. Sistem ASR diperlukan karena pada dasarnya sebuah komputer mengalami kesulitan untuk memaknai sinyal suara yang diucapkan oleh seseorang, meskipun komputer tersebut sudah dapat menangkapnya melalui sebuah mikrofon. Dalam komputer tersebut, sinyal suara yang tersimpan hanya dalam bentuk berkas audio dengan format tertentu seperti mp3 atau wav. Sinyal suara tersebut harus dianalisis terlebih dahulu menggunakan sebuah pola atau algoritma tertentu, sehingga dapat diubah menjadi bentuk seperti angka, kata, ataupun teks yang lebih mudah dimengerti oleh komputer pada umumnya.

Pada awalnya, sistem atau komputer yang dapat mengenali ataupun berbicara seperti manusia hanya dapat dibayangkan dalam film-film fiksi ilmiah saja, seperti karakter “HAL” dalam film “Space Odyssey”, 2001 dan karakter “J.A.R.V.I.S” dalam “Iron Man”, 2008 [7]. Namun di zaman kini, sistem ASR tidaklah asing lagi untuk ditemukan dalam kehidupan sehari-hari. Contohnya pada *smart assistant* seperti produk Google Home dan Amazon Alexa, yaitu sebuah perangkat yang dapat mengenali percakapan atau pertanyaan yang diberikan kepadanya, kemudian membalasnya dengan mengeluarkan suara dengan jawaban yang tepat layaknya sebuah manusia. Sistem ASR juga merupakan sebuah komponen dalam semua perangkat yang bersifat *voice-controlled*, atau dikendalikan menggunakan suara manusia. Contohnya seperti pada kendaraan mobil modern yang menggunakan sinyal suara untuk mengendalikan dasbor, perangkat *smart home* seperti lampu dan mesin cuci.



Gambar 2.1. Skema Sistem Pengenalan Suara

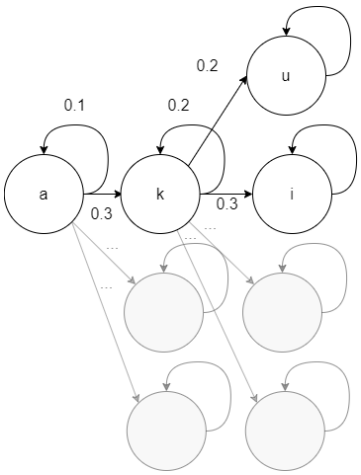
Gambar 2.1 menunjukkan sebuah sistem ASR yang pada umumnya terdiri dari 2 bagian utama, yakni *acoustic model* dan *language model* [8]. Pada *acoustic model*, dilakukan ekstraksi fitur yang mengubah suara dalam bentuk sinyal dalam komputer menjadi bentuk yang lebih mudah diproses, seperti vektor ataupun angka dalam skala tertentu, salah satunya bentuk *Mel Frequency Cepstral Coefficients* atau MFCC. Setelah dilakukan perubahan bentuk, maka dilakukan penentuan masing-masing bunyi yang membentuk sebuah kata. Hal ini pada umumnya dilakukan dengan menentukan fonem atau karakter bagi sebuah bunyi berdasarkan nilai probabilitas tertentu. Terakhir, *language model* merupakan bagian yang memberikan konteks terhadap sebuah kata atau bagaimana hubungan antara sebuah kata dengan kata lainnya dalam satu kalimat. *Language model* dapat dibentuk dengan mengumpulkan beragam kalimat dari suatu bahasa, sehingga didapatkan probabilitas satu kata muncul setelah kata lainnya.

2.1.1. Perkembangan Sistem ASR

Awal mula sistem penganalan suara otomatis terjadi pada tahun 1950-an, pada saat dibuatnya mesin pengenal angka 0-9 dari suara oleh Bell Telephone Laboratories dan mesin pengenal kosakata sederhana oleh Olson dan Belar [6] dari RCA Laboratories. Pada saat itu, mesin pengenalan suara bekerja dengan cara mengukur kemudian mencocokkan frekuensi resonansi yang dihasilkan oleh suara saat membuat suatu bunyi dengan pola yang sudah dikenali dan ditentukan sebelumnya [7]. Mesin pengenalan suara pada zaman tersebut masih memiliki banyak kekurangan, seperti kurangnya kemampuan mesin tersebut dalam

memahami banyak bunyi atau kata, serta memiliki ketergantungan pada pembicara dan kondisi yang spesifik, sehingga tidak dapat digunakan secara umum.

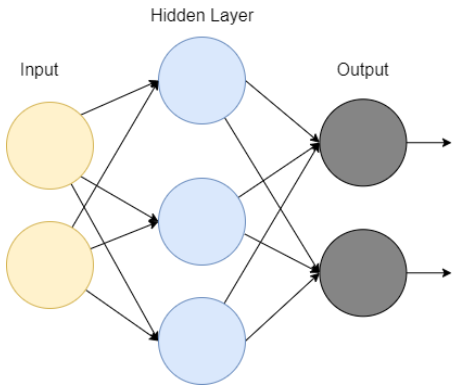
Terdapat sedikit perkembangan terjadi pada dekade selanjutnya dengan adanya bantuan komputer, seperti pada [9], di mana dibentuknya sebuah algoritma berbasis *Dynamic Programming* untuk memecahkan masalah pengenalan suara. Namun, perkembangan besar pada sistem ASR terdapat pada tahun 1970 hingga 1980-an [7]. Pertama, pada awal tahun 1970-an ditemukannya Sistem VIP-100, yang merupakan sistem ASR komersial pertama di dunia. Sistem ini mendorong adanya pendanaan dari pemerintahan Amerika Serikat untuk riset bidang pemahaman bahasa atau *Speech Understanding Research* (SUR) di negara tersebut, yang mengembangkan berbagai sistem ASR lainnya pada tahun-tahun berikutnya [7].



Gambar 2.2. Contoh *Hidden Markov Model* Dalam Sistem ASR

Perkembangan besar selanjutnya pada sistem ASR terdapat pada penggunaan *Hidden Markov Model* (HMM) dalam bagian *acoustic model*, yang mengandalkan peluang berdasarkan hubungan satu fonem dengan fonem lainnya pada semua kata. Gambar 2.2 menunjukkan sebuah HMM dalam Sistem ASR yang bekerja dengan memodelkan sebuah bunyi atau fitur yang telah didapatkan pada proses *feature extraction* kepada sebuah fonem dari kumpulan kosakata yang telah ditentukan. Hal tersebut dilakukan dengan melihat peluang terbesar dari kemunculan sebuah fonem berdasarkan fonem sebelumnya dan fitur yang sedang

diproses. Sebagai contoh, pada Gambar 2.2 sistem berhasil mengenali fonem “a”, yang kemudian memiliki peluang untuk dilanjutkan dengan mengulang fonem tersebut, atau dengan fonem lainnya. Keunggulan dari HMM dibandingkan dengan pendekatan sebelumnya merupakan kemampuannya mengenali percakapan dalam sebuah rangkaian sinyal suara yang panjang [6].



Gambar 2.3. Contoh *Artificial Neural Network*

Pada tahun 1980-an juga, terdapat perkembangan lain pada sistem ASR berupa penggunaan *Artificial Neural Networks* (ANN) yang melakukan pembelajaran dengan cara memprediksi bunyi atau fonem yang tepat bagi sebuah fitur. Contoh dari ANN sederhana dapat dilihat pada Gambar 2.3, yang memiliki masing-masing satu buah lapisan *input*, *hidden*, dan *output*. Jika hasil dari prediksi masih memiliki jarak yang jauh dengan teks asli pada transkrip suara yang diberikan, maka ANN mengatur nilai bobot yang dimilikinya dengan mengubah nilai bobot tersebut berdasarkan nilai gradien tertentu. Hal ini dilakukan sampai ditemukan nilai *loss* atau perbedaan yang sedikit antara hasil prediksi dengan teks transkrip asli mencapai nilai minimum. Setelah dilakukan pembelajaran, maka ANN diharapkan dapat menentukan fonem yang tepat bagi sebuah fitur dengan akurasi yang cukup tinggi. Keunggulan ANN dibandingkan pendekatan lainnya terdapat pada kemampuan menggeneralisasi atau mengenali sinyal suara dengan lebih banyak gangguan, namun pada saat itu masih mengalami kesulitan mengenali sinyal suara dengan rangkaian yang panjang [6].

Penggunaan *Deep Learning* merupakan perkembangan dari sistem ASR yang dipopulerkan pada tahun 2006. *Deep Learning* merupakan bagian dari

pembelajaran mesin yang dapat memodelkan relasi data yang kompleks dengan memanfaatkan analisis fitur menggunakan beberapa tingkatan atau level [10]. Contoh dari penggunaan *Deep Learning* terdapat pada *Deep Neural Networks* yang merupakan pengembangan dari sistem ANN yang memiliki jauh lebih banyak lapisan dan neuron. Sifat dari *Deep Learning* yang dapat menentukan relasi data yang kompleks membuatnya banyak digunakan dalam sistem pengenalan suara, seperti yang telah diimplementasikan pada [11]. Dari hasil yang ditemukan penggunaan *Deep Learning* pada sistem pengenalan suara memberikan hasil yang lebih baik dibandingkan dengan model HMM pada himpunan data yang sama. Permasalahan dalam penggunaan *Deep Learning* untuk sistem ASR hanya terdapat pada kebutuhannya terhadap sumber daya data suara yang jauh lebih banyak dibandingkan HMM dan ANN biasa, beserta dengan komputasi yang dibutuhkan untuk mengolah data tersebut.

2.1.2. Kriteria Penilaian Sistem ASR

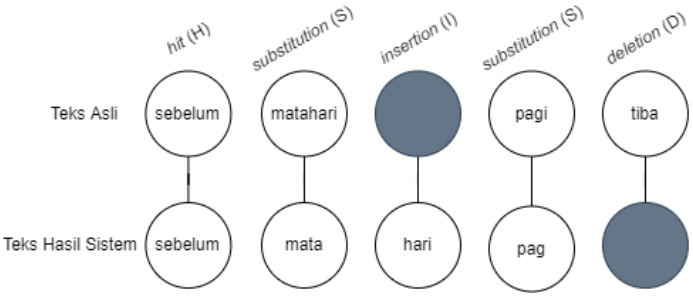
Agar dapat membedakan sistem penganalan suara yang baik dengan yang buruk, terdapat beberapa kriteria yang biasa digunakan pada penelitian terdahulu. Kriteria utama yang paling sering digunakan dalam sistem ASR pada saat ini adalah *Word Error Rate* (WER), yang menyatakan perbandingan antara jumlah kata yang diproses secara salah dengan jumlah yang diproses secara benar [12]. WER untuk sistem ASR dapat didefinisikan berdasarkan persamaan (2.1).

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{H + S + D} \tag{2.1}$$

Dengan:

- N = jumlah total yang ada pada teks asli
- H = jumlah kata yang dapat ditafsirkan dengan benar
- S = jumlah kesalahan kata akibat pergantian kata dari teks asli dengan kata lain
- D = jumlah kesalahan kata akibat adanya kata dari teks asli yang terhilang
- I = jumlah kesalahan kata akibat adanya kata di luar teks asli yang ditambahkan

Contoh perhitungan WER dapat dilihat pada Gambar 2.4.



Gambar 2.4. Contoh Perhitungan WER

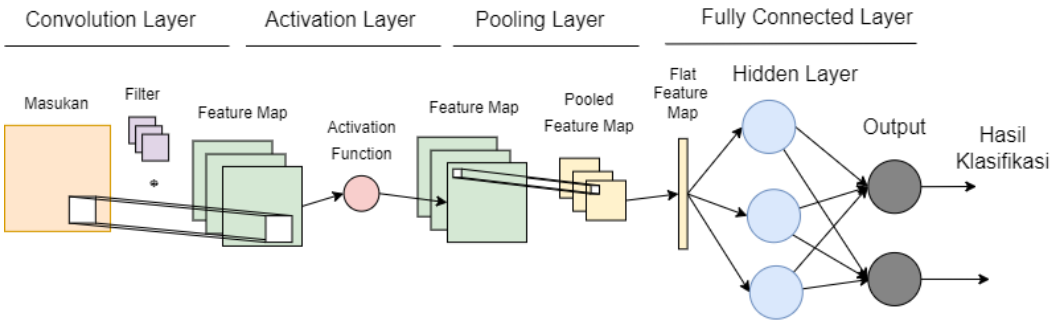
Gambar 2.4 mengilustrasikan perhitungan WER yang membandingkan kalimat teks hasil sistem berupa “sebelum mata hari pag” dengan teks transkrip asli “sebelum matahari pagi tiba”. Pertama, kata yang sudah dihasilkan secara benar oleh sistem seperti kata “sebelum” dihitung sebagai sebuah *hit*. Lalu, semua kata yang tergantikan dengan kata lainnya seperti kata “matahari” yang digantikan dengan “mata”, serta kesalahan penulisan kata seperti “pagi” menjadi “pag” dihitung sebagai *substitution* yang menandakan pergantian kata. Terakhir, untuk kata yang ditambahkan pada teks hasil sistem seperti “hari” dan kata yang tidak dideteksi oleh sistem seperti “tiba”, menggambarkan terjadinya *insertion* dan *deletion* pada teks.

Selain WER, terdapat dua kriteria lainnya yang juga digunakan dalam mengevaluasi sebuah sistem pengenalan suara, yaitu *Character Error Rate* (CER) dan *Phoneme Error Rate* (PER). Keduanya memiliki perhitungan yang serupa dengan WER namun pada masing-masing huruf dan fonem (dapat berupa kumpulan huruf) yang ada pada suatu kata. Meskipun lebih jarang digunakan dibandingkan WER, kedua kriteria ini tetap menjadi salah satu poin evaluasi performa bagi sebuah sistem ASR.

2.2. Convolutional Neural Networks

Convolutional Neural Networks (CNN) merupakan salah satu jenis jaringan saraf tiruan dalam *Deep Learning* yang pada umumnya digunakan untuk mengklasifikasi masukan berupa sebuah matriks (seperti gambar) terhadap label tertentu. CNN, yang pertama kali dikemukakan pada tahun 1989, merupakan model

pembelajaran mesin yang terbagi menjadi dua bagian, yakni bagian *feature extractor* (pengekstraksi fitur) dan bagian *classifier* (pengklasifikasi) [13]. Bagian pengekstraksi fitur berfungsi untuk menelaah informasi yang dibutuhkan dari masukan sehingga dapat melakukan klasifikasi yang tepat pada bagian pengklasifikasi. CNN sering kali digunakan untuk permasalahan pemrosesan dan klasifikasi gambar, seperti mengklasifikasikan gambar tulisan huruf atau angka.



