BAB 2
LANDASAN TEORI

2.1 Pengenalan Ucapan Otomatis

Pengenalan ucapan otomatis atau yang lebih dikenal sebagai *automatic speech recognition* (ASR) adalah teknologi yang mengubah input ucapan menjadi teks [29]. ASR sendiri bermula pada tahun 1950-an dengan sistem yang bernama Audrey. Pada pengembangan Audrey, sistem ini dirancang untuk mendeteksi angka dari frekuensi forman [30]. Namun seiring berkembangnya teknologi, kecanggihan ASR telah membuka peluang-peluang baru sehingga pengenalan ucapan otomatis yang berawal hanya untuk mendeteksi ucapan suara untuk angka dan berevolusi ke sistem deteksi ucapan. Alhasil dengan adanya pengembangan sistem ini, manusia dapat memberikan input ke dalam komputer dengan hanya menggunakan suaranya tanpa harus mengetik atau menggunakan tangannya untuk memberi input kepada komputer. Tidak hanya itu, sistem ini juga pada telah dikembangkan untuk sistem-sistem berguna lainnya seperti sebagai alat penerjemah bahasa dan asisten pribadi virtual [31, 32].

Sistem ini bekerja dengan menggunakan teknik populer yang sering disebut sebagai pemodelan akustik dan pemodelan bahasa. Pemodelan akustik digunakan untuk merepresentasikan hubungan statistik antara segmen linguistik sinyal audio dan fonem. Lalu, pemodelan bahasa merepresentasikan distribusi probabilitas segmen kata dalam urutan kata tertentu [33, 34].

Performa sistem sistem ini dapat dievaluasi menggunakan dua parameter utama, yaitu :

1. Akurasi : variabel ini merupakan persentase kesalahan dalam mendeteksi kata dan frasa yang diucapkan menjadi teks.
2. Kecepatan : variabel ini merupakan tingatan sejauh mana mesin dapat mengimbangi konversi frasa yang diucapkan menjadi teks dengan kecepatan ucapan pembicaraan manusia.

Dalam implementasi sistem ini, terdapat beberapa aspek yang harus diperhatikan diantaranya [34, 35] :

1. Lingkungan suara yang bising Lingkungan yang bising dapat mengurangi akurasi dan kestabilan sistem pengenalan ucapan, sehingga menantang

peneliti untuk memproses data suara dengan mengekstrak fitur noise selama proses pengenalan.

2. Data bahasa Sumber daya yang terbatas dapat sangat mempengaruhi proses pengenalan suara, di mana lebih banyak data latih menghasilkan akurasi yang lebih baik. Namun, terbatasnya sumber daya dapat mengakibatkan model yang kurang baik.

3. Aksen Aksen bicara juga merupakan tantangan dalam pengembangan model pengenalan suara karena setiap daerah memiliki aksennya masing-masing dan tidak terlupakan bahwa aksen juga dapat dipengaruhi oleh situasi sosial dan pribadi, seperti aspek fisiologis dan budaya seseorang.

4. Kecepatan ucapan pada suara Kecepatan ucapan sering sekali dikira tidak penting. Namun, kecepatan ucapan yang bervariatif juga mempengaruhi pengucapan. Hal ini dikarenakan adanya perubahan dalam pelafalan fonem, perluasan waktu, dan kompresi.

5. Homofon pada bahasa Homofon adalah kata-kata yang memiliki arti berbeda tetapi terdengar sama saat diucapkan, seperti ”Bank” dan ”Bang” atau ”Massa” dan ”Masa” [34]. Dalam sistem pengenalan suara, mengenali kata yang dimaksud merupakan hal yang sulit namun memiliki dampak yang sangat besar dari makna sebuah kalimat yang diucapkan.

Tetapi kembali lagi kepada kecanggihan teknologi yang semakin berkembang. Penelitian terkait model pengenalan ucapan sudah banyak diteliti sehingga implementasinya juga sudah dipermudahkan dengan adanya *pre-trained* model seperti Wav2Vec2.

2.2 Teknik Pembelajaran Mesin

Pada umumnya, teknik pembelajaran mesin terbagi menjadi tiga jenis yaitu *supervised learning*, *unsupervised learning*, dan *self-supervised learning*. Perbedaan utama antara ketiga teknik ini terletak pada cara pelabelan data yang digunakan dalam proses pelatihan model.

2.2.1 Supervised Learning

Pada teknik *supervised learning*, model dilatih menggunakan *dataset* yang terdiri dari pasangan input-output yang sudah diketahui. Dalam teknik ini, setiap contoh dalam *dataset* sudah dilengkapi dengan label output yang jelas sehingga model dapat memprediksi output dengan menggunakan data yang terdokumentasi dengan baik. Namun, metode pembelajaran ini termasuk yang paling mahal dalam hal kebutuhan sumber daya, karena memerlukan pelabelan data yang lengkap untuk setiap input dan outputnya [36].

2.2.2 Unsupervised Learning

Berbeda dengan teknik *supervised learning*, teknik *unsupervised learning* tidak membutuhkan dokumentasi yang lengkap. Dalam teknik ini, model belajar dari data yang tidak berlabel dengan mengidentifikasi pola dan struktur yang ada dalam data tersebut. Misalnya, model dapat dilatih untuk mengelompokkan data atau menemukan anomali berdasarkan karakteristik data. Teknik ini cocok untuk menemukan pola tersembunyi dalam data, tetapi tidak cocok untuk tugas-tugas yang memerlukan prediksi spesifik dengan akurasi tinggi [37].

2.2.3 Self-Supervised Learning

Terakhir, *self-supervised learning* adalah teknik di mana model belajar dari data yang tidak berlabel dengan menggunakan label yang dihasilkan secara otomatis dari data itu sendiri. Teknik ini memanfaatkan struktur atau informasi inherent dalam data untuk membuat tugas pelatihan yang tidak memerlukan pelabelan manual. Misalnya, model dapat dilatih untuk memprediksi bagian yang hilang dari data berdasarkan konteks sekitarnya [38]. Teknik ini efektif mengurangi kebutuhan data berlabel dalam jumlah besar dan memungkinkan ekstraksi fitur yang kuat dari data tidak berlabel [39].

