2. LANDASAN TEORI

Pada Bab ini akan membahas mengenai konsep, teori, metode, serta hasil dari penelitian-penelitian sebelumnya yang relevan dengan topik sistem *Automatic Speech Recognition* yang akan dikembangkan pada skripsi ini.

2.1 Suara

Suara merupakan gelombang hasil dari sebuah objek bergetar yang menyebabkan gelombang suara terbentuk. Manusia memiliki kapabilitas untuk mendengarkan suara dengan frekuensi di antara 20 Hz hingga 20 kHz (Reybrouck et al., 2019). Suara sendiri telah digunakan oleh manusia untuk berkomunikasi secara verbal dan non verbal.

Gelombang suara terdiri dari 3 komponen utama, yaitu frekuensi, amplitudo, dan durasi. Masing-masing komponen berperan penting karena membawa serangkaian informasi yang unik, dan juga memiliki panjang yang berbeda-beda. Frekuensi merupakan jumlah getaran yang terjadi dalam 1 detik dan diukur halam Hertz (Hz), Amplitudo mengukur besarnya energi pada gelombang suara, yang juga dapat menunjukan tingkat kebisingan dari suara dengan satuan ukur dB (desibel).

2.2 *Automatic Speech Recognition* (ASR)

Automatic Speech Recognition atau pengenalan suara otomatis merupakan teknologi yang memungkinkan komputer untuk menerima input suara berbentuk gelombang suara, dan memproses suara yang diucapkan menjadi sebuah kata-kata yang dapat dikenali manusia. ASR memperbolehkan manusia untuk menggunakan suara sebagai input perintah pada komputer, sehingga dapat membuat interaksi manusia dengan komputer menjadi lebih mudah, cepat, dan tepat (Trabelsi et al., 2022). Berdasarkan (Malik, M. 2020) proses tersebut dapat dinotasikan dengan input *sequence* X, dimana dapat dinotasikan dengan rumus (2.1):

$$X = X_1, X_2, \ldots\ldots, X_n \tag{2.1}$$

Dan n merupakan panjang dari input. kemudian output *sequence* Y, dimana dapat dinotasikan dengan rumus (2.2):

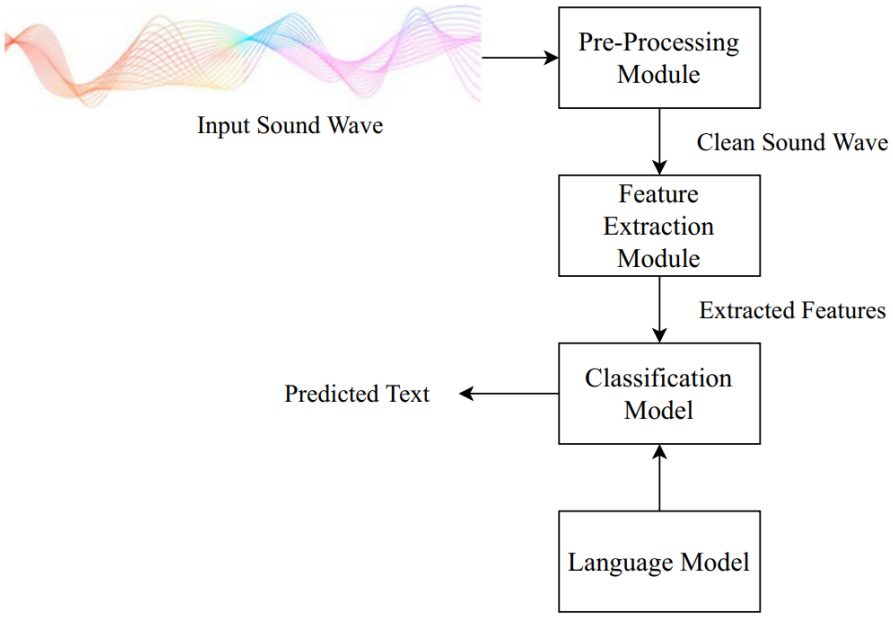
$$Y = Y_1, Y_2, \ldots\ldots, Y_m \tag{2.2}$$

Dan m merupakan panjang dari output. output *sequence* Y merupakan hasil probabilitas posterior tertinggi P(Y|X) atau bisa dinotasikan dengan rumus (2.3):

$$\begin{aligned} W &= argmax\, P(W|X) \\ &= argmax\, \frac{P(W)\, P(W|X)}{P(X)} \end{aligned} \tag{2.3}$$

Dimana P(W) merupakan probabilitas dari munculnya sebuah kata, P(X) adalah probabilitas dari X terdapat pada sinyal yang dikirim, dan P(W|X) adalah probabilitas dari sinyal *acoustic* W muncul berkoresponden dengan kata X.