Gambar 2.5. Arsitektur Umum Model CNN [13]

Gambar 2.5 menunjukkan arsitektur dari sebuah model CNN terdiri dari beberapa sub-bagian yang disebut sebagai *layer* (lapisan) [14]. Pertama, pada bagian pengekstraksi fitur terdapat *Convolution Layer*, yang merupakan bagian utama dalam pengekstraksi fitur. Lapisan ini bekerja dengan cara melakukan proses konvolusi pada masukan terhadap sebuah matriks penyaring, atau *filter*. *Filter* yang digunakan dapat berjumlah satu atau lebih, dengan masing-masing *filter* berfungsi untuk mengekstraksi satu fitur tertentu dari masukan yang diberikan. Jumlah *filter* yang digunakan menentukan jumlah matriks yang dihasilkan pada *Convolution Layer*. Secara matematis, proses konvolusi terdapat pada persamaan (2.2).

$$(I * h)[x, y] = \int_0^x \int_0^y I(x - i, y - j) \times h(i, j)\, didj \qquad (2.2)$$

Dengan *I* adalah matriks masukan dan *h* merupakan matriks *filter*, kemudian *x* dan *y* merupakan masing-masing titik yang ada dari matriks masukan tersebut. Pada proses konvolusi, terdapat nilai parameter *stride* yang menentukan seberapa jauh perpindahan dari *filter* yang konvolusi di atas masukan tersebut.

$$ReLu(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (2.3)$$

Setelah dilakukan proses konvolusi, maka nilai dari hasil yang disebut sebagai *feature map* dikalkulasi kembali melalui sebuah *activation function* seperti *Rectified Linear Unit* (ReLu) yang terdapat pada persamaan (2.3). Berfungsi untuk mengubah skala dari nilai yang dihasilkan pada hasil proses sebelumnya.

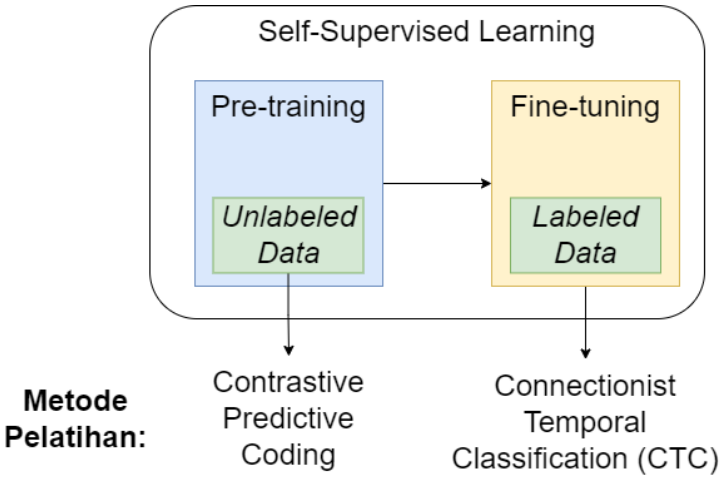
Lapisan selanjutnya pada arsitektur model CNN adalah *Pooling Layer*, yang berfungsi untuk mereduksi dimensi dari matriks yang telah dihasilkan, sehingga mengurangi komputasi yang dibutuhkan pada saat klasifikasi [14]. Hal tersebut dapat dilakukan dengan menggabungkan nilai-nilai yang berdekatan pada *feature map* melalui fungsi tertentu, seperti penjumlahan atau perhitungan rata-rata. Pada proses reduksi dimensi ini perlu ditentukan seberapa besar luas yang digabungkan dari matriks awal, beserta dengan *stride* yang dapat digunakan untuk melompati beberapa bagian dari *feature map* seperti pada proses konvolusi.

Proses ekstraksi fitur pada CNN dapat dilakukan secara berulang kali dengan ketiga lapisan pada pengekstraksi fitur menggunakan masukan berupa hasil dari proses ekstraksi fitur sebelumnya. Kemudian, agar hasil *feature* map akhir dapat digunakan dalam sebuah *Neural Network*, maka dilakukan proses *flattening* (perataan) terlebih dahulu pada awal lapisan *Fully Connected Layer*. Setelah itu, maka hasil ekstraksi fitur dapat digunakan sebagai masukan pada jaringan saraf tiruan seperti pada Gambar 2.5, yang memberikan hasil klasifikasi dari matriks awal yang diberikan.

2.3. Contrastive Predictive Coding

Pada umumnya, dalam pembelajaran mesin terdapat tiga jenis pembelajaran yang digunakan, yakni *supervised learning*, *unsupervised learning*, dan *semi-supervised learning* atau disebut juga *reinforcement learning*. *Supervised Learning* merupakan jenis pembelajaran berdasarkan data masukan yang sudah diberi label sebagai harapan hasil dari model, kemudian model mempelajari bagaimana cara mendapatkan label yang sesuai bagi setiap masukan. *Unsupervised Learning* mempelajari data tanpa adanya label atau diberi tahu hasil yang diharapkan terlebih

dahulu, dengan cara melihat pola-pola yang dapat disimpulkan dari data tersebut. *Reinforcement Learning* adalah proses pembelajaran di mana model diberikan sedikit data yang sudah diberikan label, namun secara bersamaan dengan data yang belum diberikan label dalam jumlah yang jauh lebih banyak [15].



Gambar 2.6. Proses Pembelajaran *Self-supervised Learning*

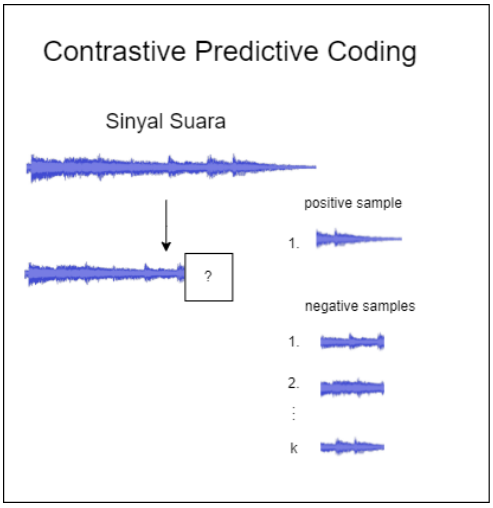
Berbeda dengan ketiganya, *self-supervised learning* merupakan jenis pembelajaran yang perlu diberikan data tanpa label (*unlabeled data*) dalam jumlah yang banyak beserta dengan data dengan label (*labeled data*) dalam jumlah yang lebih sedikit. Tujuan dari proses pembelajaran tersebut berupa mengurangi *labeled data* yang diperlukan untuk melatih model, namun menghasilkan akurasi yang sama atau lebih baik dibandingkan dengan model *supervised learning* lainnya [16]. Gambar 2.6 mengilustrasikan proses pembelajaran *self-supervised learning* yang dilakukan pada model ASR. Pada proses pembelajaran tersebut, terdapat dua tahap yang dijalankan, yaitu *pre-training* dan *fine-tuning*.

Pertama, model yang melakukan pembelajaran secara *self-supervised learning* menerima masukan berupa data tanpa label (*unlabeled data*) untuk mempelajari hubungan yang terdapat pada data tersebut beserta membentuk sebuah representasi darinya, dalam tahap yang disebut *pre-training*. Pada tahap ini, model melakukan pembaruan parameter bobot sehingga mempelajari fitur-fitur dari data dan menghasilkan representasi data yang menyimpan informasi mengenai fitur-fitur tersebut. Salah satu metode yang dapat digunakan untuk mengoptimalkan

parameter model dalam proses *pre-training* berupa *Contrastive Predictive Coding* (CPC) [17].

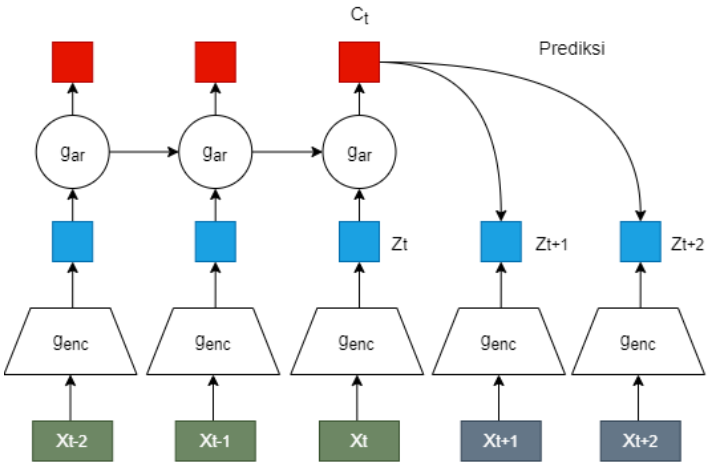
Setelah model melalui proses *pre-training*, maka model dilatih kembali dengan data yang diberikan label (*labeled data*) pada proses *fine-tuning*. Pada proses ini, model mempelajari masukan yang diberikan, kemudian memberikan hasil prediksi label dari masukan tersebut. Jika label hasil prediksi model belum sesuai dengan label asli dari masukan yang diberikan, maka model melakukan pembaruan parameter. Untuk sistem ASR, metode yang digunakan untuk mengoptimalkan parameter model adalah *Connectionist Temporal Classification* (CTC) [18].

Untuk menentukan struktur atau representasi umum dari data tanpa label pada proses *pre-training*, maka salah satu metode yang dapat digunakan adalah *Contrastive Predictive Coding* (CPC) [17]. *Contrastive Predictive Coding* bekerja dengan cara menghilangkan sebagian acak dari data secara sementara, kemudian membuat prediksi untuk mengembalikan bagian yang hilang tersebut. Data yang digunakan dapat bermacam-macam, seperti menghilangkan satu area dari gambar, satu kata dalam sebuah teks, atau cuplikan dengan durasi tertentu dari sinyal suara dengan durasi yang lebih panjang. Proses CPC terjadi pada tahap *pre-training* dalam *self-supervised learning*, sehingga data yang dibutuhkan hanya masukan yang diprediksi oleh model, tanpa perlu mengetahui hasil keluarkan yang diharapkan dari model tersebut.



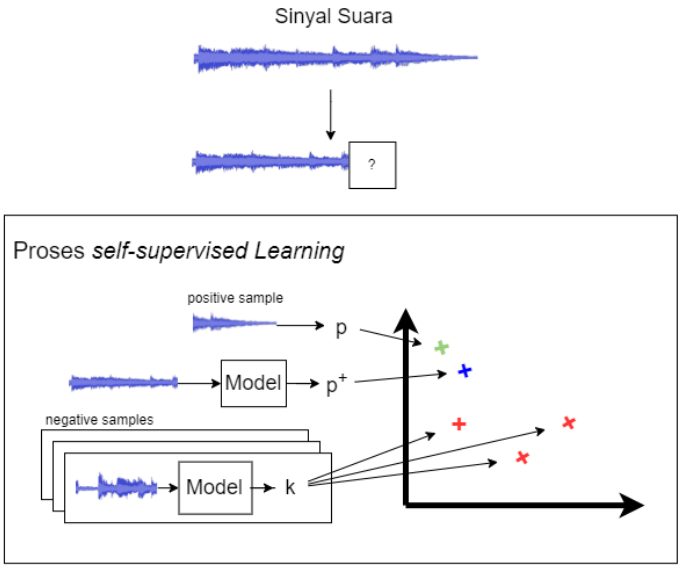
Gambar 2.7. Proses *Contrastive Predictive Coding* Pada Model ASR

Gambar 2.7 mengilustrasikan proses *Contrastive Predictive Coding* yang dilakukan pada model ASR dengan *self-supervised learning*. Pertama, terdapat sinyal suara yang didapatkan dari masukan model, kemudian dihilangkan satu cuplikan darinya dengan durasi tertentu. Cuplikan yang telah dihilangkan disebut sebagai *positive sample*. Selain dari *positive sample*, model juga mengambil secara acak sejumlah cuplikan lain dari semua masukan yang diberikan yang digunakan sebagai *negative sample*. Jika model *self-supervised learning* dapat mengklasifikasikan berbagai *positive sample* dari sejumlah *negative sample* lainnya, maka dapat dikatakan bahwa model telah mempelajari representasi umum dari data masukan yang diberikan, tanpa mengetahui kegunaan dari data tersebut yang didapatkan dari suatu label.



Gambar 2.8. Arsitektur *Self-Supervised Learning* Dengan CPC [17]

Gambar 2.8 merupakan salah satu contoh arsitektur model dalam tahap *pre-training* dengan metode CPC untuk melakukan pembelajaran secara *self-supervised learning*. Pertama, model melakukan proses berupa mengolah data yang diketahui (X_t, X_{t-1}, X_{t-2}, dst.) untuk memprediksi data yang dihilangkan dari masukan (X_{t+1}, X_{t+2}, dst.) [17]. Semua data tersebut diproses melalui *encoder* (g_{enc}) yang mengekstrak fitur-fitur yang dibutuhkan dari data, beserta dengan mereduksi dimensi dari data tersebut menjadi lebih sederhana. Salah satu *encoder* yang digunakan pada [17] dan [5] adalah CNN. Hasil dari *encoder* tersebut (Z_t) adalah *latent representation*, berupa representasi informasi mengenai fitur-fitur yang terdapat pada data. Kemudian, hasil *latent representation* tersebut diproses kembali ke dalam sebuah *autoregressive model* (g_{ar}) seperti RNN, GRU, atau LSTM, sehingga data yang tidak dihilangkan dapat dipelajari beserta dengan konteksnya terhadap data-data sebelumnya. Hasil akhir dari arsitektur CPC adalah prediksi terhadap data yang dihilangkan [17] [19].



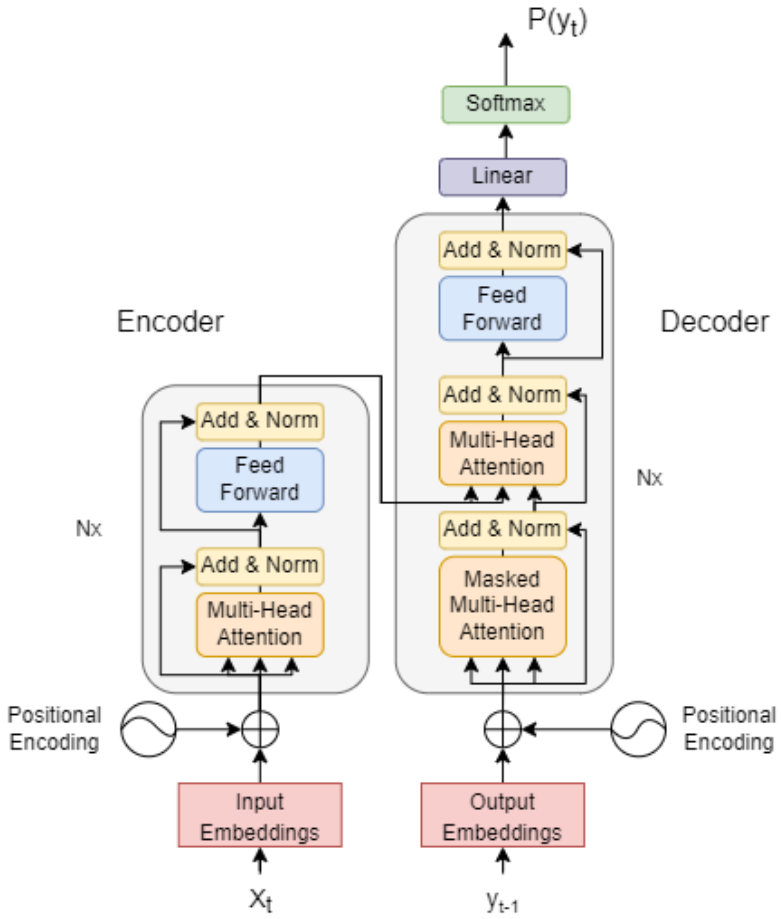
Gambar 2.9. Metode *Self-supervised Learning* Pada Sinyal Suara [20]

Agar model *self-supervised learning* dapat menentukan data yang dihilangkan dengan tepat, maka model dengan arsitektur CPC menggunakan *contrastive loss* yang menggambarkan relasi antar data dengan jarak. *Contrastive loss* memiliki konsep bahwa data dengan memiliki makna serupa atau dapat saling menggantikan memiliki jarak yang minimal, sedangkan jarak pada data yang tidak memiliki *makna* dimaksimalkan. Contoh dari metode *self-supervised learning* pada sebuah sinyal suara terdapat pada Gambar 2.9. Dari sebuah sinyal suara, diambil satu cuplikan yang disebut sebagai *positive sample* (*p*). Selain dari *positive sample* tersebut, model juga mengambil secara acak beberapa *negative sample* (k) dari semua masukan yang diberikan. Model harus dapat mengklasifikasikan *positive sample* di antara semua *negative sample*, kemudian membuat representasi baru dari sinyal suara yang mendahului *positive sample* tersebut (p^+). Setelah menentukan *positive sample*, model membuat jarak antara representasi data p^+ mendekati *p* dalam sebuah dimensi, serta menjauhi jarak antara representasi data tersebut dengan semua *negative sample* (k). Dengan melakukan proses tersebut, maka model *self-supervised learning* tidak hanya dapat membedakan sebuah *positive sample* dengan *negative samples*, namun juga dapat membentuk representasi baru dari data yang serupa dengan data yang dihilangkan (*positive sample*).