2.2.4 Perbedaan Dalam Pelabelan Data

1. *Supervised Learning*: Membutuhkan *dataset* berlabel di mana setiap contoh data memiliki output yang diinginkan yang telah ditentukan sebelumnya. Pelabelan ini sering kali dilakukan secara manual dan bisa memakan banyak waktu dan biaya [40].

2. *Unsupervised Learning*: Tidak memerlukan label sama sekali. Model bekerja dengan data mentah untuk menemukan pola atau struktur yang tersembunyi, sehingga tidak ada biaya pelabelan data yang terlibat [41].

3. *Self-Supervised Learning*: Tidak memerlukan label yang terdokumentasi secara manual karena model ini dapat menentukan label dengan menggunakan bagian dari data yang dimiliki. Hal ini membuat teknik ini lebih efisien dalam hal biaya dan waktu pelabelan [42].

2.3 Feature Extraction

Setelah membahas tentang jenis-jenis pembelajaran mesin dan apa perbedaanya. Perlu diketahui bahwa mesin membutuhkan data yang akan digunakan untuk pembelajaran. Setelah data telah dipersiapkan, proses berikutnya adalah memahami cara kerja gambar dalam mengestraksi fitur yang ada sehingga dapat menghasilkan output yang optimal.

Pertama-tama, *feature extraction* merupakan sebuah proses yang mengubah data mentah menjadi representasi fitur yang lebih sederhana, efisien, dan informatif. Tujuannya adalah untuk mengekstrak informasi yang relevan dari data yang kompleks atau besar sehingga dapat digunakan lebih efektif dalam analisis atau pembelajaran mesin [43].

Proses feature extraction umumnya dilakukan dalam beberapa langkah. Pertama, data mentah ditangkap, semisalnya gambar atau sinyal suara. Kemudian, fitur-fitur penting diekstraksi dari data ini dengan menggunakan metode tertentu, seperti transformasi matematis, deteksi pola, atau algoritma pembelajaran mesin. Proses ini dapat menghasilkan representasi fitur yang lebih sederhana dan lebih mudah diinterpretasikan daripada data asli [44, 45].

Contoh dari feature extraction dalam konteks pengolahan gambar adalah ekstraksi fitur seperti tepi, tekstur, atau bentuk objek dari gambar. Dalam pengenalan suara, fitur-fitur seperti spektrum frekuensi atau pola kejadian tertentu dalam sinyal suara diekstraksi untuk digunakan dalam analisis model pengenalan ucapan seperti Wav2Vec2 [44, 45].

2.3.1 Perbedaan Mel Spectogram dan MFCC

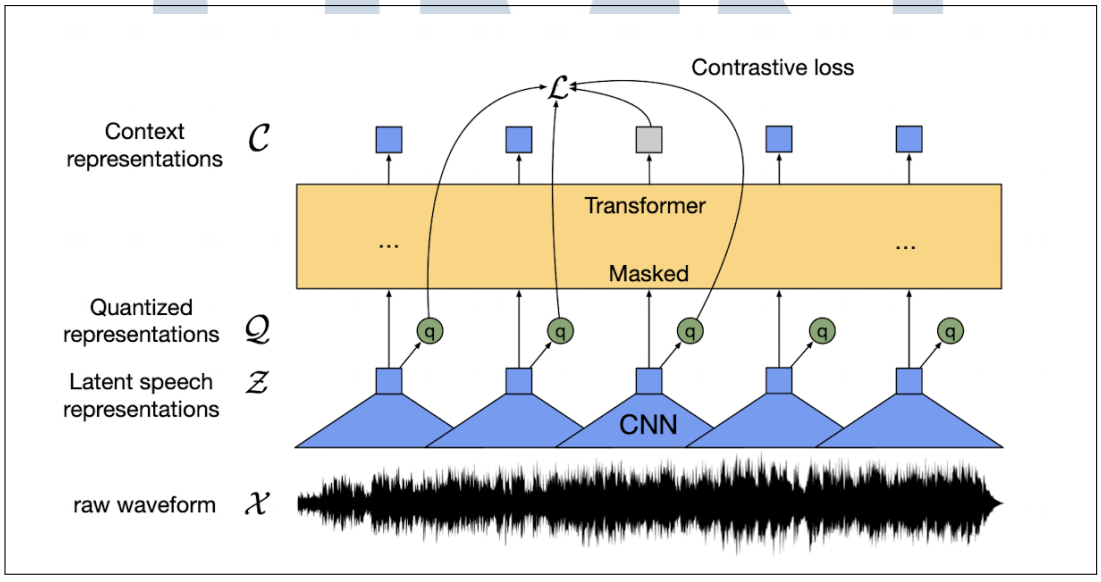
Mel spectrogram dan MFCC (Mel-frequency cepstral coefficients) adalah dua representasi fitur penting dalam pemrosesan sinyal suara. Pada satu sisi,

mel spectrogram menggambarkan spektrum frekuensi suara dalam rentang waktu tertentu, sementara MFCC menghasilkan representasi cepstral dari sinyal suara.

Terlebih dari itu, perbedaan utama antara keduanya adalah Mel spectrogram memberikan detail distribusi energi frekuensi suara, sementara MFCC menghasilkan representasi yang lebih abstrak dan kompak yang meniru cara manusia memproses suara. MFCC lebih efisien untuk diproses oleh model pembelajaran mesin karena mengurangi dimensi data sambil mempertahankan informasi yang paling relevan, sedangkan Mel spectrogram memberikan informasi detail yang mungkin hilang dalam MFCC. Pada model Wav2Vec2 sendiri, representasi fitur yang digunakan mirip dengan MFCC dalam beberapa aspek, yang bertujuan untuk menghasilkan representasi fitur suara yang efisien dan informatif.

2.4 Wav2Vec2

Wav2Vec2 merupakan model yang dirancang oleh Facebook AI dengan menggunakan teknik pembelajaran *self-supervised learning* sehingga pengimplementasian model ini tidak memerlukan label. Wav2Vec2 dirilis pada September 2020 oleh Alexei Vaevski, Michael Auli, dan Alex Conneau. Pada perancangan model ini, penemu menggunakan konsep *novel constrastive pre-training objective* dimana model ini dapat mempelajari representasi ucapan yang kuat dengan angka lebih dari 50.000 jam suara bahasa inggris tanpa label [34].