ASR dapat dibagi menjadi 4 modul, yaitu *preprocessing*, *feature extraction*, *classification model*, dan juga *language model* yang bisa dilihat pada Gambar 2.1.



Gambar 2.1 Modul ASR

Sumber : Malik, M., Malik, M. K., Mehmood, K., & Makhdoom, I. (2020). Automatic speech recognition: A survey. *Multimedia Tools and Applications*, *80*(6), 9411–9457.

Dapat dilihat pada gambar 2.1 dimana sebelum menghasilkan prediksi kata-kata dari suara, secara umum terdapat 4 tahapan yang akan dilakukan pada sinyal suara. Pada tahap pertama atau pada modul *pre-processing*, file suara akan terlebih dahulu diproses untuk

beberapa tujuan, yaitu mengurangi noise, menghapus durasi, mengamplifikasi frekuensi tertentu, atau apapun sesuai dengan kebutuhan yang dibutuhkan. Kemudian pada tahap kedua atau modul *Feature Extraction,* file suara yang telah diproses akan dikelola menggunakan metode *feature extraction* seperti MFCC (*Mel-Frequency Cepstral Coefficient)*. Pada tahap *feature extraction*, file suara akan diubah menjadi matriks 2 dimensi dengan bentuk baris sebagai total *frame*, dan kolom sebagai *feature Coefficients*. Hasil matriks inilah yang biasa disebut sebagai *feature vector*, yang mana akan menjadi data *training* yang akan dipakai pada modul selanjutnya, yaitu modul *classification model*. Pada modul *classification model*, *feature vector* akan di *train* berdasarkan label dari tiap-tiap *frame*, sehingga akan menghasilkan transkrip dari file suara. Hasil transkrip yang dihasilkan dari modul *classification model* ini belum final dan akan diproses lagi pada modul terakhir, yaitu modul *language model*. Pada tahap terakhir ini, hasil transkrip akan disunting lagi berdasarkan aturan dan semantik dari *corpus* yang ada pada *language model* yang ada.

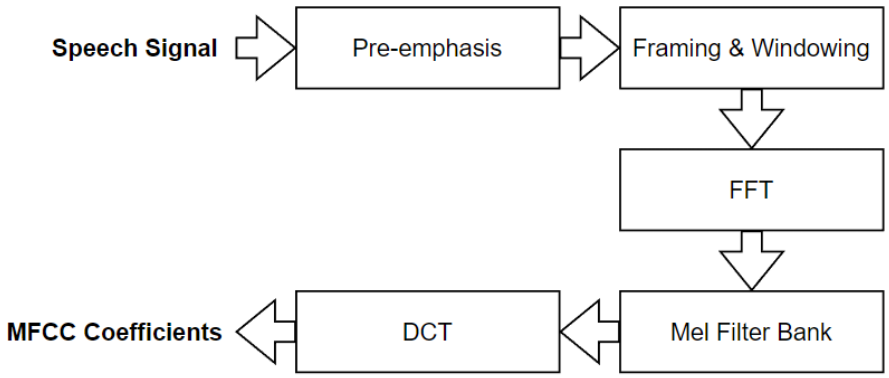
2.3 Mel-Frequency Cepstral Coefficient (MFCC)

MFCC merupakan salah satu metode terbaik dan sering digunakan untuk melakukan *feature extraction* pada ASR (Aldarmaki et al., 2022). MFCC ini merepresentasikan koefisien-koefisien dari MFC (Mel-Frequency Cepstrum), dimana MFC merefleksikan hubungan non linear dari telinga manusia terhadap frekuensi dari suara (Deng et al., 2020). MFCC menggunakan Mel-Scale untuk men-scale frekuensi agar sesuai dengan suara yang dapat didengar oleh manusia, yaitu pada range 20Hz hingga 20kHz. Frekuensi yang didapat akan dikonversi pada Mel-Scale dengan rumus (2.4):

$$Mel(f) = 2595\, log(1 + \frac{f}{700}) \quad (2.4)$$

Sehingga suara akan di filter secara linear untuk frekuensi dibawah 1000 Hz dan *logarithmic* untuk frekuensi diatas 1000 Hz (Paulose et al., 2017). Pembatasan filter tersebut dilakukan berdasarkan bagaimana telinga manusia mendengarkan suara. Pada frekuensi dibawah 1 kHz, telinga manusia meresponi suara secara linear, sehingga peningkatan frekuensi sebanyak 100 Hz akan dirasakan sebagai penambahan yang konstan pada pitch suara, sedangkan pada frekuensi diatas 1 kHz akan merespon suara secara *logarithmic* (Huang et al., 2001).