2.4. Transformer Networks

Transformer Networks, atau secara singkat *Transformer*, merupakan salah satu jenis model dalam *Deep Learning* yang pertama dikemukakan pada tahun 2017 dalam penelitian [21]. *Transformer* merupakan model yang digunakan untuk menerima dan mempelajari data *sequence*, atau kumpulan data yang berantai, kemudian memberikan sebuah prediksi berdasarkan data tersebut yang juga menghasilkan kumpulan data berantai (*sequence to sequence*). Model *Transformer* pada umumnya digunakan dalam pembelajaran *supervised learning*, namun dapat digunakan juga dalam *self-supervised learning*, seperti pada [16].

Model ini diajukan sebagai pengembangan dari model *autoregressive* yang sudah ada seperti RNN, LSTM, dan GRU. Ketiganya merupakan model yang membaca dan mempelajari sebuah rangkaian data secara satu per satu, kemudian menyimpan hasil pembelajaran pada setiap data masukan. Namun, ketiga model ini memiliki kelemahan serupa, yang dikembangkan lebih lanjut pada model *Transformer*. RNN memiliki permasalahan menghilangnya gradien yang disebabkan oleh rangkaian data yang panjang, sehingga informasi dari data yang sudah lama dimasukkan terhilang dan tidak mempengaruhi prediksi pada baru. Contohnya, agar model dapat memprediksi kata "Indonesia" dari kalimat "Saya dari Indonesia dan bisa berbicara Bahasa ___", maka model harus mempelajari hubungan dari kata "Indonesia" pada awal kalimat. Hal ini menjadi masalah bagi RNN pada kalimat yang panjang, karena hilangnya gradien tersebut. Selain itu, ketiga model tersebut membutuhkan waktu yang lama untuk melakukan pembelajaran, karena harus memasukkan rangkaian data secara satu per satu [21]. Hal ini menghambat penggunaan *Graphics Processing Unit* (GPU) pada komputer untuk dapat melakukan paralelisasi, karena masukan harus tetap berurutan.



Gambar 2.10. Arsitektur Umum *Transformer* [21]

Arsitektur *Transformer Networks* seperti yang terlihat pada Gambar 2.10 terbagi menjadi dua bagian, yaitu *encoder* dan *decoder*. *Encoder* berfungi untuk mempelajari data masukan yang diberikan (X_t), sedangkan *decoder* berfungsi untuk mempelajari data masukan beserta dengan semua keluaran yang dikeluarkan oleh model *Transformer* sebelumnya (y_{t-1}), sehingga dapat membentuk prediksi rangkaian data selanjutnya ($P(y_t)$). Masing-masing *encoder* dan *decoder* dapat ditumpukkan dengan *encoder* atau *decoder* lainnya (N_x), sesuai dengan arsitektur yang diinginkan. Selain itu, masukan yang diberikan baik pada *encoder* dan *decoder* tidak harus dilakukan secara satu per satu, melainkan dapat dilakukan secara *batch*. Kedua hal ini membuka peluang untuk melakukan paralelisasi pada model *Transformer*, sehingga mengurangi waktu yang dibutuhkan dalam proses pelatihan secara signifikan [21].

Pertama, baik pada bagian *encoder* ataupun *decoder*, semua masukan yang diterima oleh model diubah menjadi bentuk *embeddings* terlebih dahulu. *Embeddings* merupakan sebuah representasi matriks yang diberikan bagi data sehingga mempermudah proses pengolahan data yang dilakukan. Satu baris pada matriks merepresentasikan satu sampel data, yang memiliki representasi fitur sejumlah dengan kolom matriks tersebut. Salah satu contoh *embedding* adalah GloVe: Global Vectors for Word Representation yang merepresentasikan kata dalam bentuk vektor dan merupakan salah satu *embeddings* yang paling sering digunakan dalam pemrosesan bahasa natural [22]. Pada GloVe dan jenis *embedding* lainnya, data yang memiliki relasi dekat atau makna serupa memiliki jarak yang rendah, dan sebaliknya.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right) \tag{2.4}$$

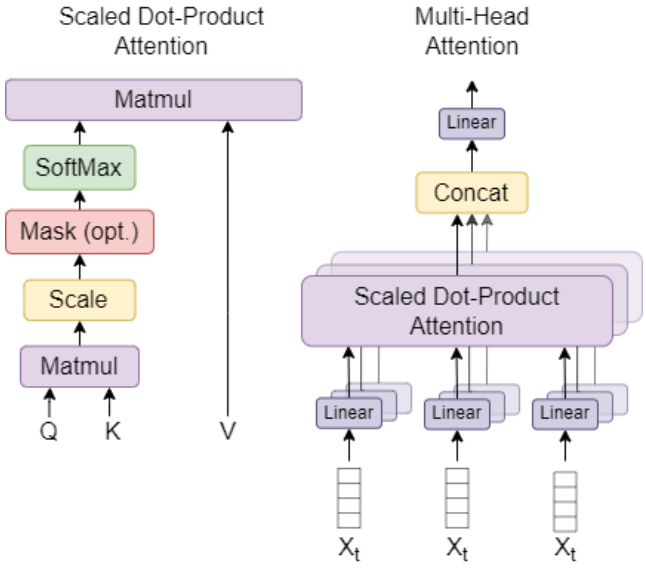
$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right) \tag{2.5}$$

Dengan:

- PE = representasi posisi dari sebuah data di dalam kumpulan data lainnya
- PO = posisi dari sebuah data di dalam kumpulan data lainnya
- i = posisi satu fitur antara fitur lainnya pada satu sampel data
- d_{model} = dimensi kolom matriks atau jumlah fitur yang digunakan untuk merepresentasikan data

Setelah mengubah semua masukan menjadi bentuk representasi vektor, maka dilakukan juga *positional encoding* yang berfungsi untuk memberikan konteks urutan data tersebut dalam rangkaian data. Hal ini diperlukan karena perubahan sebuah data menjadi representasi vektor, seperti sebuah kata pada model GloVe, menghilangkan posisi data tersebut terhadap sampel data lainnya, seperti posisi sebuah kata dalam satu kalimat.

Terdapat banyak fungsi yang dapat digunakan untuk memasukkan informasi pada data, namun penggunaan fungsi yang sederhana seperti menjumlahkan atau mengalikan posisi data dengan representasinya dapat mengubah skala dari data tersebut pada kumpulan data yang panjang. Oleh karena itu, pada [21] digunakan fungsi untuk menentukan *positional encoding* berupa fungsi sinus dan kosinus yang terdapat pada persamaan (2.4) dan persamaan (2.5) secara berurutan. Kedua fungsi tersebut tidak hanya menambahkan informasi posisi pada representasi data dan dapat digunakan untuk berapa pun panjang kumpulan data yang digunakan. Kedua fungsi tersebut memiliki hasil yang serupa sehingga dapat digunakan secara bergantian [21]. Pada model *Transformer Networks*, *positional encoding* yang didapatkan kemudian dijumlahkan dengan representasi awal dari data sehingga mendapatkan representasi baru yang di dalamnya terhadap konteks mengenai posisi atau indeks data dalam kumpulannya.



Gambar 2.11. Arsitektur *Multi-Head Attention* Pada *Transformer Networks* [21]

Transformer mengatasi permasalahan hilangnya gradien (*vanishing gradient problem*) dengan menggunakan arsitektur *encoder-decoder* yang memanfaatkan mekanisme *attention*. *Attention* berfungsi untuk mempelajari hubungan yang terdapat pada data, namun tetap menyimpan informasi mengenai hubungan data tetap tersimpan meskipun data tersebut sudah lama tidak diproses. Semua blok *Multi-Head Attention* baik pada *encoder* ataupun *decoder* memiliki 3

lapisan *linear* serupa dengan *fully-connected layer* pada *neural network* pada umumnya, yang memberikan 3 keluaran vektor yaitu *query* (Q), *key* (K), dan *value* (V). Vektor *query* dan *key* digabungkan melalui perkalian vektor, kemudian dilakukan pengaturan skala, yaitu membagi hasil vektor perkalian dengan akar kuadrat dari dimensi data.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.6}$$

Setelah itu, digunakan fungsi *softmax* pada persamaan (2.6) yang mengubah nilai elemen dari vektor tersebut menjadi skala 0 hingga 1, dengan hasil penjumlahan semua elemen vektor bernilai 1. Nilai 1 pada fungsi *softmax* yang digunakan merepresentasikan hubungan yang dekat antara dua data.

Hasil vektor dari blok *Scaled Dot-Product Attention* digunakan sebagai masukan bagi sebuah blok *linear*. Namun, blok *Multi-Head Attention* pada umumnya menggunakan lebih dari satu *Scaled Dot-Product Attention*. Oleh karena itu, terdapat blok *concat* yang menggabungkan semua keluaran terlebih dahulu sebelum dimasukkan kepada blok *linear*.

Setelah melalui blok *Multi-Head Attention*, dilakukan proses penjumlahan dan normalisasi hasil dengan data awal. Kemudian, hasil normalisasi dilewatkan ke sebuah blok *Feed Forward* yang berisi beberapa blok *linear* dengan *activation function* tertentu di dalamnya. Penggunaan penjumlahan dan normalisasi sebelum blok *Feed Forward* membantu model untuk tetap menangkap gradien dan hubungan yang terdapat pada data.

Bagian *decoder* terdiri dari blok-blok yang sama seperti *encoder*, namun memiliki sedikit variasi di dalamnya. Pertama, setelah melakukan proses *embedding* dan *positional decoding* terdapat sebuah blok *Masked Multi-Head Attention*. Blok ini serupa dengan blok *Multi-Head Attention* pada umumnya, namun menutup bagian-bagian dari label yang seharusnya belum diketahui oleh model. Hal ini dilakukan untuk memastikan keluaran pada suatu waktu hanya bergantung pada keluaran-keluaran sebelumnya, tidak setelahnya [21].

Setelah melalui blok *decoder*, maka hasilnya beserta dengan hasil dari *encoder* digabungkan melalui blok *Multi-Head Attention* yang sama, sehingga dapat ditentukan relasi antara data masukan dan label data. Hasil akhir dari *decoder* merupakan probabilitas *dataset* selanjutnya yang muncul pada rangkaian data, yang telah teroptimasi melalui semua proses yang sudah dilalui.

Tabel 2.1. Contoh Matriks Hasil Mekanisme *Self-Attention* Pada Kalimat "Saya Suka Nasi Goreng"

Key / *Query*	Saya	Suka	Nasi	Goreng
Saya	0.89	0.5	0.1	0.1
Suka	0.2	0.7	0.2	0.1
Nasi	0.1	0.2	0.9	0.4
Goreng	0.1	0.3	0.66	0.85

Tabel 2.2. Contoh Matriks Hasil Mekanisme *Attention* Pada Kalimat "Saya Suka Nasi Goreng"

Key / *Query*	Saya	Suka	Nasi	Goreng
I	0.7	0.5	0.1	0.1
Like	0.1	0.6	0.2	0.1
Fried	0.3	0.2	0.5	0.8
Rice	0.3	0.3	0.9	0.4

Tabel 2.1 dan Tabel 2.2 menunjukkan contoh mekanisme *attention* dalam sebuah *Tranformer Network*, pada sebuah permasalahan penerjemahan teks Bahasa Indonesia menjadi Bahasa Inggris. Pada kedua tabel, masing-masing kata yang terdapat pada baris tabel (*query*), memiliki suatu nilai di antara 0 dan 1 yang merepresentasikan seberapa besar hubungan kata tersebut pada masing-masing kolom tabel (*key*). Angka pada Tabel 2.1 merupakan nilai hubungan antara satu sampel data dengan yang lainnya pada kumpulan data, yang didapatkan setelah melalui proses *self-attention*. Pada Tabel 2.2, terdapat hasil dari proses *attention* yang merepresentasikan hubungan antara data yang diproses dengan hasil yang diharapkan dari data tersebut. Semakin tinggi nilai hasil mekanisme *attention* memiliki arti bahwa pasangan *query* dan *key* tersebut memiliki hubungan makna yang kuat, yang perlu dipelajari oleh model *Transformer*. Sebagai contoh, pada Tabel 2.2 kata "*fried*" memiliki nilai yang tinggi terhadap kata "goreng", karena memiliki persamaan makna meskipun berada pada posisi berbeda dalam kedua kalimat. Hasil dari kedua tabel ini tidak hanya merepresentasikan hubungan antara

sesama masukan pada model, namun juga hubungan masing-masing masukan terhadap semua keluaran.

Tabel 2.3. Contoh Hasil Dari Mekanisme *Masked Self-attention*

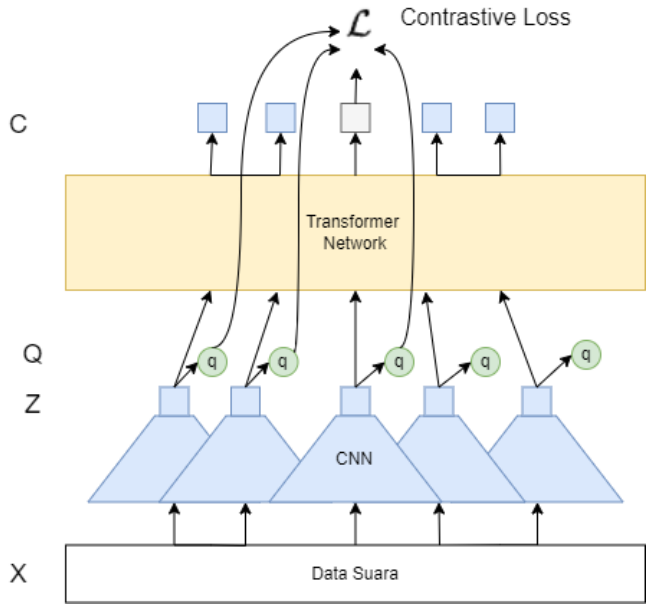
Key / *Query*	*I*	*Like*	*Fried*	*Rice*
I	0.7			
Like	0.1	0.6		
Fried	0.3	0.2	0.6	
Rice	0.3	0.3	0.3	0.8

Tabel 2.3 menunjukkan mekanisme *masked self-attention* yang terdapat pada blok *decoder* dalam sebuah *Transformer Network*. Model yang digunakan pada contoh tersebut memiliki konteks yang sama seperti Tabel 2.1 dan Tabel 2.2, yaitu menerjemahkan teks dalam Bahasa Indonesia menjadi Bahasa Inggris. Perbedaan mekanisme *self-attention* pada blok *decoder* dibandingkan dengan blok *encoder* terdapat pada proses *masking*, berupa penutupan hasil-hasil yang belum perlu diketahui oleh model *Transformer* pada saat melakukan prediksi. Hal tersebut dilakukan untuk menghindari adanya informasi yang tidak dapat diketahui oleh model pada saat melakukan penerjemahan. Sebagai contoh, jika model *Transformer* memulai penerjemahan dengan kata "I", maka model harus menentukan kata selanjutnya hanya dari masukan Bahasa Indonesia yang diberikan beserta dengan kata "I" tersebut. Kata-kata lain yang terdapat pada hasil penerjemahan saat pelatihan model diabaikan terlebih dahulu, karena tidak dapat diketahui oleh model pada saat penyusunan kalimat dilakukan dalam kondisi pengujian. Setelah model membuat penerjemahan baru untuk kata selanjutnya, kata tersebut baru dibandingkan dengan *key* selanjutnya, yaitu kata "Like".