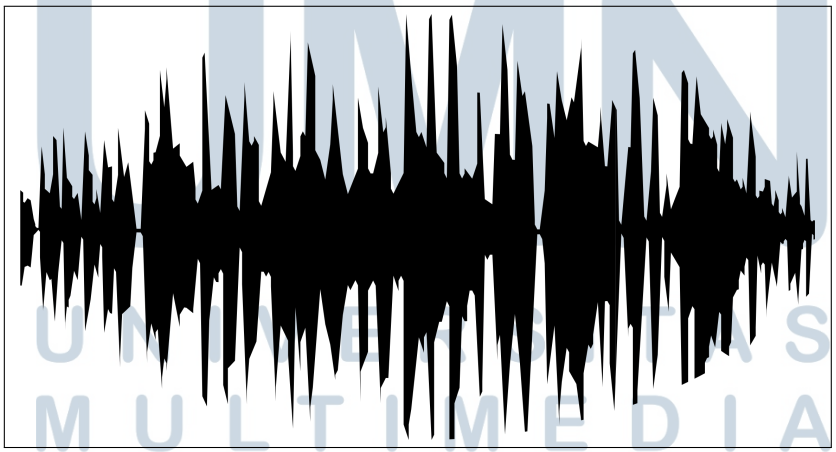
Gambar 2.1. Gambaran model arsitektur wav2vec2

Gambaran arsitektur yang dapat dilihat pada Gambar 2.1 menujukkan bahwa

model ini bekerja dengan memanfaatkan beberapa jaringan. Pertama, terdapat lapisan *convulational neural network* yang digunakan untuk menangkap fitur-fitur awal dari sinyal audio mentah. Kemudian, terdapat lapisan transformer, bagian utama dari Wav2Vec2, yang terdiri dari banyak lapisan transformer untuk menangkap representasi konteks global dari data audio. Selanjutnya, model ini menggunakan konsep pembilang (quantizer) yang mirip dengan VQ-VAE untuk memetakan representasi laten menjadi vektor diskrit yang diambil dari codebook [34]. Dilengkapi dengan fungsi loss kontras yang digunakan untuk memaksimalkan kesamaan antara representasi yang benar dan meminimalkan kesamaan dengan representasi negatif. Terakhir, ada lapisan proyeksi yang mengubah representasi dari transformer menjadi vektor fitur yang lebih kecil dan lebih mudah dikelola untuk tugas downstream [36–40].

2.5 Convolutional Neural Network

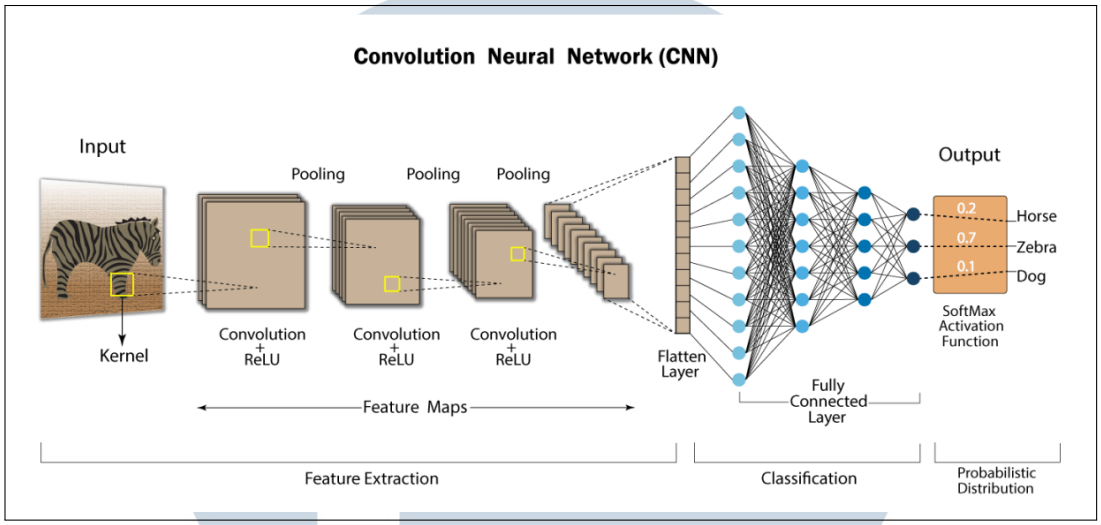
Dalam konteks implementasi *convolutional neural network* (CNN), pendekatan deteksi suara sama dengan deteksi gambar. Dimana pada gambar umumnya akan memiliki fitur-fitur yang tersusun menyerupai pola. Walau gambar dapat dilihat secara langsung dan suara tidak. Suara yang ditangkap oleh mesin akan menghasilkan gelombang frekuensi suara seperti yang dapat dilihat pada Gambar 2.2.



Gambar 2.2. Gambaran gelombang frekuensi suara

Setelah gelombang frekuensi ini tertangkan oleh mesin. CNN akan merubah input gambar ke dalam lapisan-lapisan arsitektur CNN untuk mendapatkan hasil akhir. Walaupun arsitektur CNN ini memiliki banyak variasi, biasanya lapisan ini