MFCC memiliki beberapa tahapan proses dalam mengekstraksi coefficient yang dapat dilihat pada Gambar 2.2.



Gambar 2.2 Tahapan Proses MFCC

Diolah oleh penulis

2.3.1 *Pre-emphasis*

Pre-emphasis merupakan tahapan *pre-processing* pada MFCC yang membantu mengurangi pengaruh dari *noise* dan juga menekankan *frequency band* pada frekuensi tinggi dari sebuah sinyal suara (Huang et al., 2001). Tahapan ini dilakukan pada sinyal suara sebelum ditransformasi menjadi domain frekuensi menggunakan metode *Fourier Transform* seperti *Discrete Fourier Transform* (DFT), ataupun *Fast Fourier Transform* (FFT).

Filter dari pre-emphasis ini diaplikasikan pada sinyal suara secara *first-order high-pass* sehingga dapat meningkatkan energi pada komponen-komponen pada frekuensi yang tinggi, seperti pada rumus (2.5)

$$y(t) = x[t] - \alpha\, x[t-1] \tag{2.5}$$

Dimana x[t] merupakan input dari sinyal suara, dan y[t] merupakan output setelah *pre-emphasis* pada waktu t , α merupakan koefisien dari *pre-emphasis* yang sering kali bernilai 0.9 hingga 0.97 sesuai dengan karakteristik dari sinyal yang di proses (Huang et al., 2001). Pre-emphasis juga membantu meningkatkan resolusi spektral dari sebuah sinyal suara sehingga menghasilkan estimasi yang lebih baik untuk *spectral envelope*.

2.3.2 *Framing and Windowing*

Framing and windowing merupakan tahapan setelah *pre-emphasis* yang dimaksudkan untuk mengolah sinyal suara menjadi bagian-bagian yang lebih kecil. Proses ini dilakukan dengan cara memecah sinyal suara menjadi bagian-bagian kecil yang disebut *frame* dengan tujuan mengurangi efek dari *spectral leakage* yang tercipta karena proses *fourier transform* dengan cara melakukan *tapering* pada setiap awal dan akhir dari setiap *frame* dengan menggunakan *window function* tertentu (Skiadopoulos & Stergiou, 2020). *Spectral leakage* merupakan kondisi kebocoran yang terjadi oleh proses *fourier transform*, dimana sinyal dengan panjang tertentu di transformasikan dari domain waktu menjadi domain frekuensi. Hal tersebut terjadi dikarenakan sinyal dianggap memiliki durasi waktu *infinite* sehingga menyebabkan diskontinuitas pada awal dan akhir dari sebuah sinyal suara. Oleh karena itu, energi spektral bocor dari lobus utama sebuah frekuensi dan akan tersebar pada *frequency bins* yang berdekatan, hingga menyebabkan distorsi pada spektrum frekuensi (Skiadopoulos & Stergiou, 2020).

Tahapan ini juga dilakukan agar sebuah sinyal suara untuk dapat diproses maupun diolah secara independen berdasarkan *frame/window* masing-masing. Untuk panjang dari *frame* biasanya bervariasi antara 20 ms hingga 40 ms. Jenis dari *window function* juga bervariasi seperti *Hamming, Hanning, Blackman windows* yang memiliki karakteristik yang berbeda.

2.3.3 *Fast Fourier Transform* (FFT)