2.5. Wav2vec 2.0

Wav2vec 2.0 adalah sebuah model yang dikembangkan oleh beberapa pada peneliti dari Facebook AI pada tahun 2020 [5]. Model ini dapat mempelajari representasi vektor dari masukan sinyal suara dengan proses pembelajaran *self-supervised learning* (dijelaskan pada sub bab 2.3). Salah satu kegunaan utama dari model ini terdapat dalam pengembangan sistem pengenalan suara otomatis terutama bagi bahasa dengan sumber daya minim, di mana sulit untuk mendapatkan

himpunan data dengan label. Pada awal dikemukakan, model ini mendapatkan hasil 4.8/8.2 WER untuk Bahasa Inggris dengan 53 ribu jam data percakapan, namun hanya dengan 10 menit data yang diberikan label.



Gambar 2.12. Arsitektur Model wav2vec 2.0 [5]

Model wav2vec 2.0 terdiri dari beberapa bagian, yakni *feature encoder*, *contextualized representation with Transformer* (dijelaskan pada sub bab 2.4), dan *quantization module*. Pertama, terdapat *feature encoder* yang terdiri dari beberapa blok CNN yang diikuti dengan *activation function* GELU. Bagian ini berfungsi untuk menentukan fitur-fitur penting dari masukan sinyal suara (X) yang direpresentasikan menjadi bentuk vektor (Z). Setelah itu, hasil representasi tersebut dimasukkan ke dalam sebuah arsitektur *Transformer* untuk mempelajari hubungan yang terdapat pada data tersebut dan menghasilkan sebuah representasi vektor dari sinyal suara beserta dengan informasi relasi antar sinyal suara tersebut (C). Dalam proses pelatihan, terdapat bagian *quantization module* yang mengubah nilai representasi awal (Z) menjadi nilai diskrit, berdasarkan nilai-nilai yang tersimpan dalam sebuah *codebook* (Q).

$$\mathcal{L} = \mathcal{L}_m + \alpha \mathcal{L}_d \tag{2.7}$$

$$\mathcal{L}_m = -\log\left[\frac{\exp\left(\frac{sim(c_t, q_t)}{k}\right)}{\sum_{\tilde{q}\sim Q_t}\exp\left(\frac{sim(c_t, \tilde{q})}{k}\right)}\right] \tag{2.8}$$

$$\mathcal{L}_d = \frac{1}{GV}\sum_{g=1}^{G}\sum_{v=1}^{V}\bar{p}_{g,v}\log\bar{p}_{g,v} \tag{2.9}$$

$$sim(a, b) = \frac{a^T b}{\|a\|\|b\|} \tag{2.10}$$

Dengan:

- $\mathcal{L}$ = Nilai *loss* pada model
- $\mathcal{L}_m$ = Nilai *contrastive loss* pada model
- $\mathcal{L}_m$ = Nilai *diversity loss* pada model
- α = Sebuah parameter pengatur skala *diversity loss* yang dipelajari oleh model
- c_t = Hasil representasi keluaran kontekstual (keluaran akhir) model pada waktu t yang dihilangkan
- q_t = Hasil representasi keluaran kuantisasi (Q) model pada waktu t yang termasuk ke dalam sampel positif
- $\tilde{q}$ = Hasil representasi keluaran kuantisasi (Q) model yang termasuk ke dalam sampel negatif
- k = Jumlah sampel negatif yang digunakan
- sim = Sebuah fungsi *similarity* yang menunjukkan kedekatan antara dua nilai
- G = Jumlah *codebook* yang digunakan
- V = Jumlah entri di dalam setiap *codebook*
- $\bar{p}_{g,v}$ = Probabilitas penggunaan entri v pada suatu *codebook* g

Pada saat dilakukan proses pembelajaran *pretraining* dalam *self-supervised learning* wav2vec 2.0, terdapat sebuah nilai *loss* yang dihitung melalui persamaan

(2.7). Nilai *loss* pada persamaan tersebut terdiri dari dunia nilai *loss* lainnya yang diminimalkan, yaitu *contrastive loss* ($\mathcal{L}_m$) dan *diversity loss* ($\mathcal{L}_d$), masing-masing terdapat pada persamaan (2.8) dan (2.9) secara berurutan. *Contrastive loss* merepresentasikan akurasi model wav2vec dalam mengklasifikasi sebuah sampel representasi data suara pada satu bagian yang telah dihilangkan dari data masukan (c_t). Hal tersebut dilakukan dengan mencari representasi data suara yang mendekati representasi yang dihilangkan, atau dapat disebut sampel positif (q_t), di antara representasi data suara lain yang diambil secara acak dari *dataset*, yang dapat disebut sebagai sampel negatif ($\tilde{q}$). Untuk meminimalkan nilai dari *contrastive loss*, maka nilai dari kedekatan yang dihitung dengan persamaan *similarity* antara c_t dengan q_t harus bernilai kecil, sedangkan nilai antara c_t dengan semua sampel negatif $\tilde{q}$ harus bernilai lebih besar. Fungsi *similarity* yang digunakan [5] adalah *cosine similarity* antara dua nilai a dan b seperti pada persamaan (2.10).

Diversity loss merepresentasikan akurasi model wav2vec dalam menggunakan *codebook* atau nilai representasi kuantisasi untuk menggantikan hasil keluaran dari model tersebut pada proses ekstraksi fitur. Nilai representasi kuantisasi tersebut merupakan salah satu parameter yang dipelajari oleh model seperti melalui metode *clustering*. Proses kuantisasi hasil representasi sinyal suara yang mengubah masukan dari model wav2vec menjadi representasi diskrit ditemukan memberikan hasil WER yang lebih rendah dibandingkan dengan pendekatan tanpa kuantisasi dan pendekatan kuantisasi dengan nilai yang sudah ditetapkan (*fixed quantization*) [23]. *Loss* tersebut dapat dikalkulasikan dengan menentukan nilai *entropy* yang dihasilkan pada probabilitas penggunaan sebuah entri v dalam *codebook* g yang dinotasikan pada persamaan (2.9).

Setelah melalui proses *pretraining*, maka model wav2vec dilatih dengan data yang sesuai dengan permasalahan (*downstream task*) yang ingin diselesaikan. Pada proses ini, ditambahkan sebuah blok *linear* yang diberi masukan hasil representasi konteks data (C), berfungsi untuk melakukan klasifikasi representasi konteks sinyal suara terhadap fonem yang sudah didapatkan dari data dengan label. Sebagai contoh, dalam proses penerjemahan sinyal suara menjadi Bahasa Indonesia, maka blok *linear* tersebut memetakan hasil representasi yang telah dibentuk oleh

model menjadi karakter-karakter dalam Bahasa Indonesia. Pada proses *fine-tuning*, dapat digunakan fungsi *loss* yang berbeda untuk menentukan akurasi model dalam mengolah suara menjadi teks, seperti dengan metode CTC Loss [18].

Dalam implementasi model wav2vec yang digunakan pada penelitian terdahulu, terdapat beberapa varian model yang dapat digunakan. Varian tersebut ditentukan berdasarkan arsitektur model *Transformer* yang digunakan pada model wav2vec 2.0 beserta dengan data yang digunakan dalam proses *pretraining* untuk melatih model tersebut. Berdasarkan arsitektur model, terdapat dua varian wav2vec 2.0, yaitu wav2vec 2.0 Base yang memiliki 12 blok *Transformer*, besaran dimensi model 768, besaran dimensi internal model 3072, dan 8 *Attention Head*, dan wav2vec 2.0 Large yang memiliki 24 blok *Transformer*, besaran dimensi model 1024, besaran dimensi internal model 4096, dan 16 *Attention Head* [5]. Dalam proses *pretraining*, kedua varian model tersebut dilatih dengan menggunakan 250.000 dan 320.000 sampel data audio percakapan Bahasa Inggris secara berurutan. Kedua model tersebut juga dapat diturunkan menjadi varian lain dengan arsitektur yang sama, namun menggunakan data pelatihan yang berbeda dalam proses *pretraining*. Salah satu contoh varian yang dibentuk berdasarkan arsitektur wav2vec 2.0 Large merupakan varian wav2vec 2.0 XLSR-53 [24]. Varian wav2vec 2.0 XLSR-53 dilatih pada proses *pretraining* menggunakan kombinasi dari dataset MLS, CommonVoice, dan Babel, yang berisi sinyal audio percakapan dalam berbagai bahasa.

2.6. Common Voice

Dataset Common Voice 8.0 terdiri dari berkas audio percakapan dalam berbagai bahasa beserta dengan transkrip dari berkas audio yang dikumpulkan secara sukarela oleh lebih dari 340 individu melalui situs web Common Voice [25]. Salah satu bahasa yang disediakan dari *dataset* Common Voice adalah himpunan Bahasa Indonesia. Pada himpunan data Common Voice Bahasa Indonesia terdapat sekitar 50 jam berkas audio dengan format berkas MP3, yang memiliki ukuran berkas total sebesar 1.3 GB.

Tabel 2.4. Persebaran Bagian *Dataset* CommonVoice 8.0 Bahasa Indonesia

Nama Bagian	Jumlah Baris Data
train	5.302
dev	3.207
test	3.608
validated	22.874
other	22.385
invalidated	2.442
Total	59.818

Dataset Common Voice Bahasa Indonesia terbagi menjadi beberapa bagian, di antaranya *train*, *test*, *dev*, *validated*, *invalidated*, dan *other*. Jumlah persebaran data dapat dilihat pada Tabel 2.4. Persebaran Bagian *Dataset* CommonVoice 8.0 Bahasa Indonesia. Masing-masing bagian tersebut ditentukan berdasarkan ulasan yang telah diberikan oleh pengguna lainnya. Dataset bagian *train*, *test*, dan *dev*, merupakan bagian yang sudah diberikan validasi terhadap kualitas data yang didapatkan, sehingga sudah cocok untuk digunakan dalam pelatihan model kecerdasan buatan seperti model pengenalan suara yang dirancang. Bagian lainnya seperti *other* dan *invalidated* merupakan bagian yang belum diberikan ulasan oleh pengguna lainnya sehingga belum tentu merupakan data dengan kualitas baik, namun bagian-bagian tersebut adalah sumber utama yang mengandung mayoritas data dari himpunan data Common Voice Bahasa Indonesia.

2.7. Penelitian Terkait

Terdapat beberapa penelitian terkait penggunaan wav2vec 2.0 sebagai model pengenalan suara dalam berbagai bahasa. Penelitian pertama untuk model wav2vec 2.0 melakukan uji coba terhadap himpunan data Librispeech, yang merupakan sebuah himpunan data suara percakapan berbahasa Inggris [5]. Penelitian tersebut membandingkan hasil WER dua varian model wav2vec 2.0 (wav2vec 2.0 Base dan wav2vec 2.0 Large) dengan hanya diberikan 10 menit data suara pada saat dilakukan *fine-tuning*.

Tabel 2.5. Hasil WER Model Wav2Vec 2.0 Pada *Dataset* Librispeech Dengan Durasi Total 10 Menit Untuk *Fine-tuning* [5].

Model	Language Model	dev		test	
		clean	other	clean	other
10 menit data dengan label					
Base	4-gram	8.9	15.7	9.1	15.6
	Transformer	6.6	13.2	6.9	12.9
Large	Transformer	6.6	10.6	6.8	10.8
	Transformer	4.6	7.9	4.8	8.2

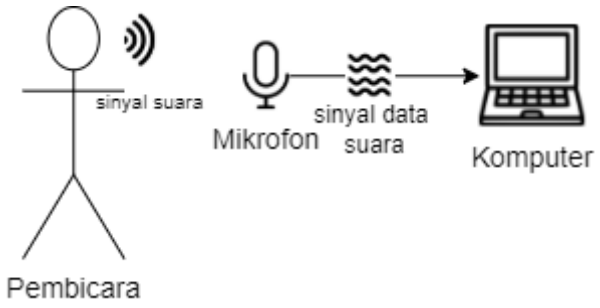
Selain penelitian tersebut, terdapat penelitian [26] yang membandingkan performa wav2vec 2.0 Base dan wav2vec 2.0 Large pada bahasa lain, beberapa dengan sumber daya minim, yaitu Bahasa Mandarin, Jepang, Inggris, Arab, Jerman, dan Spanyol. Pada penelitian tersebut, ditemukan bahwa varian wav2vec 2.0 Large memiliki performa terbaik untuk semua bahasa, terutama ketika ditambah dengan sebuah *language model*. WER yang ditemukan bernilai 35.62%, 28.16%, 28.32%, 25.7%, dan 39.11% untuk Bahasa Arab, Mandarin, Jepang, Jerman, dan Spanyol secara berurutan, ditambah dengan bahasa terbaik yang merupakan bahasa Inggris dengan angka 16.07%.

Pada penelitian [24] dan [27] dilakukan pengujian terhadap varian model wav2vec 2.0 XLSR-53 yang sudah dilatih dalam proses *pretraining* dalam berbagai bahasa, salah satunya dengan penggunaan Bahasa Indonesia. Himpunan data yang digunakan merupakan gabungan dari Common Voice, Multilingual Librispeech, dan BABEL, dengan pengujian dilakukan hanya pada beberapa bahasa yang ditentukan, seperti Italia, Spanyol, dan Turki. Dalam pengujian model wav2vec XLSR-53, ditemukan bahwa penggunaan berbagai bahasa dalam proses *pretraining* memberikan hasil yang lebih baik dibandingkan pelatihan yang hanya dilakukan dengan data suara percakapan dalam satu bahasa [24]. Selain itu, model ini juga efektif untuk digunakan dalam penerjemahan bahasa yang tidak digunakan dalam data *pretraining* ataupun *finetuning* sebelumnya, dengan melakukan pencocokan fonem data pada bahasa yang digunakan saat pelatihan dengan fonem data pada bahasa yang diuji [27]. Meskipun model tersebut tidak diuji kembali dalam Bahasa Indonesia, ditemukan bahwa model wav2vec 2.0 XLSR-53 dapat menyaingi hasil dari varian model wav2vec 2.0 Large pada berbagai bahasa.