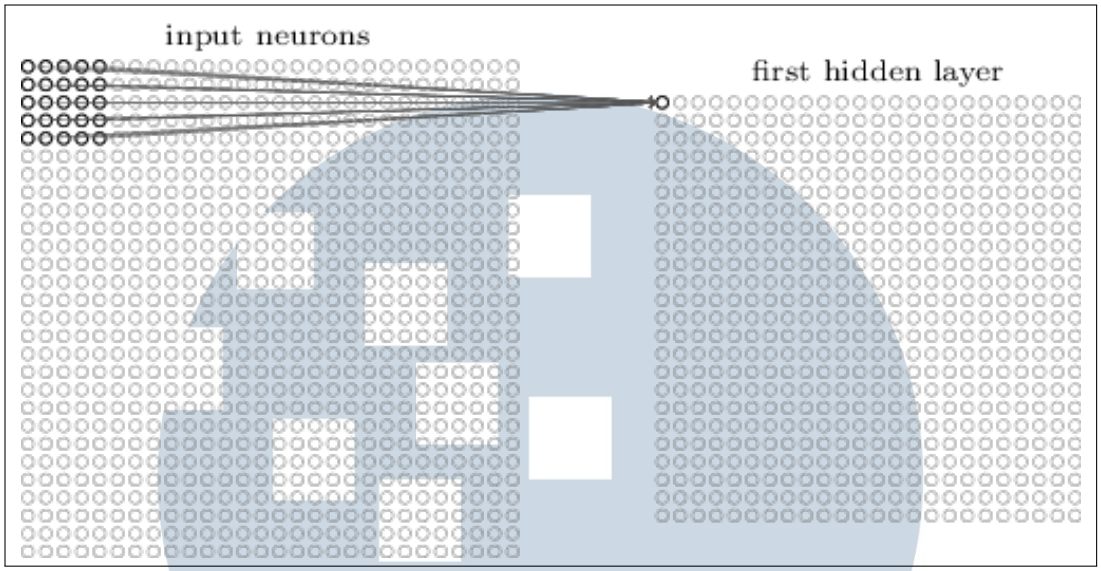
terdiri dari *convolution layer*, *pooling layer*, dan dilanjuti dengan satu atau lebih *fully connected layer* [46] seperti yang dapat dilihat pada Gambar 2.3.



Gambar 2.3. Arsitektur *convolutional neural network*

2.5.1 Convolution Layer

Convolution layer atau lapisan konvolusion berguna untuk melakukan ekstraksi fitur yang dapat digunakan untuk pembelajaran mesin melalui representasi-representasi fitur pada input. Pada Gambar 2.4 dapat dilihat bahwa output dihasilkan melalui perkalian produk dalam antara bidang penerima lokal (*local receptive field*) dan filter. Agar dapat mendapatkan semua hasil perkalian produk dalam, bidang penerima lokal harus dapat digeser dengan nilai langkah (*stride value*) [46]. Lalu lapisan ini juga akan memanfaatkan fungsi aktivasi non-linear, seperti lapisan yang sedang populer belakangan ini yaitu ReLU karena kinerjanya yang lebih cepat dibandingkan dengan fungsi sigmoid dan tangen hiperbolik [46].



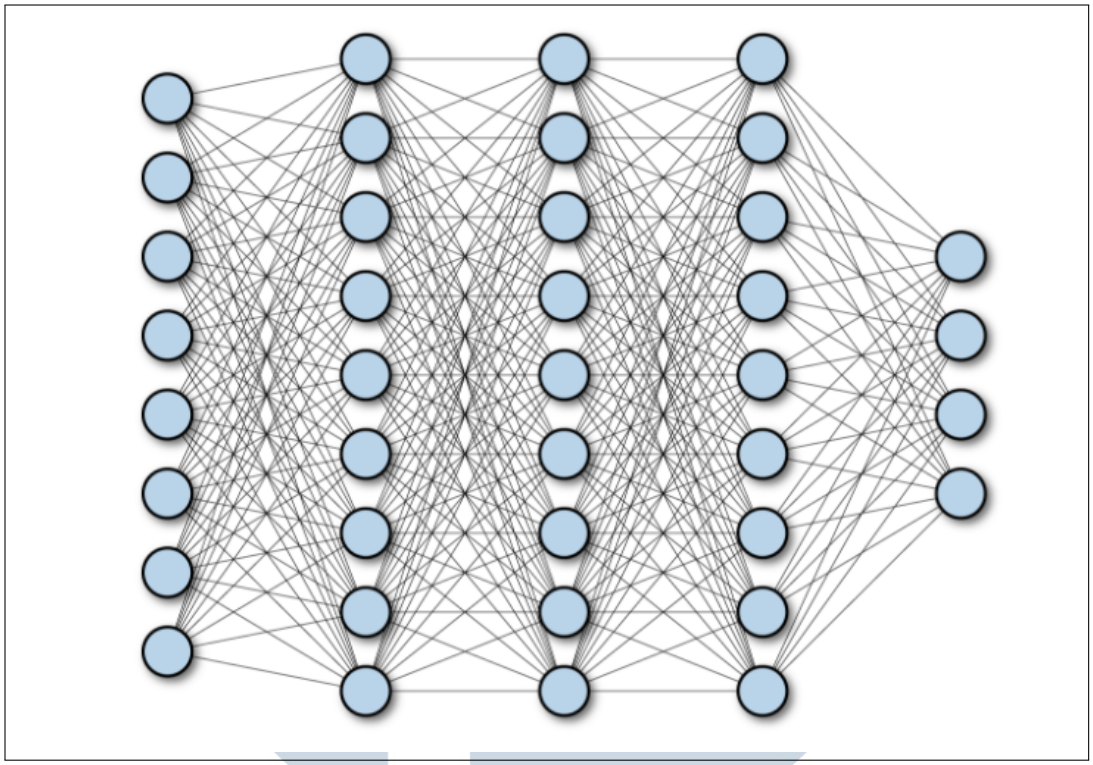
Gambar 2.4. Lapisan konvolusion

2.5.2 Pooling Layer

Pooling layer atau lapisan *pooling* berguna untuk mengurangi resolusi spasial dari *feature map*. Salah satu lapisan *pooling* yang sering digunakan adalah *max pooling* karena kemampuannya untuk mencari nilai maksimum dari suatu area lokal [46].

2.5.3 Fully Connected Layer

Fully connected layer, seperti yang ditunjukkan pada Gambar 4, adalah lapisan terakhir dalam arsitektur CNN yang menghubungkan setiap neuron dari *pooling layer* ke neuron output. Untuk klasifikasi, fungsi yang sering digunakan pada lapisan output adalah SoftMax [46].