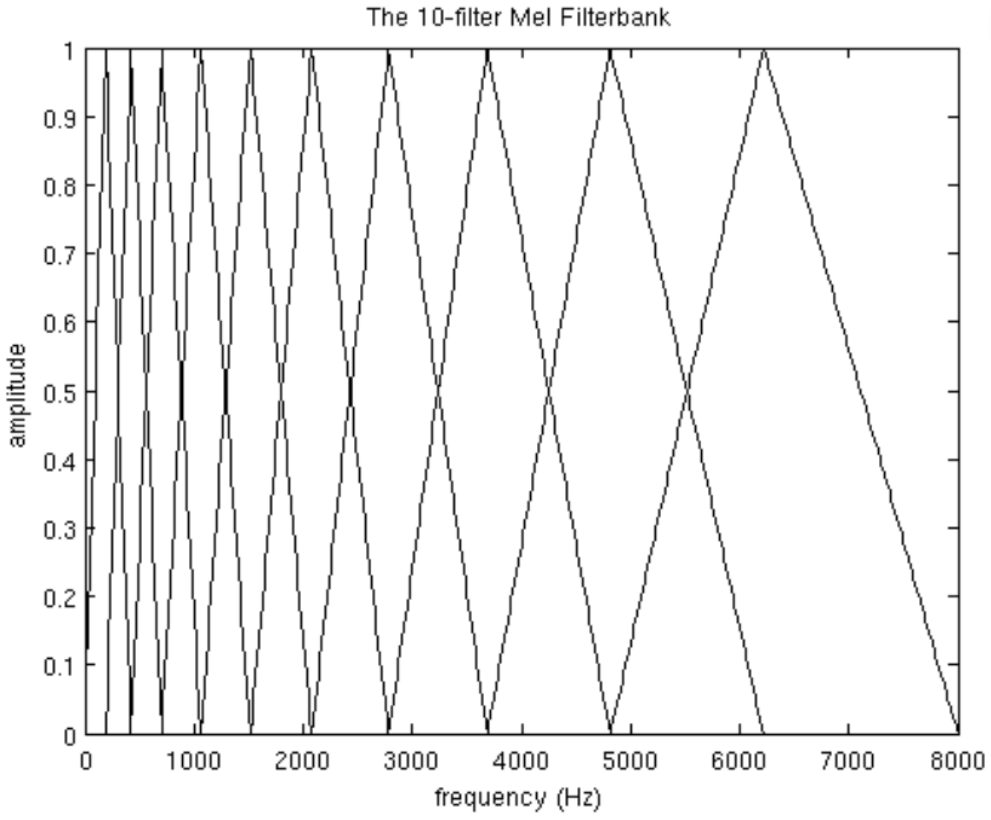
Fast Fourier Transform merupakan algoritma fourier yang digunakan untuk menghitung transformasi fourier pada sinyal digital sehingga dapat mengubah sinyal waktu menjadi sinyal frekuensi. Berdasarkan Bracewell, R. (2000), *Fast Fourier Transform* sendiri merupakan penyempurnaan dari penghitungan *Discrete Fourier Transform* (DFT) yang berhasil mengurangi waktu komputasi dari O(n^2) menjadi O(n log n), dimana n merupakan panjang dari sinyal input. DFT sendiri memiliki rumus yang bisa dilihat pada rumus (2.6)

$$x_k = \sum_{N=0}^{N-1} x_n e^{\pi i k n / N} \tag{2.6}$$

Dimana x_k merupakan hasil transformasi DFT frekuensi ke k, dan x_n adalah sinyal ke n dari sinyal input. Terdapat juga beberapa variasi FFT seperti *Cooley-Tukey Algorithm* dan juga *the Bluestein Algorithm*.

2.3.4 *Mel Filterbank*

Mel filterbank merupakan tahapan lanjutan setelah FFT yang akan memetakan sinyal ke dalam domain frekuensi yang lebih sesuai dengan sistem pendengaran manusia. Hal ini dilakukan menggunakan beberapa *filterbank* yang didesain sesuai dengan skala frekuensi atas respon dari sistem pendengaran manusia. Jumlah *filterbank* yang biasa dipakai untuk MFCC ada di kisaran 20-40 *filterbank.* Dimana untuk proses peletakan *filterbank* akan diletakkan pada frekuensi-frekuensi tertentu yang mempunyai jarak sama pada *Mel-Scale*nya. Visualisasi dari *Mel Filterbank* yang berisi 10 *filterbank*, dimulai dari frekuensi 0 Hz hingga 8 kHz bisa dilihat pada Gambar 2.3.



Gambar 2.3 Visualisasi *Mel Filterbank* dengan 10 *Filterbank* Dengan 8 kHz Frekuensi.

Sumber: Practical Cryptography. (n.d.). Mel Frequency Cepstral Coefficient (MFCC) tutorial. Retrieved March 29, 2023,

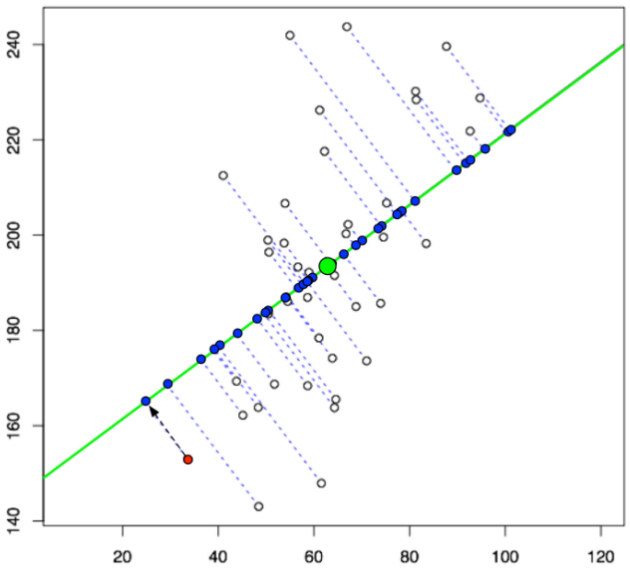
From http://practicalcryptography.com/media/miscellaneous/files/10_filt_melfb.png