BAB 3
PERANCANGAN SISTEM AUTOMATIC SPEECH RECOGNITION

3.1. Desain Arsitektur Sistem

Pada rancang bangun sistem pengenalan suara otomatis yang digunakan, terdapat sebuah komputer yang dihubungkan dengan satu mikrofon. Sistem ini menangkap sinyal suara dari seorang pembicara melalui mikrofon, kemudian memberikan keluaran berupa teks yang ditampilkan pada layar. Saat sinyal data suara dikirim oleh mikrofon dan diterima oleh komputer, maka digunakan sebagai masukan dari proses pengenalan suara. Sistem tersebut diilustrasikan secara visual pada Gambar 3.1.

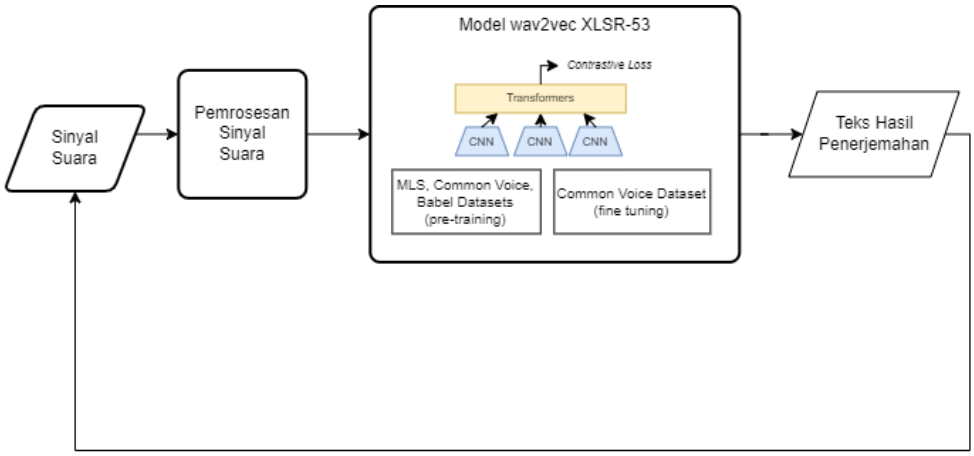


Gambar 3.1. Rancangan Sistem Pengenalan Suara

Proses yang dijalankan pada komputer dalam sistem merupakan sebuah program Python yang di dalamnya mengolah sinyal suara dengan sebuah model kecerdasan buatan. Model ini kemudian merepresentasikan bagian-bagian dari sinyal suara yang masuk menjadi nilai-nilai vektor yang dapat diterjemahkan menjadi huruf-huruf atau bunyi pembentuk suatu kata. Setelah memproses sinyal suara yang diberikan dari mikrofon kepada komputer, program Python tersebut memberikan keluaran berupa transkrip atau kata-kata yang telah diucapkan melalui sinyal suara dalam bentuk teks.

3.2. Desain Model Pengenalan Suara

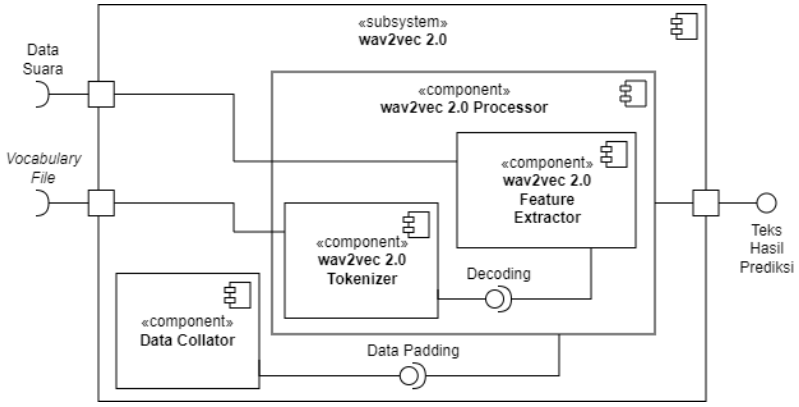
Model pengenalan suara yang telah didesain memiliki masukan berupa sinyal suara yang diterjemahkan, kemudian melakukan pemrosesan terhadap sinyal suara tersebut sebelum diserahkan kepada model kecerdasan buatan yang digunakan, yaitu model wav2vec 2.0 XLSR-53. Setelah dilakukan pemrosesan sinyal suara dan pengolahan oleh model, maka model memberikan keluaran berupa teks hasil penerjemahan dari sinyal suara tersebut.



Gambar 3.2. Skema Model Pengenalan Suara

Gambar 3.2 mengilustrasikan skema model pengenalan suara dengan menggunakan model wav2vec 2.0 yang telah dirancang. Pertama, sinyal suara diproses sebelum digunakan sebagai masukan pada model wav2vec 2.0. Varian dari model wav2vec 2.0 yang digunakan berupa wav2vec 2.0 XLSR-53 yang memiliki arsitektur yang sama dengan wav2vec 2.0 Large, namun menggunakan data suara dari berbagai bahasa dalam proses *pretraining* seperti yang telah dijelaskan pada sub bab 2.3. Setelah dilakukan pemrosesan awal, sinyal suara diberikan kepada model untuk mengklasifikasikan huruf-huruf berdasarkan bunyi yang terdapat pada sinyal suara, sehingga membentuk suatu kata. Hasil keluaran akhir dari model wav2vec 2.0 XLSR-53 berupa hasil penerjemahan dari sinyal suara tersebut dalam bentuk teks, atau kumpulan kata-kata dari huruf-huruf yang diklasifikasikan.

Sebagai sistem pengenal suara, model wav2vec terbagi menjadi dapat dibagi menjadi beberapa komponen, seperti yang digambarkan pada Gambar 3.3.



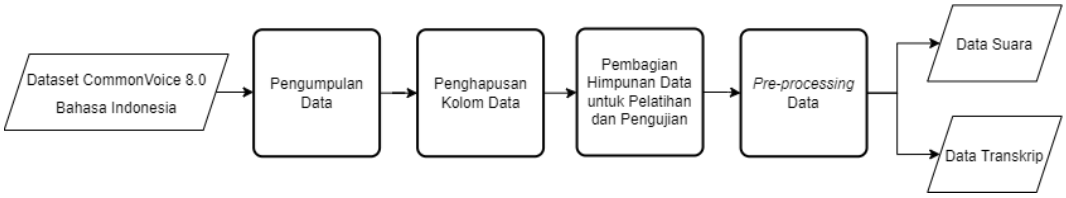
Gambar 3.3. Skema Komponen Model Wav2Vec 2.0

Gambar 3.3 mengilustrasikan masing-masing komponen dalam model wav2vec 2.0, yang terdiri dari sebuah wav2vec 2.0 *Processor* yang berfungsi untuk mengolah data masukan sinyal suara dari model kemudian memberikan nilai keluaran berupa teks hasil prediksi pengenalan suara. Komponen wav2vec 2.0 *Processor* terdiri dari 2 sub-komponen yaitu wav2vec 2.0 *Feature Extractor* yang berfungsi untuk mengubah sinyal suara menjadi nilai vektor, dan wav2vec 2.0 *Tokenizer* yang mengubah nilai vektor tersebut menjadi huruf atau bunyi yang membentuk sebuah kata. Dalam implementasi model wav2vec 2.0 yang telah dirancang, digunakan juga sebuah *Data Collator* yang berfungsi untuk melakukan *padding* atau penambahan pada data sehingga semua data suara dalam sebuah *batch* memiliki durasi yang serupa.

3.3. Persiapan Data

Sebelum model diberikan data berupa sinyal suara beserta transkripnya, dilakukan beberapa tahapan yang dilakukan untuk mempersiapkan data tersebut. Pertama, data dikumpulkan dari *dataset* CommonVoice 8.0 Bahasa Indonesia. *Dataset* tersebut berisi sinyal suara percakapan dalam Bahasa Indonesia dari berbagai pembicara beserta dengan transkrip dari percakapan tersebut. Selain sinyal suara dan transkrip, pada data CommonVoice terdapat juga informasi mengenai pembicara untuk setiap pasangan suara dan transkrip, yang tidak diperlukan untuk melatih model ASR. Oleh karena itu, dilakukan penghapusan data tersebut pada tahap selanjutnya. Setelah itu, tahapan selanjutnya adalah melakukan pembagian

himpunan data untuk pelatihan dan pengujian model, serta melakukan *pre-processing* sehingga bentuk data sesuai dengan kebutuhan model. Hasil dari persiapan data berupa data suara dan transkrip yang dibutuhkan model untuk melakukan pelatihan dan pengujian. Proses yang dilakukan divisualisasikan pada Gambar 3.4.



Gambar 3.4. Langkah Persiapan Data

Gambar 3.4 mengilustrasikan langkah persiapan data yang dilakukan sebelum data tersebut digunakan dalam proses pelatihan dan pengujian model. Terdapat dua hasil yang digunakan sebagai masukan pada model. Pertama, data yang diperlukan untuk melakukan pelatihan, pengujian, dan percobaan model pengenalan suara otomatis berbasis wav2vec berupa sinyal suara yang dapat berasal dari berkas audio seperti format MP3 atau WAV. Untuk mendapatkan hasil yang optimal, maka sinyal suara yang digunakan berupa potongan-potongan percakapan dengan durasi sekitar 5-30 detik. Sinyal suara yang sudah disiapkan digunakan sebagai masukan bagi model yang memprediksi hasil teks berisi kata-kata yang diucapkan dalam percakapan tersebut.

Dalam melakukan pelatihan model, dibutuhkan juga label atau transkrip percakapan dari data yang digunakan. Hal ini dilakukan karena untuk membentuk sebuah model yang akurat maka dibutuhkan juga penyesuaian nilai-nilai bobot yang terdapat pada model tersebut, berdasarkan perbedaan hasil prediksi keluaran dari model dengan keluaran asli atau transkrip percakapan yang telah disediakan. Oleh karena itu, untuk menjalankan pelatihan dari model yang sudah dirancang, maka dibutuhkan berkas audio beserta dengan transkrip dari masing-masing berkas audio yang mencatat teks dari percakapan asli yang telah diutarakan.

3.3.1. Pengumpulan Data

Untuk dapat mengakses *dataset* CommonVoice dalam percobaan, digunakan *library* Datasets yang dikeluarkan oleh tim HuggingFace [28]. *Library* ini memudahkan pengambilan data dari beberapa *dataset* yang sering digunakan seperti CommonVoice. Pada *library* Datasets, data yang sudah diunduh melalui Internet kemudian disimpan dalam sebuah folder pada perangkat komputer, sehingga tidak perlu dilakukan pengunduhan ulang apabila ingin menggunakan *dataset* yang sama, meskipun pada program yang berbeda. Untuk dapat mulai menyimpan sebuah himpunan data dan kemudian menggunakannya, maka digunakan fungsi load_dataset yang disediakan oleh *library* ini seperti yang dilakukan pada Gambar 3.5.

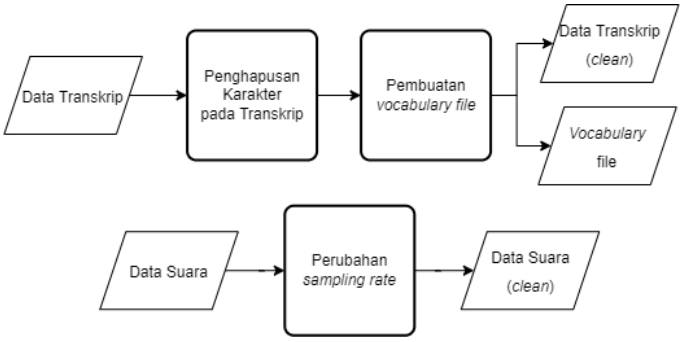
```
train_dataset = load_dataset("common_voice", "id", split="train+validation")

eval_dataset = load_dataset("common_voice", "id", split="test")
```

Gambar 3.5. Potongan Kode Untuk Mengumpulkan Data Dari *Library* Datasets

Gambar 3.5 menunjukkan potongan kode yang berfungsi untuk menyimpan *dataset* Common Voice Bahasa Indonesia melalui fungsi load_dataset dari *library* Datasets. Fungsi load_dataset juga dapat digunakan untuk membagi himpunan data untuk pelatihan dan pengujian model, melalui parameter split yang menentukan sub himpunan dari *dataset* Common Voice yang disertakan dalam *library* datasets.

3.3.2. Pre-processing Data

Sebelum data dapat sinyal suara dan transkrip yang telah disiapkan dapat digunakan untuk model, maka dilakukan *pre-processing* terlebih dahulu sehingga bentuk dari kedua data tersebut sesuai dengan kebutuhan model. Terdapat tahapan *pre-processing* yang berbeda dilakukan pada data transkrip dengan data sinyal suara. Tahapan yang dilakukan diilustrasikan pada Gambar 3.6.



Gambar 3.6. Tahapan *Pre-processing* Pada Data

Gambar 3.6 mengilustrasikan tahap *pre-processing* data untuk model wav2vec 2.0 digunakan. Untuk data transkrip yang berbentuk teks berupa penghapusan karakter atau simbol yang tidak digunakan beserta dengan pembuatan *vocabulary file*. Untuk data suara, dilakukan perubahan *sampling rate* untuk mengurangi dimensi dari data yang digunakan.

```
def remove_special_characters(batch):
    batch["sentence"] = re.sub(chars_to_ignore_regex, '',
                               batch["sentence"]).lower() + " "
    return batch
```

Gambar 3.7. Potongan Kode Untuk Menghilangkan Karakter Yang Tidak Diinginkan Pada Data Transkrip

Potongan kode pada Gambar 3.7 menunjukkan penghilangan simbol-simbol selain huruf dan karakter spasi dari transkrip data suara yang digunakan. Hal tersebut dapat dicapai dengan menghilangkan karakter-karakter yang tidak diinginkan menggunakan *Regular Expression* (Regex), yang mencari dan mengganti karakter tersebut dengan karakter spasi pada masing-masing transkrip dalam himpunan data. Beberapa contoh dari karakter yang dihilangkan dengan Regex berupa simbol tanda baca serta huruf yang memiliki tanda aksen.

Pada tahap *pre-processing* data, diterapkan juga *feature extraction* yang berfungsi untuk membentuk sebuah *Vocabulary File* dari semua data transkrip yang digunakan.