Gambar 2.5. Gambaran *fully connected layer*

2.6 Lapisan Transformer

Lapisan Transformer, seperti yang dapat dilihat pada Gambar 2.6, adalah komponen utama dari arsitektur model Transformer, yang diperkenalkan oleh Vaswani et al. dalam makalah ”Attention is All You Need” pada tahun 2017 [47]. Model ini telah menjadi dasar dari banyak kemajuan dalam pemrosesan bahasa alami (NLP), termasuk model seperti BERT, GPT, dan T5. Tetapi dalam konteks sistem pengenalan ucapan otomatis, Transformer juga digunakan untuk meningkatkan akurasi dan efisiensi dalam mengenali dan mengonversi sinyal suara menjadi teks. Komponen utama dari lapisan transformer merupakan *self-attention mechanism*, *feed-forward neural network*, *layer normalization*, dan *residual connections*.

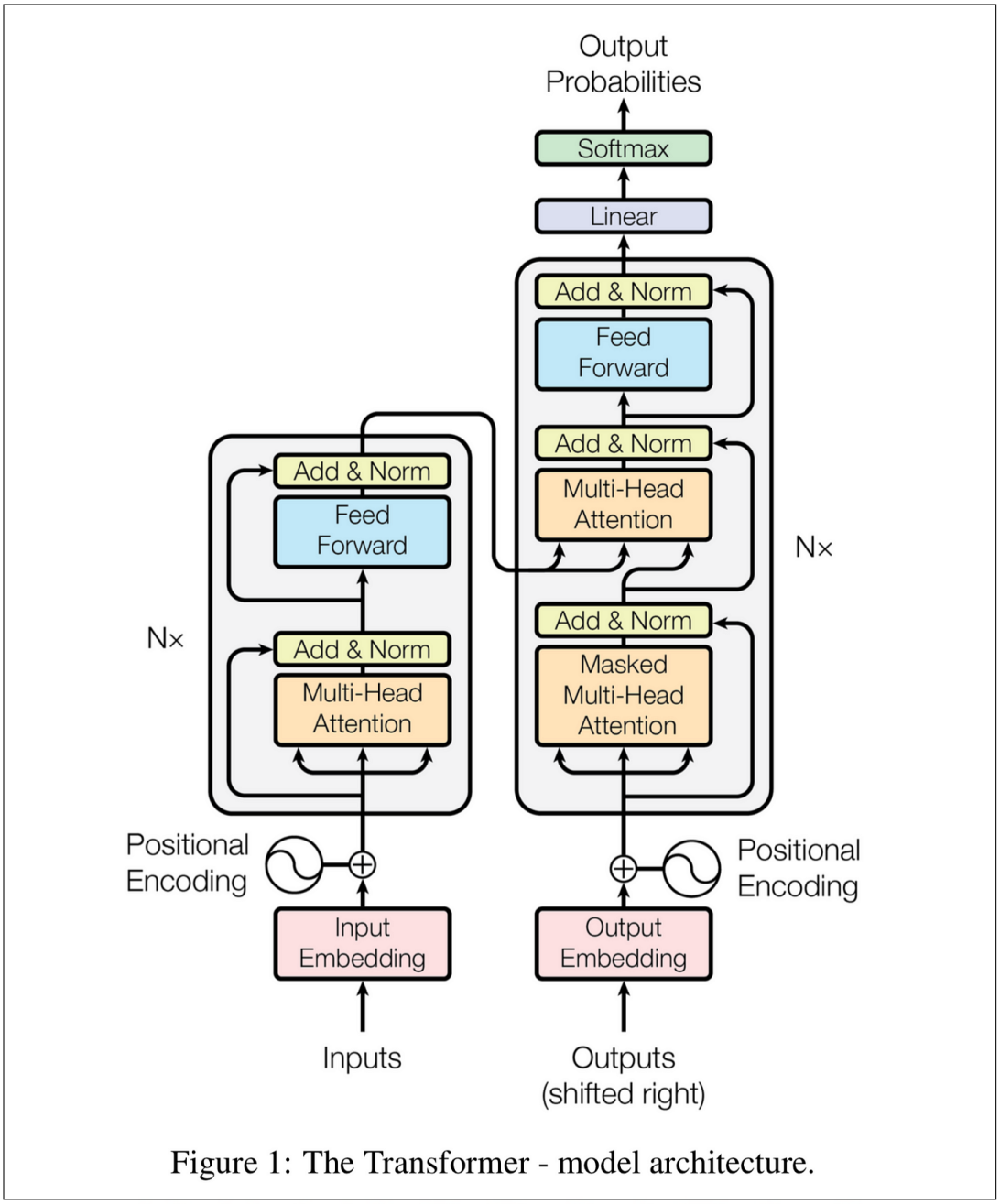


Figure 1: The Transformer - model architecture.

Gambar 2.6. Arsitektur model transformer

2.6.1 Self-Attention Mechanism

Komponen ini dapat digunakan dan bertugas untuk memperhatikan seluruh urutan audio saat memproses setiap frame atau segmen suara. Hal ini sangat penting karena suara memiliki konteks temporal, dan memahami hubungan antara berbagai bagian dari sinyal audio membantu dalam mengenali pola suara secara lebih akurat.

Mekanisme ini menghitung skor perhatian antara setiap pasangan frame audio, yang kemudian digunakan untuk menghasilkan representasi kontekstual baru untuk setiap frame.

2.6.2 Feed-Forward Neural Network

Komponen ini bertanggung jawab dalam memproses representasi kontekstual yang dihasilkan oleh mekanisme self-attention melalui pemrosesan non-linear untuk mengidentifikasi fitur-fitur yang relevan dari sinyal suara. Struktur jaringan ini terdiri dari dua lapisan linier dengan fungsi aktivasi seperti ReLU di antaranya, yang membantu dalam memodelkan kompleksitas sinyal audio.

2.6.3 Layer Normalization

Komponen ini berfungsi untuk menormalkan output dari self-attention dan feed-forward network, terlebih dari itu juga membantu dalam stabilisasi dan percepatan pelatihan model. Layer normalization mengurangi ketergantungan pada distribusi input dan memastikan stabilitas numerik selama pelatihan.

2.6.4 Residual Connections

Komponen yang terakhir yaitu *residual connections* digunakan agar dapat membantu dalam melatih model yang lebih dalam dengan menghindari masalah degradasi gradien, memungkinkan sinyal gradien untuk mengalir lebih lancar ke lapisan sebelumnya, yang sangat penting untuk mempelajari representasi suara yang kompleks. Residual connections menambahkan input asli ke output dari lapisan tertentu, memastikan bahwa informasi penting tidak hilang selama pemrosesan.