Gambar 3.8. Contoh Isi Dari *Vocabulary File*

Gambar 3.8 menunjukkan contoh dari sebuah *Vocabulary File* yang dapat digunakan dalam proses *decoding* oleh model wav2vec 2.0. Berkas ini berisi masing-masing karakter huruf yang dapat ditemukan pada data transkrip setelah dilakukan penghilangan karakter pada proses sebelumnya. Selain karakter huruf, terdapat juga beberapa *directive* atau acuan yang digunakan oleh model untuk mengenali konteks dari sebuah suara percakapan. *Directive* yang digunakan berupa tanda "|" yang menggantikan karakter spasi sehingga lebih mudah untuk dikenali, tanda "UNK", atau tanda *unknown*, yang diberikan untuk karakter dari suara yang tidak dikenal, serta tanda "PAD" yang melambangkan keadaan saat suara berhenti dalam sebuah percakapan, seperti setelah mengucapkan sebuah kata. *Vocabulary File* yang telah dibentuk kemudian disimpan dalam komputer, sehingga dapat digunakan oleh model baik dalam tahap pelatihan, pengujian, maupun penggunaan. Urutan masing-masing huruf dalam *Vocabulary File* tidak harus sesuai dengan abjad dan dibaca oleh *tokenizer model* secara sama.

```
train_dataset = train_dataset.cast_column("audio", Audio(sampling_rate=16_000))

eval_dataset = eval_dataset.cast_column("audio", Audio(sampling_rate=16_000))
```

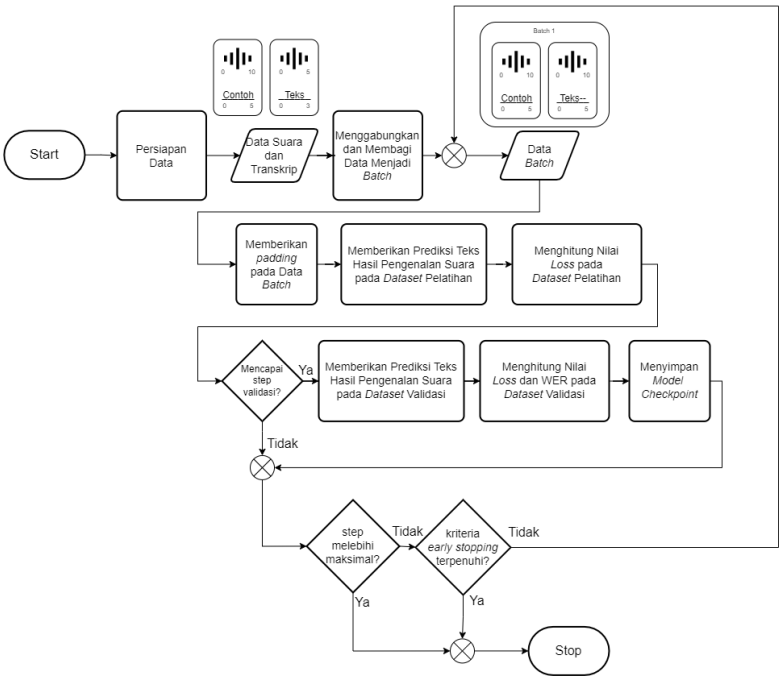
Gambar 3.9 Potongan Kode Untuk Mengubah *Sampling Rate* Dari Data

Gambar 3.9. menunjukkan proses perubahan *sampling rate* menggunakan fungsi cast_column dari *library* Datasets. Perubahan *sampling rate* merupakan satu-satunya perubahan yang dilakukan pada data sinyal suara, yang dilakukan untuk mereduksi dimensi dari sinyal tersebut sehingga lebih mudah dan lebih cepat diproses oleh model. Pada *dataset* CommonVoice 8.0 Bahasa Indonesia yang

menggunakan 48 kHz untuk masing-masing sinyal suara, dilakukan perubahan menjadi 16 kHz atau hanya menggunakan satu dari setiap tiga nilai berurutan pada sinyal suara awal.

3.4. Rancangan Proses Pelatihan Model

Setelah dilakukan persiapan data dan melalui tahapan seperti pengumpulan serta *pre-processing* data, maka data dihasilkan data suara yang transkrip dengan bentuk dan tipe yang sesuai dengan model. Data tersebut sudah siap digunakan dalam proses pelatihan. Proses pelatihan dari model secara garis besar berupa penggabungan data menjadi *batch*, kemudian melakukan prediksi teks transkrip dari sinyal suara pada setiap *batch* dan menghitung *loss* atau perbedaan dari teks transkrip yang diprediksikan oleh model dengan teks transkrip asli untuk masing-masing sinyal suara. Setelah itu, proses validasi model dilakukan setiap sejumlah *step* atau jumlah data pelatihan tertentu dilalui, untuk menentukan model terbaik yang disimpan. Kedua proses tersebut, kemudian dilakukan secara berulang seperti yang diilustrasikan pada Gambar 3.10 selama model belum mencapai *step* maksimal atau kriteria *early stopping* belum terpenuhi.



Gambar 3.10. Skema Proses Pelatihan Model wav2vec 2.0

Gambar 3.10 menunjukkan proses pelatihan model wav2vec 2.0 yang dilakukan secara berulang terhadap data yang diberikan, dimulai dari penggabungan dan pembagian data menjadi *batch* sesuai parameter yang ditentukan untuk pelatihan tersebut. Ukuran *batch* yang lebih tinggi dapat mempercepat proses pelatihan, namun dapat menambah ukuran dari model wav2vec 2.0 yang harus dimuat kepada GPU, sehingga menggunakan RAM pada GPU yang berlebih. Pada setiap *batch*, dilakukan juga penambahan *padding* untuk menyamakan durasi masing-masing berkas suara, sehingga satu sampel data tidak terlalu banyak mempengaruhi model dalam satu kali perulangan. Setelah model selesai melakukan pelatihan terhadap satu *batch*, maka model menandai bahwa satu perulangan atau *step* telah berakhir. Kemudian, proses pelatihan dimulai kembali dari awal dengan *batch* lain dalam *step* berikutnya.

Proses pelatihan model wav2vec 2.0 yang dilakukan terdiri dari dua fase, yakni fase pelatihan pertama dan fase validasi, yang berhubungan dengan *dataset* yang digunakan pada kedua fase tersebut. Pada fase pelatihan pertama digunakan *dataset* Common Voice Bahasa Indonesia gabungan bagian *train* dan *dev* (30% dari total data). Pada fase ini, dilakukan pemrosesan data oleh model yang memberikan prediksi hasil penerjemahan suara percakapan menjadi teks, kemudian menghitung nilai *loss* yang dihasilkan oleh prediksi tersebut dengan transkrip asli suara percakapan. Fase ini berfungsi untuk mencari nilai bobot yang paling tepat bagi model dalam melakukan pengenalan suara dan prediksi menjadi bentuk teks. Perubahan nilai bobot yang dilakukan berdasar kepada *learning rate* yang telah ditentukan sebelumnya. *Learning rate* yang terlalu besar membuat bobot dari model wav2vec 2.0 menjadi mudah berganti, sehingga tidak menemukan titik bobot optimum yang dapat menghasilkan akurasi tertinggi. Sebaliknya, *learning rate* yang terlalu kecil membuat proses pembelajaran model menjadi membutuhkan waktu yang sangat lama karena membuat perubahan kepada bobot menjadi sangat kecil setiap iterasi.

Fase kedua dari proses pelatihan model wav2vec 2.0, fase validasi, yang tidak dijalankan pada setiap *step*, namun hanya dilakukan setiap sejumlah *step* tertentu. Fase ini menggunakan bagian *dataset* Common Voice Bahasa Indonesia

yang berbeda, yakni bagian *test* (14% dari total data). Fase ini ditujukan untuk melakukan pengecekan terhadap bobot yang sudah ditentukan pada fase pelatihan pertama terhadap himpunan data lain yang tidak digunakan sebelumnya, dengan melakukan perhitungan *loss* terhadap *dataset test* pada Common Voice. Pada fase validasi juga dilakukan perhitungan WER secara keseluruhan yang menggambarkan performa model wav2vec 2.0 saat melakukan prediksi teks hasil pengenalan suara. Jika performa model melebihi performa pada fase validasi yang telah dilakukan sebelumnya, maka model disimpan dalam komputer sehingga dapat digunakan untuk sistem ASR.

Proses pelatihan model wav2vec 2.0 terus melakukan perulangan selama terdapat dua kondisi yang masih terpenuhi. Pertama, terdapat maksimal jumlah *step* yang ditentukan dan diterapkan selama pelatihan berlangsung. Model secara langsung mengakhiri pelatihan ketika jumlah *step* melebihi nilai maksimal tersebut. Kedua, pada rancangan pelatihan model yang telah dibuat diimplementasikan *Early Stopping* yang dapat memberhentikan proses pelatihan meskipun belum mencapai maksimal jumlah *step* yang ditentukan. *Early Stopping* digunakan agar tidak terjadi *overfitting* pada model yang telah dilatih di mana model memiliki performa yang baik pada pelatihan namun tidak pada data lainnya. Hal ini disebabkan oleh proses pelatihan yang terlalu lama. *Early Stopping* memberhentikan model ketika terjadi penurunan *loss* yang kecil setelah sejumlah *step* tertentu, yang menandakan proses pelatihan sudah cukup dilakukan.

3.5. Rancangan Proses Pengujian Model

Skenario pengujian model wav2vec 2.0 yang direncanakan berupa memuat model pada GPU untuk digunakan, kemudian melakukan prediksi terhadap data Common Voice Bahasa Indonesia bagian *test*. Dari hasil prediksi yang dilakukan dihasilkan vektor nilai suara yang diterjemahkan menjadi bentuk teks, serupa dengan transkrip asli dari dataset. Masing-masing hasil prediksi teks tersebut kemudian dibandingkan dengan transkrip asli dari dataset. Terakhir, dikalkulasikan WER secara keseluruhan berdasarkan perbandingan masing-masing teks transkrip pada *dataset* dengan hasil dari model wav2vec 2.0, yang menggambarkan performa keseluruhan dari model tersebut.

BAB 4
IMPLEMENTASI DAN ANALISIS SISTEM AUTOMATIC SPEECH RECOGNITION

4.1. Implementasi Sistem

Sistem yang dioptimalkan dan diuji merupakan model wav2vec 2.0 XLSR-53 yang telah dilatih lebih lanjut melalui proses *fine-tuning* dengan sejumlah data dari *dataset* Common Voice 8.0. Pertama, model wav2vec 2.0 dibandingkan dengan model Deep Neural Network (DNN) lainnya untuk mendapatkan perspektif mengenai performa model tersebut dalam mengenali suara Bahasa Indonesia. Pada model ini, berdasarkan spesifikasi perangkat keras yang telah diberikan, digunakan *batch size* sebesar 4 sebagai nilai terbesar yang dapat diolah oleh GPU.

Sebagai pembanding, digunakan model dengan arsitektur DNN yang sering digunakan dalam permasalahan ASR [29], berupa jaringan Bidirectional Long Short-Term Memory (LSTM) yang menggunakan jaringan untuk ekstraksi fitur dengan CNN. Model ini tidak mengolah sinyal suara secara langsung layaknya wav2vec 2.0, melainkan menggunakan transformasi Mel Frequency Cepstral Coefficients (MFCC) untuk mendapatkan 40 vektor fitur utama yang menggambarkan suatu potongan suara 25 ms. Arsitektur dari model ini terdiri dari 3 lapisan konvolusi, masing-masing dengan dimensi keluaran 32, ukuran kernel (3, 3, 3), *stride* (3, 1, 1) secara berurut pada CNN untuk melakukan ekstraksi fitur lebih lanjut dari hasil MFCC. Kemudian, hasil representasi dari CNN diolah oleh 2 lapisan LSTM dengan 512 *hidden neuron*, sebelum dipetakan kepada 28 kemungkinan label berupa 26 huruf, satu karakter spasi, dan satu karakter untuk melambangkan kekosongan (*padding*).

Tabel 4.1. Spesifikasi Komputer

Tipe	Jenis	Nama/Versi
Hardware	*Processor* (CPU)	Intel Core i7-8700 @ 3.20 GHz (12 CPUs)
	Graphics Card (GPU)	NVIDIA GeForce RTX 2080 SUPER (8GB)
	RAM	32 GB DDR 4
	Storage	2 TB HDD
Software	*Operating System*	Ubuntu 20.04 LTS Linux
	Conda	4.11.0
	Python	Python 3.9.12
	PyTorch	1.11.0
	CUDA *Toolkit*	11.3
	Automatic Speech Recognition (ASR) Model	wav2vec 2.0 XLSR-53
	Integrated Development Environment (IDE)	Jupyter Notebook 6.4.11

Untuk membangun model kecerdasan buatan yang dilatih, diuji, dan dijalankan, dibutuhkan sebuah perangkat keras yaitu sebuah komputer. Spesifikasi dari komputer yang digunakan dicantumkan pada Tabel 4.1. Spesifikasi Komputer Spesifikasi yang digunakan tidak semuanya berpengaruh kepada hasil dan akurasi model. Namun, spesifikasi GPU sangat berpengaruh kepada kecepatan model dalam proses pelatihan dan pengujian, di mana GPU dengan VRAM yang besar dapat meningkatkan jumlah *batch size* yang diproses oleh model dalam satu waktu. Selain itu, jumlah CUDA Core dalam GPU juga berpengaruh kepada waktu yang dibutuhkan untuk memproses satu sampel.

4.2. Skenario Pengujian

Terdapat dua skenario pengujian yang dilakukan bagi model wav2vec 2.0, pertama yakni pengujian performa keseluruhan dari model wav2vec 2.0 XLSR-53 dan perbandingannya dengan model CNN-BiLSTM serupa dengan yang digunakan pada [29]. Pada proses pelatihan kedua model, jumlah data yang digunakan sama, sehingga keduanya dapat dibandingkan dalam konteks yang sama pula. Setelah dilakukan perbandingan kedua model, maka dapat dilakukan percobaan terkait internal model wav2vec 2.0 yang membandingkan berbagai konfigurasi model wav2vec 2.0 yang berbeda untuk mendapatkan hasil yang terbaik dari model tersebut berdasarkan konteks data suara Bahasa Indonesia. Pada semua pengujian

yang dilakukan, *dataset* yang digunakan berupa *dataset* Common Voice 8.0 dengan himpunan bagian *test*. Selain itu, semua pengujian yang dilakukan diulang sebanyak lima kali dan diambil rata-rata beserta standar deviasi dari kelima percobaan, sehingga menggambarkan hasil yang lebih konsisten.

4.2.1. Skenario Pengujian Performa Model wav2vec 2.0

Terdapat dua kriteria utama yang digunakan untuk melakukan pengujian dan perbandingan model wav2vec 2.0 XLSR-53 dengan model CNN-BiLSTM. Kriteria pertama adalah WER yang menggambarkan performa model ketika melakukan pengenalan dan pembuatan teks terhadap sinyal suara. WER dari model perlu diuji karena untuk membuat sistem pengenalan suara yang baik, tentunya diperlukan juga akurasi perubahan *speech-to-text* yang tinggi pula, salah satunya digambarkan oleh nilai WER yang rendah. Dalam pengujian kriteria ini, perlu diperhatikan juga pengaruh jumlah potongan suara yang digunakan dalam pelatihan model, karena mempengaruhi kemampuan model tersebut dalam mempelajari bahasa *low-resource*, seperti Bahasa Indonesia.

Kriteria kedua yang diuji merupakan ukuran dari model saat dimuat ke dalam GPU, atau seberapa besar memori pada GPU yang dialokasikan untuk menyiapkan dan menjalankan kedua model beserta dengan ukuran berkas dari model saat disimpan. Model yang membutuhkan alokasi memori lebih besar menyebabkan spesifikasi minimum perangkat komputer dalam rancang bangun sistem menjadi lebih tinggi, terutama pada komponen GPU. Hal ini mengakibatkan biaya yang diperlukan untuk menjalankan sistem pengenalan suara secara luring menjadi naik dan lebih sulit untuk diimplementasikan. Oleh karena itu, dilakukan pengujian terhadap penggunaan memori GPU dalam mengukur performa kedua model.

4.2.2. Skenario Pengujian Internal Model wav2vec 2.0

Pada skenario pengujian internal model wav2vec 2.0, kriteria yang diuji berupa penggunaan *learning rate* pada model serta jumlah data dalam proses pelatihan. Pengujian terhadap kriteria *learning rate* dilakukan karena pengaturan

learning rate merupakan proses yang cukup penting dalam membangun model yang tidak hanya mempelajari data yang diberikan, namun dapat melakukannya dalam waktu yang cukup singkat. Dalam pengujian ini dilakukan percobaan dengan nilai *learning rate* $1 \times 10^{-3}, 1 \times 10^{-4}$, dan 1×10^{-5}. Setelah ditemukan skala yang tepat bagi *learning rate*, maka dapat dioptimalkan lebih lanjut berdasarkan satuan utama. Satuan yang diuji adalah 1, 3, dan 5. Selain *learning rate*, dilakukan juga pengujian terhadap pengaruh variasi jumlah data pelatihan untuk model wav2vec 2.0, yang dilakukan untuk menentukan apakah performa dari model tersebut masih dapat ditingkatkan seiring dengan jumlah data pelatihan yang ditambahkan. Pada percobaan ini, jumlah data suara percakapan yang digunakan dalam pelatihan berjumlah sekitar 8000, 16000, dan 32000 sampel. Hal yang diuji pada percobaan tersebut adalah nilai WER dari ketiga variasi jumlah sampel data pelatihan ketika dilakukan penerjemahan oleh model tanpa *language model* (LM) dan dengan menggunakan LM.

4.3. Pengujian dan Perbandingan Performa Model wav2vec 2.0

Berdasarkan skenario yang telah dijelaskan sebelumnya mengenai pengujian dan perbandingan model wav2vec 2.0, berikut adalah hasil yang didapatkan:

4.3.1. Pengujian 1: Pengujian Model tanpa Language Model

Pada pengujian ini, model wav2vec 2.0 dibandingkan dengan model CNN-BiLSTM sebagai *baseline*. Model wav2vec 2.0 menggunakan *hyperparameter* yang sudah ditetapkan yaitu *learning rate* sebesar 10^{-4} dan *batch size* 4. Dalam pelatihan, model wav2vec 2.0 menggunakan data dengan label sebanyak 8.000 potongan suara yang didapatkan dari bagian *train* dan *dev*. Sebagai pembanding, model CNN-BiLSTM juga diuji dengan sejumlah potongan suara tersebut, beserta dengan 50.000 potongan suara total yang ada pada Common Voice. Hal ini ditujukan untuk membandingkan kemampuan model wav2vec 2.0 untuk mengenali sebuah *low-resource language* saat dibandingkan dengan model lain yang pada umumnya membutuhkan jumlah data yang jauh lebih banyak. Pengujian berupa proses pelatihan dengan 50.000 sampel data tidak dilakukan pada model wav2vec

2.0 dikarenakan keterbatasan pada Video RAM yang ada di GPU, yang tidak dapat mengakomodasi kebutuhan memori model wav2vec 2.0 yang tinggi. Pada pengujian kali ini, kedua model tidak menggunakan *language model*. Berikut adalah hasil yang ditemukan:

Tabel 4.2. Hasil WER Model wav2vec 2.0 XLSR-53 Dengan Model CNN-BiLSTM.

Model	**Jumlah Data dengan Label**	**WER**
CNN-BiLSTM	8k	$92{,}72\% \pm 3{,}67$
	50k	$50{,}05\% \pm 2{,}03$
wav2vec 2.0 XLSR-53	8k	$\mathbf{25{,}65\% \pm 1{,}6}$
	50k	-

Pada Tabel 4.2, dapat diamati bahwa model wav2vec 2.0 XLSR-53 memiliki performa yang jauh lebih baik dibandingkan pendekatan CNN-BiLSTM yang banyak digunakan sebelumnya. Model wav2vec 2.0 mampu mendapatkan WER sebesar 25,65%, dengan selisih 30% dari model CNN-BiLSTM namun dengan jumlah data yang lebih sedikit. Jika dibandingkan pada jumlah data yang sama, maka model wav2vec 2.0 jauh lebih baik dibandingkan model CNN-BiLSTM yang mendapat WER 93%. Nilai WER tersebut terlalu tinggi, sehingga tidak dapat diintegrasikan dalam sistem pengenalan suara otomatis jika hanya diberikan 8.000 data suara, seperti yang dilakukan pada pengujian.

Kemampuan wav2vec 2.0 untuk mengenali suara meskipun hanya diberikan data dengan label yang sangat minim disebabkan oleh adanya *pre-training* dalam proses *self-supervised learning* pada model wav2vec 2.0 yang membuat model tersebut mempelajari representasi umum dari sebuah percakapan sebelum dilakukan pelatihan melalui *fine-tuning.* Dengan mempelajari representasi umum tersebut, maka model hanya perlu menyesuaikan parameter yang sudah ditetapkan sebelumnya dengan data yang diberikan. Berbalikan dengan metode ini, model tradisional seperti CNN-BiLSTM harus dilatih ulang dari awal, ataupun dilatih melalui proses *transfer learning*, keduanya yang membutuhkan jumlah data dengan label yang cukup banyak.

Meskipun itu, Tabel 4.3 menunjukkan bahwa terdapat kelemahan dari model wav2vec 2.0 terdapat pada sumber daya komputasi yang dibutuhkan olehnya,

yang mengakibatkan *dataset* dengan 50.000 data tidak dapat dijalankan pada spesifikasi perangkat keras yang digunakan. Untuk mengilustrasikan hal tersebut, dilakukan pengujian terhadap dua kriteria, yaitu penggunaan VRAM pada model dalam GPU sebelum dan saat *training*, beserta dengan penggunaan *storage* model tersebut saat disimpan sebagai *file* PyTorch.

Tabel 4.3. Penggunaan Sumber Daya Komputasi Model wav2vec 2.0 XLSR-53 Dan Model CNN-BiLSTM.

Model	**Penggunaan VRAM (MB)**		**Penggunaan Storage (MB)**
	Sebelum training	**Saat training**	
CNN-BiLSTM	**1.203**	**2.723**	**136**
wav2vec 2.0 XLSR-53	2.097	7.643	1.300

Model wav2vec 2.0 XLSR-53 yang membutuhkan VRAM dan *storage* membuatnya menjadi lebih sulit untuk dilakukan pelatihan dan *deployment* pada perangkat keras dengan sumber daya komputasi dan penyimpanan yang minim. Sebelum dan saat *training*, model wav2vec 2.0 menggunakan 2 kali lipat VRAM dibandingkan dengan model CNN-BiLSTM. Selain itu, model wav2vec 2.0 juga membutuhkan *storage* yang jauh lebih besar. Kedua hal ini disebabkan oleh arsitektur wav2vec 2.0 yang lebih kompleks, dengan jumlah layer dan dimensi yang lebih banyak, sehingga memiliki parameter seperti bobot, bias, dan lainnya, dengan jumlah besar. Penggunaan model Transformer yang terdapat pada wav2vec 2.0 juga pada umumnya membutuhkan sumber daya komputasi lebih tinggi dibandingkan dengan model *autoregressive* dalam pengolahan masukan rangkaian waktu, karena memanfaatkan kemampuan paralelisasi pada GPU secara lebih maksimal.

4.3.2. Pengujian 2: Pengujian Model dengan Language Model

Selain menguji dan membandingkan WER model wav2vec 2.0 dengan model CNN-BiLSTM, dilakukan juga perbandingan performa model saat ditambah dengan sebuah *language model* (LM). *Language model* dapat digunakan untuk menambah performa sebuah *acoustic model* seperti model DNN, dengan melakukan komputasi kata-kata mana yang dapat bersebelahan dalam sebuah kalimat serta memperbaiki semua kesalahan penulisan huruf yang dibentuk dalam proses inferensi model. Adapun *language model* yang digunakan berupa 2-gram

language model yang disusun dari semua transkrip pada *dataset* Common Voice 8.0 Bahasa Indonesia dengan menggunakan *software* KenLM. Meskipun penggunaan *dataset* yang sama untuk pelatihan dan *language model* dapat membuat hasil WER lebih baik karena sudah sesuai dengan data yang digunakan dalam proses inferensi, *language model* ini tetap dapat digunakan sebagai contoh penambahan LM untuk meningkatkan performa model.

Tabel 4.4. Hasil WER Model wav2vec 2.0 XLSR-53 Dengan Model CNN-BiLSTM Saat Menggunakan *Language Model*

Model	**Jumlah Data dengan Label**	**WER**
CNN-BiLSTM	8k	$63{,}8\% \pm 1{,}67$
	50k	$21{,}66\% \pm 0{,}43$
wav2vec 2.0 XLSR-53	8k	$\mathbf{13{,}64\% \pm 0{,}5}$
	50k	-

Dapat dilihat dari hasil pada Tabel 4.4, yang menunjukkan bahwa penggunaan *language model* mengurangi nilai WER pada kedua model, berapa pun jumlah data yang digunakan. Selain itu, *language model* juga mengurangi standar deviasi dari kedua model, karena membuat hasil inferensi yang dilakukan oleh model menjadi lebih konsisten. Hal ini disebabkan oleh LM yang membentuk probabilitas yang tidak berubah untuk menggambarkan hubungan kata, meskipun nilai parameter dari model sedikit berubah. Meskipun itu, model wav2vec 2.0 tetap memiliki performa yang lebih baik dibandingkan dengan model CNN-BiLSTM yang menghasilkan WER sebesar 13,64% dibandingkan dengan 21,66% secara berurutan, meskipun hanya menggunakan data dengan label yang jauh lebih sedikit.

4.4. Pengujian Internal Model wav2vec 2.0

Berdasarkan skenario yang telah dijelaskan sebelumnya mengenai pengujian internal model wav2vec 2.0, berikut adalah hasil yang didapatkan:

4.4.1. Pengujian 3: Pengujian Variasi Learning Rate

Pada pengujian ini, dilakukan pengujian terhadap variasi nilai *learning rate* yang digunakan dalam proses pembelajaran model wav2vec 2.0 XLSR-53.

Tabel 4.5. Hasil WER Model wav2vec 2.0 XLSR-53 Dengan Variasi Skala *Learning Rate*.

Learning Rate	WER
1×10^{-3}	$85{,}05\% \pm 8$
1×10^{-4}	$\mathbf{40{,}2\% \pm 0{,}8}$
1×10^{-5}	$98{,}55\% \pm 2{,}2$

Dari hasil pengujian yang didapatkan pada Tabel 4.5, diperoleh WER terbaik pada model wav2vec 2.0 XLSR-53 yaitu pada saat *learning rate* bernilai 10^{-4}, yang memberikan nilai WER 40,2% hanya dalam 10 epoch. Jika dibandingkan dengan skala lainnya, maka dapat dilihat bahwa pada skala 10^{-3} terdapat pembelajaran namun nilai *learning rate* tersebut terlalu besar mengakibatkan model tidak dapat menemukan titik yang mendekati *global optimum* sehingga memiliki performa yang kurang baik. Hal tersebut juga dicerminkan pada nilai standar deviasi yang tinggi di antara semua percobaan yang dilakukan. Selain itu, pada skala 10^{-5} pembelajaran berlangsung secara sangat lambat, sehingga model kurang mengalami perubahan pada bobotnya dan hanya dapat menghasilkan WER sebesar 98,55% pada saat pengujian. Oleh karena itu, didapatkan skala 10^{-4} yang dapat melatih model paling baik dalam waktu yang paling singkat.

Setelah didapatkan skala tersebut, maka dapat dicari nilai *learning rate* spesifik yang paling cocok untuk digunakan.

Tabel 4.6. Hasil WER Model wav2vec 2.0 XLSR-53 Dengan Variasi Satuan *Learning Rate*s

Learning Rate	WER
1×10^{-4}	$\mathbf{40{,}2\% \pm 0{,}8}$
3×10^{-4}	$44{,}3\% \pm 0{,}66$
5×10^{-4}	$53{,}15\% \pm 0{,}48$

Dari hasil pada Tabel 4.6, ditemukan bahwa meningkatkan nilai *learning rate* tidak memberikan hasil yang lebih optimal. Hal tersebut disebabkan oleh nilai 3×10^{-4} dan 5×10^{-4} yang terlalu besar untuk digunakan pada saat model wav2vec 2.0 telah mengalami pembelajaran dalam waktu yang cukup lama. Saat itu, model hanya membutuhkan dorongan kecil untuk mencapai *global optimum*, namun kedua nilai yang lebih besar mengakibatkan model melakukan perubahan yang terlalu signifikan, sehingga menghasilkan WER yang lebih tinggi.

4.4.2. Pengujian 4: Pengujian Variasi Jumlah Data Pelatihan

Pengujian yang dilakukan pada subbab ini berupa pengujian variasi jumlah data pasangan sampel suara dan transkripnya yang digunakan dalam proses pelatihan model wav2vec 2.0 XLSR-53. Pada pengujian ini, dilakukan 2 percobaan, yaitu pengujian WER hasil penerjemahan suara menjadi teks tanpa *Language Model* (LM) dan model beserta *Language Model*. *Language Model* yang digunakan pada pengujian ini dibentuk dari semua transkrip pada *dataset* Common Voice 8.0 Bahasa Indonesia dengan menggunakan KenLM.

Tabel 4.7. Hasil WER Model wav2vec 2.0 XLSR-53 Dengan Variasi Jumlah Data Pelatihan

Jumlah Sampel Data Pelatihan	WER tanpa LM	WER dengan LM
8.239	$25{,}65\% \pm 1{,}6$	$13{,}64\% \pm 0{,}5$
16.533	$23{,}46\% \pm 0{,}18$	$11{,}13\% \pm 0{,}47$
33.066	$\mathbf{15{,}6\% \pm 0{,}47}$	$\mathbf{8{,}59\% \pm 0{,}52}$

Tabel 4.7 menunjukkan pengaruh dari variasi jumlah sampel data suara percakapan Bahasa Indonesian dalam proses pelatihan *fine-tuning* model wav2vec 2.0 XLSR-53. Dari hasil pada tabel tersebut, ditemukan bahwa jumlah sampel data pelatihan memiliki hubungan yang berbanding lurus dengan performa model, di mana semakin banyak data yang digunakan maka semakin rendah WER dari model tersebut. Di antara ketiga variasi jumlah data yang digunakan, tanpa menggunakan *language model*, dapat ditemukan terdapat peningkatan yang sedikit pada jumlah sampel ~8.000 dan ~16.000, yang bernilai 25,65% dan 23,46% secara berurutan. Penurunan WER yang lebih signifikan terdapat pada jumlah sampel ~32.000 yang memiliki nilai WER 15,6%, yang juga merupakan hasil WER terbaik di antara ketiga variasi jumlah sampel data pelatihan. Pola yang sama dapat terlihat pada hasil penerjemahan model dengan menggunakan LM, yang menunjukkan bahwa performa dari model wav2vec 2.0 XLSR-53 masih dapat ditingkatkan dengan menambahkan jumlah data pelatihan yang digunakan dalam proses *fine-tuning*.

4.5. Analisa Keseluruhan

Dari pengujian model wav2vec yang telah dilakukan, terdapat 2 skenario utama yang digunakan. Pertama, dilakukan pengujian berupa kemampuan inferensi

atau penerjemahan model wav2vec 2.0 dibandingkan dengan sebuah model CNN-BiLSTM dalam mengubah sinyal suara percakapan Bahasa Indonesia menjadi teks. Skenario kedua berupa pengujian yang dilakukan terhadap internal model wav2vec 2.0 XLSR-53, yang membandingkan model tersebut dengan dirinya sendiri namun dengan konfigurasi nilai *learning rate* dan jumlah data pelatihan yang berbeda.