2.7 Algoritma Klasifikasi Connectionist Temporal Classification

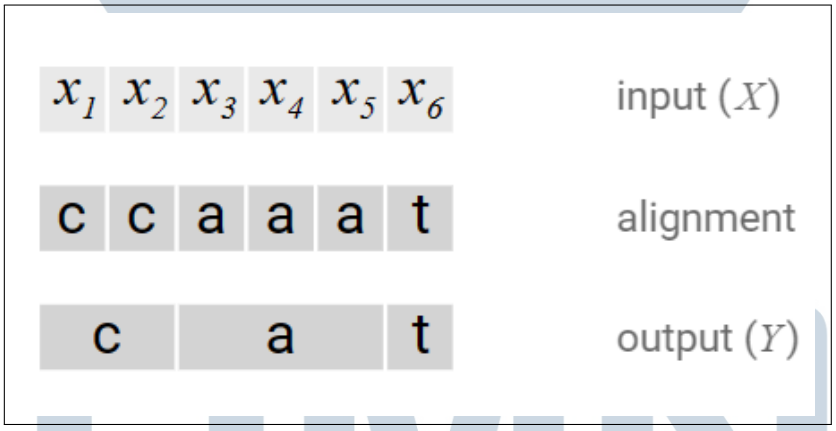
Dalam membuat sistem pengenalan ucapan, secara matematis, terkadang input dan output yang dimiliki tidak akan selalu selaras. Secara matematis, pertimbangkan urutan input X = [X1, X2, X3, ..., Xm] dan urutan output [Y1, Y2, ..., Yn]. Jika dilakukan pemetaan, permasalahan-permasalahan yang akan timbul adalah sebagai berikut :

1. Panjang X (m) dan panjang Y (n) berbeda.

2. Rasio panjang antara X dan Y akan bervariasi dari orang ke orang.

3. Tidak ada pencocokan urutan antara X dan Y.

Tanpa adanya implementasi CTC, sebetulnya sebuah sistem dapat bekerja dengan baik seperti yang dapat dilihat pada Gambar 2.7. Pada gambar tersebut, dapat dilihat bahwa input yang masuk akan selaras dengan hasil akhir yang diinginkan. Tetapi hal ini menjadi masalah ketika model diminta untuk memprediksi kata-kata yang memiliki karakter yang duplikat seperti 'l' pada kata 'Hello'. Dengan diimplementasikannya metode CTC, CTC memungkinkan model untuk belajar menyesuaikan urutan input dan output selama pelatihan. Tujuannya adalah untuk menemukan penyesuaian yang paling selaras antara X dan Y. Selain membantu dalam mengoptimalkan prediksi *output* pada saat pelatihan, metode klasifikasi ini juga dapat membantu dalam menangani *dataset* yang berbeda selama pelatihan dan sekaligus menyederhanakan proses pelatihan secara signifikan [**?**].



Gambar 2.7. Cara model Wav2Vec2 tanpa mengimplementasikan "blank" token

Seperti yang sudah dijelaskan pada paragraf sebelumnya terkait kata-kata yang memiliki karakter duplikat yang bersebrangan, Pada Gambar 2.7 dapat dilihat bahwa deteksi audio untuk menghasilkan kata "cat" berhasil. Namun, apabila dilakukan hal yang sama pada kata "hello" dengan input $[h, h, e, l, l, l, o]$, input ini akan menghasilkan output "helo" akibat cara kerja model Wav2Vec2 yang akan melakukan *merging* pada karakter yang sudah dilaraskan. Oleh karena itu, dengan menggunakan metode CTC, diperkenalkan karakter baru yaitu "blank" token yang disimbolkan dengan ε. Simbol ε tidak merepresentasikan apa-apa dan akan dihapus pada output. Dengan implementasi ini, data CTC akan menjadi selaras dan dapat membedakan karakter yang seharusnya ada satu atau lebih seperti yang dapat dilihat pada Gambar 2.8.

Gambar 2.8. Cara kerja CTC setelah diperkenalkan ”blank” token

2.7.1 Dari Output Jaringan ke Pemberian Label

Mengerti cara kerja CTC secara lebih mendalam, ide dasar metode ini adalah untuk menafsirkan output jaringan sebagai distribusi probabilitas kondisional di atas semua urutan label output yang memungkinkan. Pada setiap langkah waktu, jaringan menghasilkan distribusi probabilitas di atas set label $L' = L \cup \{\text{blank}\}$, di mana L adalah semua label tugas dan blank mewakili ’tidak ada label’. Aktivasi y_t^k adalah probabilitas mengamati label k dari L' pada waktu t. Diberikan input urutan $\mathbf{x}$ dengan panjang T, probabilitas kondisional $p(\boldsymbol{\pi}|\mathbf{x})$ dari jalur π adalah [48]:

$$p(\pi \mid \mathbf{x}) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T, \tag{2.1}$$

Jalur-jalur tersebut dipetakan ke dalam urutan label dengan operasi $\mathscr{B}$ yang menghapus label berulang dan kemudian blank dalam urutan. Untuk urutan label yang diberikan $\mathbf{l} \in L^U$, $U \leq T$, lebih dari satu π sesuai dengan $\mathbf{l}$. Misalnya, $\mathscr{B}(\text{aa}--\text{ab}-) = \mathscr{B}(-\text{a}-\text{a}-\text{abb}) = \text{aab}$, di mana ’-’ melambangkan blank. Pada saat mengevaluasi probabilitas kondisional dari $\mathbf{l}$ sebagai jumlah probabilitas dari semua jalur yang sesuai dapat digunakan rumus [48]:

$$p(\mathbf{l} \mid \mathbf{x}) = \sum_{\pi \in \mathscr{B}^{-1}(\mathbf{l})} p(\pi \mid \mathbf{x}). \tag{2.2}$$

Namun, perhitungan ini memiliki permasalahan [48] karena jumlah

jalur yang sesuai tumbuh secara eksponensial dengan U. Hal ini dapat diselesaikan dengan algoritma pemrograman dinamis yang mirip dengan algoritma *forward-backward* untuk HMMs, dengan memecahkan jumlah jalur yang sesuai dengan pelabelan $\mathbf{l}$ menjadi jumlah berulang dari jalur yang sesuai. Untuk memperhitungkan blank dalam output jalur, Dapat dipertimbangkan urutan label yang dimodifikasi $\mathbf{l}' \in L'^{2U+1}$, dengan blank ditambahkan di awal dan akhir $\mathbf{l}$, serta di antara setiap pasangan label yang berurutan. Dalam menghitung probabilitas awalan $\mathbf{l}'$, transisi antara label blank dan non-blank, serta antara setiap pasangan label non-blank yang berbeda, diperbolehkan. Lalu, dapat dihitung juga fungsi kerugian CTC sebagai negatif dari log probabilitas pelabelan urutan dengan benar [48]:

$$\mathrm{CTC}(\mathbf{l}, \mathbf{x}) = -\ln p(\mathbf{l}|\mathbf{x}). \tag{2.3}$$

Selama pelatihan, untuk melakukan gradien *backpropagate* melalui lapisan output, diperlukan turunan dari fungsi kerugian yang dibandingkan dengan output $\{a_t^k | t \in [1,T], k \in L'\}$ sebelum fungsi aktivasi digunakan. Fungsi aktivasi softmax dapat ditulis sebagai berikut [48]:

$$y_k^t = \frac{e^{a_k^t}}{\sum_{k'} e^{a_{k'}^t}}, \tag{2.4}$$

di mana k' berselisihan pada L', dengan turunan terhadap a_k^t adalah:

$$\frac{\partial \mathrm{CTC}(\mathbf{l}, \mathbf{x})}{\partial a_k^t} = y_k^t - \frac{1}{p(\mathbf{l} \mid \mathbf{x})} \sum_{\substack{\pi \in \mathscr{B}^{-1}(\mathbf{l}): \\ \pi_t = k}} p(\pi \mid \mathbf{x}) \tag{2.5}$$

di mana $\sum_{\pi \in \mathscr{B}^{-1}(\mathbf{l}): \pi_t = k} p(\pi \mid \mathbf{x})$ adalah jumlah probabilitas dari semua jalur yang sesuai dengan label k pada waktu t.

Saat jaringan digunakan untuk memprediksi, semua langkah-langkah yang telah dilakukan akan dikonversi menjadi urutan label. Sebab kompleksitas komputasi yang tumbuh secara eksponensial dengan panjang jalur, Mengakibatkan penemuan urutan label yang paling mungkin $\mathbf{l}$ menjadi tidak praktis. Untungnya terdapat banyak alternatif aproksimasi seperti *best path decoding* yang menjadi salah satu metode yang paling umum digunakan. Metode ini mengasumsikan bahwa output yang paling mungkin akan sesuai dengan [48]:

$$\hat{\mathrm{l}} \approx \mathscr{B}(\pi^*)$$
$$\text{where } \pi^* = \arg\max_{\pi} p(\pi \mid \mathbf{x}). \tag{2.6}$$

Meskipun tidak dijamin untuk menemukan urutan label yang paling mungkin, solusi ini cukup baik dalam banyak kasus dan prosedur komputasinya sederhana.

2.8 Low Resource Languages

Low resource languages atau bahasa dengan sumber daya rendah mengacu kepada ketersediaan data bahasa agar dapat digunakan untuk pembelajaran mesin. Hal ini mengartikan bahwa walaupun pembicara di negara itu banyak, *low resource languages* lebih membicarakan terhadap sumber daya digital yang masih belum mendukung untuk digunakan pada mesin. Bahasa Jawa dan Sunda adalah contoh nyata dari bahasa-bahasa dengan sumber daya terbatas. Walaupun kedua bahasa ini memiliki jumlah penggunanya yang besar, bahasa-bahasa ini masih menghadapi hambatan dalam pemrosesan bahasa alami karena kekurangan data yang dapat diakses oleh publik. Keterbatasan ini membatasi kemampuan untuk mengembangkan teknologi seperti sistem pengenalan teks ke ucapan dan model bahasa yang canggih, yang sangat bergantung pada data pelatihan yang kaya dan komprehensif [9, 10].

2.8.1 Bahasa Sunda dan Jawa

Data bahasa Sunda dan Jawa berasal dari OpenSLR, yang merupakan repositori sumber terbuka untuk sumber daya pengenalan suara dan teknologi bahasa. Proyek ini, seperti yang dicatat dalam publikasi di konferensi SLTU [49], menyediakan koleksi data suara yang bertujuan untuk membantu dalam pengembangan sistem pengenalan suara untuk bahasa-bahasa yang kurang dilayani. OpenSLR memberikan akses terbuka kepada peneliti dan pengembang untuk menggunakan data ini dalam berbagai aplikasi terkait suara dan bahasa [49].

2.9 Teknik Augmentasi Audio

Selain alat-alat seperti TorchAudio, SpecAugment, dan MixSpeech. Alat yang tidak kalah menarik adalah Audiomentations. Hal yang menarik dari

teknik augmentasi audiomentations adalah sistem antarmukanya [50] yang mudah digunakan dan dokumentasinya yang lengkap. Terlebih dari itu, fleksibilitas yang ditawarkan oleh audiomentations dapat memungkinkan penggunanya dalam mengkombinasikan beberapa teknik augmentasi dengan mudah. Audiomentations menawarkan berbagai macam teknik yang dapat digunakan seperti yang ditunjukkan pada Tabel 2.1.