Pada skenario pertama yang membandingkan performa model wav2vec 2.0 XLSR-53 dengan model CNN-BiLSTM, terdapat beberapa kriteria yang digunakan sebagai metrik pengujian, di antaranya WER, penggunaan memori, dan penggunaan VRAM. Pertama, dilakukan pengujian WER dari kedua model dengan ukuran *dataset* 8.000 dan 50.000 data, sehingga dapat mengetahui kemampuan keduanya saat diberikan bahasa dengan sumber daya yang sedikit dan banyak. Hasil yang didapatkan adalah model wav2vec 2.0 XLSR-53 yang dapat mendapatkan WER sebesar $25{,}65\% \pm 3{,}6$ dari *dataset* dengan 8.000 sampel. Nilai WER ini dua kali lipat lebih baik dibandingkan dengan model CNN-BiLSTM yang menggunakan banyak sampel dengan $50{,}05\% \pm 2{,}03$. Nilai tersebut membuktikan kemampuan model wav2vec 2.0 dalam mengenal sebuah *low-resource language*. Dilakukan juga pengujian kedua model saat dilengkapi dengan sebuah *language model*, untuk meningkatkan performa kedua model tersebut. Hasil yang didapatkan semua model mengalami penurunan WER, dengan model CNN-BiLSTM dengan LM menjadi menyaingi model wav2vec 2.0 tanpa LM. Selain itu, penambahan LM juga mengurangi standar deviasi dari perulangan yang dilakukan, sehingga membuat pengujian menjadi lebih stabil.

Selain WER, kriteria penggunaan sumber daya komputasi berupa VRAM dan *storage* juga diuji. Didapatkan hasil bahwa model wav2vec 2.0 membutuhkan sumber daya komputasi yang jauh lebih banyak dibandingkan model CNN-BiLSTM. Model wav2vec 2.0 membutuhkan sekitar 2 kali lipat VRAM sebelum dan saat dilakukan proses pelatihan, serta membutuhkan hampir 10 kali lipat *storage* untuk menyimpan parameter model. Hal ini disebabkan oleh kompleksitas model tersebut yang memiliki lebih banyak lapisan dan neuron di masing-masing lapisannya, sehingga memiliki lebih banyak parameter yang harus dioptimalkan dan disimpan.

Pada skenario kedua dilakukan pengujian terhadap *learning rate* dari model wav2vec 2.0, untuk mendapatkan performa model yang paling optimal dengan data yang digunakan. Pada pengujian ini, dilakukan pelatihan model sebanyak 10 *epoch* sehingga pelatihan dan pengujian *learning rate* tidak berlangsung terlalu lama. Variabel yang divariasikan pada pengujian ini berupa *learning rate* dengan skala 10^{-3}, 10^{-4}, dan 10^{-5}. Setelah ditemukan skala yang tepat, maka dicari satuan dari *learning rate* dengan nilai 1, 3, dan 5. Hasil dari pengujian ini menemukan bahwa *learning rate* 10^{-4} , karena menghasilkan Parameter WER yang paling rendah dengan waktu pelatihan yang paling singkat juga. ini membantu model untuk mencari *global optimum*, sehingga dapat memiliki performa yang lebih baik.

Selain itu, pada skenario kedua juga dilakukan pengujian variasi jumlah data pelatihan dalam proses *fine-tuning* model wav2vec 2.0 XLSR-53 terhadap performa model tersebut, baik dengan *language model* (LM) atau pun tanpa LM. Dari hasil yang didapatkan, ditemukan korelasi yang berbanding lurus antara jumlah sampel data suara yang digunakan dalam pelatihan dengan performa model wav2vec 2.0, atau menghasilkan nilai WER yang lebih rendah semakin banyak data yang digunakan. Ditemukan bahwa percobaan dengan jumlah data sekitar 32.000 sampel mendapatkan hasil WER terbaik bernilai 15,6% tanpa menggunakan LM dan 8,6% jika ditambahkan dengan LM dalam proses *decoding*.

4.6. Dampak Terhadap Lingkungan, Masyarakat, dan/atau Bidang Lain

Meninjau dari bidang ke lingkungan, kemasyarakatan, dan lainnya, terdapat beberapa dampak yang diberikan dengan adanya sistem pengenalan suara otomatis berbasis model wav2vec 2.0 yang telah diuji. Dampak yang paling utama dan yang merupakan latar belakang utama dikembangkannya sistem pengenalan suara tersebut adalah membantu mengamankan data masyarakat berbentuk audio yang biasa digunakan pada saat menggunakan sistem ASR yang menggunakan Internet. Hal tersebut terjadi karena pada sistem ASR yang menggunakan Internet (*online*) seperti pada produk-produk *voice assistant* Amazon Alexa dan Google Home, adanya pengiriman data suara dari pengguna sistem tersebut ke penyedia layanan sistem ASR. Dari data suara yang dikirimkan, terdapat banyak informasi mengenai pengguna yang dapat dikenali tanpa disadari oleh pengguna tersebut, seperti lokasi,

kontak dekat, ketertarikan pengguna, dan berbagai informasi pribadi lainnya [3]. Dengan menggunakan sistem ASR berbasis wav2vec 2.0 yang bekerja secara luring (*offline*), maka pengguna tetap dapat menerjemahkan percakapan yang disuarakan menjadi teks tanpa harus khawatir terhadap keamanan dari data tersebut.

Dari dampak dalam segi penelitian, sistem ASR yang telah dibentuk dapat digunakan sebagai referensi terutama bagi penerjemahan data suara percakapan menjadi teks bagi *low-resource language* selain Bahasa Indonesia. Model wav2vec 2.0 XLSR-53 yang digunakan sebagai basis dari sistem mudah untuk dilatih kembali dengan menjalankan proses *fine-tuning*, yang membutuhkan data suara percakapan dan transkrip dalam kuantitas yang jauh lebih sedikit dibandingkan dengan arsitektur model DNN lainnya, seperti CNN-BiLSTM. Hal ini memungkinkan sistem ASR berbasis wav2vec 2.0 untuk dikembangkan lebih lanjut menjadi produk-produk seperti *voice assistant* atau *voice typing* selain tidak membutuhkan koneksi Internet, juga dapat bekerja dalam bahasa yang tidak umum seperti Bahasa Inggris, contohnya Bahasa Sunda ataupun Bahasa Jawa.

Meskipun itu, terdapat juga beberapa kekurangan dari sistem ASR berbasis wav2vec 2.0 yang telah dibuat, di antaranya dampak terhadap kebutuhan komputasi dari perangkat keras yang menjalankan sistem tersebut. Kebutuhan komputasi seperti VRAM yang cukup besar (di atas 2 GB) dalam menggunakan model wav2vec 2.0 membuat sistem ini menjadi sulit digunakan dalam perangkat keras dengan spesifikasi rendah. Pengembangan lebih lanjut juga dapat dilakukan untuk mengatasi permasalahan ini dalam penelitian-penelitian selanjutnya.

BAB 5
PENUTUP

Berdasarkan pengembangan sistem pengenalan suara otomatis berbasis wav2vec 2.0 yang telah dilakukan, didapatkan kesimpulan dan saran sebagai berikut:

5.1. Kesimpulan

Dari proses penelitian yang dilakukan, didapatkan beberapa kesimpulan dari semua tahap perancangan dan pengujian sebagai berikut:

1. Model wav2vec 2.0 dapat digunakan dalam membentuk sebuah sistem pengenalan suara yang dapat bekerja secara luring namun tetap memiliki performa baik.
2. Agar dapat membentuk model wav2vec 2.0 yang paling optimal, maka telah dibangun skenario pengujian menggunakan himpunan data Common Voice 8.0, dengan parameter yang diuji berupa WER dan penggunaan sumber daya komputasi oleh model tersebut.
3. Model wav2vec 2.0 XLSR-53 menghasilkan WER sebesar 25,96%. Model ini memiliki performa 2 kali lebih baik dibandingkan dengan pendekatan *Bidirectional* LSTM tradisional, meskipun hanya diberikan data dengan label yang jauh lebih sedikit dalam pelatihan. Hal ini memodelkan pelatihan sistem pengenal suara dengan *low-resource language*, seperti Bahasa Indonesia.
4. Penambahan *language model* untuk melakukan inferensi pada sebuah *acoustic model* seperti model wav2vec 2.0 meningkatkan performa dari model tersebut dengan mengurangi hasil WER sebesar 5-15%.
5. Model wav2vec 2.0 XLSR-53 membutuhkan sumber daya komputasi yang lebih banyak dibandingkan pendekatan Bidirectional LSTM tradisional, berupa 2 kali lebih banyak RAM dan 10 kali lebih banyak memori.
6. Untuk mengoptimalkan model wav2vec 2.0 maka dapat dilakukan dengan mengatur nilai *learning rate*. Dari pengujian yang dilakukan, ditemukan

learning rate sebesar 10^{-4} memberikan hasil WER paling rendah dalam waktu pelatihan yang singkat.

7. Performa dari model wav2vec 2.0 XLSR-53 dapat ditingkatkan lebih lanjut dengan menambahkan jumlah data suara percakapan yang digunakan dalam pelatihan, karena jumlah data pelatihan memiliki hubungan yang berbanding lurus dengan performa model.

5.2. Saran

Dari proses penelitian yang dilakukan, didapatkan beberapa saran dari semua tahap perancangan dan pengujian yang diajukan.

1. Melakukan pengintegrasian model wav2vec 2.0 XLSR-53 yang telah dioptimalkan dalam sebuah sistem pengalan suara secara *real-time*, kemudian melakukan *deployment* sistem tersebut dalam perangkat lunak yang dapat digunakan.
2. Melakukan pengoptimalan *hyperparameter* lebih lanjut dari model wav2vec 2.0, sehingga dapat dicari kembali performa yang benar-benar paling optimal.
3. Melakukan *pre-training* model wav2vec 2.0 dari awal menggunakan audio suara Bahasa Indonesia yang tidak diberikan label, untuk mencari pengaruh dari *pre-training* jika dilakukan dengan Bahasa Indonesia.

DAFTAR ACUAN

[1] J. Li, *Recent Advances in End-to-End Automatic Speech Recognition,* 2020.

[2] S. Suyanto, A. Arifianto, A. Sirwan dan A. P. Rizaendra, "End-to-End Speech Recognition Models for a Low-Resourced Indonesian Language," dalam *8th International Conference on Information and Communication Technology (ICoICT),* Yogyakarta, 2020.

[3] G. Germanos, K. Dimitris, N. Kolokotronis dan N. Georgious, "Privacy Issues in Voice Assistant Ecosystems," dalam *2020 IEEE World Congress on Services (SERVICES),* Beijing, China, 2020.

[4] Verizon, "Data Breach Investigation Report (DBIR)," Verizon, 2022.

[5] A. Baevski, H. Zhou, M. Abdelrahman dan M. Auli, *wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations,* Arxiv, 2020.

[6] J. M. Levis dan R. Suvorov, "Automatic Speech Recognition," dalam *Encyclopedia of Applied Linguistics,* Blackwell, 2012.

[7] B. H. Juang dan L. R. Rabiner, "Automatic Speech Recognition – A Brief History of the Technology Development," 10 Agustus 2004. [Online]. Available: https://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf. [Diakses 19 Desember 2021].

[8] A. Dev, A. Jain dan S. Bhatt, "Acoustic Modeling in Speech Recognition: A Systematic Review," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 11, no. 4, p. , 17 Maret 2020.

[9] T. K. Vintsyuk, “Speech Discrimination by Dynamic Programming,” *Kibernetika,* vol. 4, no. 1, pp. 81-88, 1968.

[10] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh dan K. Shalaan, “Speech Recognition Using Deep Neural Networks: A Systematic Review,” *IEEE Access,* vol. 7, pp. 19143-19165, 2019.

[11] N. Jaitly, P. Nguyen, A. Senior dan V. Vanhoucke, “Application of Pretrained Deep Neural Networks to Large Vocabulary Speech Recognition,” [Online]. Available: http://www.cs.toronto.edu/~ndjaitly/jaitly-interspeech12.pdf. [Diakses 20 Desember 2021].

[12] A. C. Morris, V. Maier dan P. D. Green, “From WER and RIL to MER and WIL: improved evaluation measures for connected speech recognition,” dalam *INTERSPEECH 2004 - ICSLP, 8th International Conference on Spoken Language Processing,* Jeju Island, 2004.

[13] Y. LeCun, L. Bottou, Y. Bengio dan P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998.

[14] S. Albawi, T. A. Mohammed dan S. Al-Zawi, “Understanding of a Convolutional Neural Network,” dalam *2017 International Conference on Engineering and Technology (ICET),* Antalya, 2017.

[15] S. Russell dan P. Norvig, Artificial Intelligence A Modern Approach, Englewood Cliffs: Prentice Hall, 1995.

[16] J. Devlin, M.-W. Chang, K. Lee dan K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Arxiv, 2019.

[17] A. v. d. Oord, Y. Li dan O. Vinyals, Representation Learning with Contrastive Predictive Coding, Arxiv, 2019.

[18] A. Graves, S. Fernández, F. Gomez dan J. Schmidhuber, “Connectionist Temporal Classification : Labelling Unsegmented Sequence Data with Recurrent Neural Networks,” dalam *ICML '06: Proceedings of the 23rd international conference on Machine learning*, Pittsburgh Pennsylvania, USA, 2006.

[19] S. Schneider, A. Baevski, R. Collobert dan M. Auli, *wav2vec: Unsupervised Pre-training for Speech Recognition,* Arxiv, 2019.

[20] J. Spijkervet dan J. A. Burgoyne, *Contrastive Learning of Musical Representations,* Arxiv, 2021.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser dan I. Polosukhin, Attention Is All You Need, Arxiv, 2017.

[22] R. S. C. M. Jeffrey Pennington, “GloVe: Global Vectors for Word Representation,” dalam *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, 2014.

[23] A. Baevski, S. Steffen dan M. Auli, *vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations,* Arxiv, 2019.

[24] A. Conneau, A. Baevski, R. Collobert, A. Mohamed dan M. Auli, *Unsupervised Cross-lingual Representation Learning for Speech Recognition,* Arxiv, 2020.

[25] R. Ardilla dan e. al., *Common Voice: A Massively-Multilingual Speech Corpus,* 2020.

[26] J. W. N. C. S. Z. B. X. Cheng Yi, *Applying wav2vec2.0 to Speech Recognition in Various Low-Resource Languages,* Arxiv, 2021.

[27] Q. Xu, A. Baevski dan M. Auli, *Simple and Effective Zero-shot Cross-lingual Phoneme Recognition,* Arxiv, 2021.

[28] Huggingface, “Datasets Documentation,” [Online]. Available: https://huggingface.co/docs/datasets/index. [Diakses 19 Desember 2021].

[29] A. P. F. Naiborhu dan S. N. Endah, “LSTM, Indonesian Continuous Speech Recognition Using CNN and Bidirectional,” dalam *2021 5th International Conference on Informatics and Computational Sciences (ICICoS)*, Semarang, Indonesia, 2021.

[30] V. Panayotov, G. Chen, D. Povey dan S. Khudanpur, *LibriSpeech: An ASR Corpus based on Public Domain Audio Books*, 2015.