Tabel 2.1. Teknik augmentasi audiomentations dan fungsinya

No.	Teknik Augmentasi	Fungsi
1	AddBackgroundNoise	Campuran suara lain untuk menambahkan latar belakang kebisingan
2	AddColorNoise	Menambahkan kebisingan dengan warna tertentu
3	AddGaussianNoise	Menambahkan kebisingan Gaussian ke sampel audio
4	AddGaussianSNR	Menyuntikkan kebisingan Gaussian menggunakan perbandingan sinyal-ke-kebisingan yang dipilih secara acak
5	AddShortNoises	Campuran berbagai suara kebisingan pendek
6	AdjustDuration	Memotong atau memanjangkan audio untuk mencocokkan durasi target
7	AirAbsorption	Menambahkan redaman yang bergantung pada frekuensi untuk mensimulasikan absorpsi udara
8	Aliasing	Menghasilkan artefak aliasing dengan mereduksi sampel tanpa low-pass filtering dan kemudian upsampling

Lanjut pada halaman berikutnya

Tabel 2.1 Teknik augmentasi audiomentations dan fungsinya (lanjutan)

No.	Teknik Augmentasi	Fungsi
9	ApplyImpulseResponse	Mengonvolusi audio dengan respons impuls yang dipilih secara acak
10	BandPassFilter	Melakukan filtering band-pass dalam parameter-parameter yang diacak
11	BandStopFilter	Melakukan filtering band-stop (notch) dalam parameter-parameter yang diacak
12	BitCrush	Melakukan reduksi bit tanpa dithering
13	Clip	Mengklip sampel audio ke nilai minimum dan maksimum yang ditentukan
14	ClippingDistortion	Merusakkan sinyal dengan mengklip persentase sampel secara acak
15	Gain	Mengalikan audio dengan faktor gain acak
16	GainTransition	Secara bertahap mengubah gain selama rentang waktu yang acak
17	HighPassFilter	Melakukan filtering high-pass dalam parameter-parameter yang diacak
18	HighShelfFilter	Melakukan filtering high shelf dengan parameter-parameter yang diacak
19	Lambda	Menggunakan transformasi yang ditentukan pengguna
20	Limiter	Melakukan kompresi rentang dinamis membatasi sinyal audio

Lanjut pada halaman berikutnya

Tabel 2.1 Teknik augmentasi audiomentations dan fungsinya (lanjutan)

No.	Teknik Augmentasi	Fungsi
21	LoudnessNormalization	Mengaplikasikan gain untuk mencocokkan loudness target
22	LowPassFilter	Melakukan filtering low-pass dalam parameter-parameter yang diacak
23	LowShelfFilter	Melakukan filtering low shelf dengan parameter-parameter yang diacak
24	Mp3Compression	Mengompresi audio untuk menurunkan kualitas
25	Normalize	Mengaplikasikan gain sehingga level sinyal tertinggi menjadi 0 dBFS
26	Padding	Menggantikan bagian acak dari awal atau akhir dengan padding
27	PeakingFilter	Melakukan filtering peaking dengan parameter-parameter yang diacak
28	PitchShift	Menggeser pitch ke atas atau ke bawah tanpa mengubah tempo
29	PolarityInversion	Membalik sampel audio ke atas ke bawah, membalik polaritas mereka
30	RepeatPart	Mengulangi subbagian audio sejumlah kali
31	Resample	Meresample sinyal ke sampling rate yang dipilih secara acak
32	Reverse	Membalik audio sepanjang sumbu waktunya
33	RoomSimulator	Mensimulasikan efek ruangan pada sumber audio

Lanjut pada halaman berikutnya

Tabel 2.1 Teknik augmentasi audiomentations dan fungsinya (lanjutan)

No.	Teknik Augmentasi	Fungsi
34	SevenBandParametricEQ	Menyesuaikan volume dari 7 pita frekuensi
35	Shift	Menggeser sampel ke depan atau ke belakang
36	SpecChannelShuffle	Mengacak saluran dalam spektrogram
37	SpecFrequencyMask	Mengaplikasikan masker frekuensi pada spektrogram
38	TanhDistortion	Melakukan distorsi tanh untuk merusak sinyal
39	TimeMask	Membuat bagian acak dari audio menjadi senyap
40	TimeStretch	Mengubah kecepatan tanpa mengubah pitch
41	Trim	Memotong kebisuan awal dan akhir dari audio

Dari ke 41 teknik yang terdaftar pada Tabel 2.1, teknik yang akan digunakan pada riset ini adalah *gaussian noise*, *time stretch*, *pitch shift*, dan *shift*.

2.9.1 Gaussian Noise

Teknik ini umumnya digunakan untuk menambahkan variasi pada data suara dengan menghasilkan suara yang lebih realistis. Dengan menambahkan kebisingan Gaussian ke audio, model dapat dilatih untuk menjadi lebih toleran terhadap gangguan suara yang mungkin terjadi dalam lingkungan nyata [51].

2.9.2 Time Stretch

Teknik ini memanipulasi durasi audio tanpa mempengaruhi *pitch*, sehingga membantu model untuk belajar dari variasi kecepatan bicara yang mungkin terjadi dalam berbagai situasi [52].

2.9.3 Pitch Shift

Dengan mengubah tinggi nada audio tanpa mengubah kecepatan bicara, teknik ini membantu model untuk mengenali ucapan dengan nada yang berbeda [53].

2.9.4 Shift

Teknik ini menggeser audio dalam domain waktu, membantu model belajar dari variasi dalam penempatan suara, yang bermanfaat untuk mengatasi perbedaan gaya berbicara atau akustik antar pembicara [51].

2.10 Word Error Rate

Word Error Rate (WER) adalah metrik evaluasi yang umum digunakan dalam sistem pengenalan ucapan dan pemahaman ucapan. Metrik ini mengukur tingkat kesalahan dalam transkripsi teks hasil dari sistem pengenalan ucapan, dengan membandingkan teks asli dengan teks yang dihasilkan oleh sistem. WER dihitung sebagai rasio dari jumlah kata yang disalin, dihapus, atau ditambahkan oleh sistem pengenalan terhadap jumlah kata dalam teks asli [54].

Rumus Word Error Rate (WER) dapat dilihat dari rumus 4.13.

$$WER = \frac{S + D + I}{N} \times 100\% \tag{2.7}$$

Di mana: S adalah jumlah kata yang disalin, D adalah jumlah kata yang dihapus, I adalah jumlah kata yang ditambahkan, dan N adalah jumlah total kata dalam teks asli.

Metrik ini memberikan gambaran yang akurat tentang kinerja sistem pengenalan ucapan, dengan nilai WER yang lebih rendah menunjukkan tingkat kesalahan yang lebih rendah dan kualitas transkripsi teks yang lebih baik.