2.3.5 *Discrete Cosine Transform* (DCT)

DCT merupakan transformasi yang mirip dengan DFT, namun hanya menggunakan *real numbers,* tidak seperti DFT yang terdapat bilang kompleks (Prabakaran & Sriuppili, 2021). Pada Tahapan ini, hasil dari *Mel filterbank* akan terlebih dahulu di aplikasikan pada *power spectrum* yang didapat dari tahap FFT, kemudian akan diubah menjadi skala *logarithmic* dengan sebutan *log Mel-Spectrogram*. Kemudian *log Mel-spectrogram* akan ditransformasi menggunakan DCT untuk mengubah sinyal diskrit domain waktu menjadi sinyal diskrit dalam domain frekuensi. Dengan melakukan DCT, maka koefisien yang didapat hanyalah koefisien yang memiliki informasi yang penting dan akan membuang koefisien yang kurang penting, sehingga dapat mengurangi waktu komputasi sehingga lebih efisien. Setelah melakukan DCT, maka koefisien yang didapat merupakan koefisien akhir dari *Mels-frequency Cepstral* atau biasa disebut MFCCs (*Mels-frequency Cepstral Coefficients*).

2.4 *Principal Component Analysis* (PCA)

PCA merupakan metode unsupervised yang sering digunakan dalam mengidentifikasi sebuah pattern karena dapat melakukan dimension reduction. PCA sangat berguna untuk melakukan data dimension reduction pada dataset yang besar, dimana tidak banyak informasi yang hilang pada prosesnya (Jolliffe & Cadima, 2016). PCA bekerja dengan cara menemukan *principal components* dari data yang memiliki varians maksimal , dimana *principal components* akan saling orthogonal satu sama lain dan memberikan arahan dimanakah data tersebar. PCA dituliskan sebagai transformasi linear yang memproyeksikan data menjadi sistem koordinat yang baru dimana sumbu sebagai *Principal Component* (Kherif & Latypova, 2020). Visualisasi dari data 2 dimensi pada PCA dapat lihat pada Gambar 2.4



Gambar 2.4 Visualisasi PCA pada data 2 dimensi

Sumber: Programmathically.(2022). Principal Components Analysis Explained for Dummies. Retrieved March 29, 2023,

From https://programmathically.com/principal-components-analysis-explained-for-dummies/

Berdasarkan (Kherif & Latypova, 2020), transformasi didapatkan dengan cara menghitung *eigenvector* dan *eigenvalues* dari kovarians data, dimana *eigenvector* sesuai dengan *Principal Components* dan *eigenvalues* menunjukan varians dari tiap-tiap *Principal Component*. Kemudian *Principal Components* diurutkan berdasarkan *eigenvalues*-nya, sehingga data dapat diproyeksikan pada dimensi yang diperkecil dengan memilih nPCs tertinggi dengan *eigenvalues* terbesar. nPCs inilah yang akan menjadi representasi data yang baru sebanyak yang dibutuhkan.

2.5 *Deep Learning*

Deep learning merupakan cabang dari *machine learning* yang menggunakan *neural network* dengan banyak lapisan untuk mempelajari dan merepresentasikan pola yang kompleks dalam data. *Deep learning* telah digunakan pada berbagai bidang seperti *computer vision, medical application, intelligent transportation system, Natural Language Processing,* dan *speech recognition* (Dong et al., 2021). Bahkan berdasarkan (Alzubaidi et al., 2021), penggunaan *Deep learning* dalam bidang *medical application* dapat membantu dalam diagnosa penyakit dan dalam mengembangkan obat baru. Salah satu arsitektur *deep learning* yang populer adalah *Convolutional Neural Network*, yang menunjukan performa bagus pada banyak

proyek *computer vision* (LeCun et al., 2015). Arsitektur populer lainnya yaitu *Recurrent Neural Network* (RNN), banyak digunakan dalam proyek NLP dan juga *Speech recognition* karena mampu untuk menyimpan ketergantungan temporal (Graves et al., 2013).

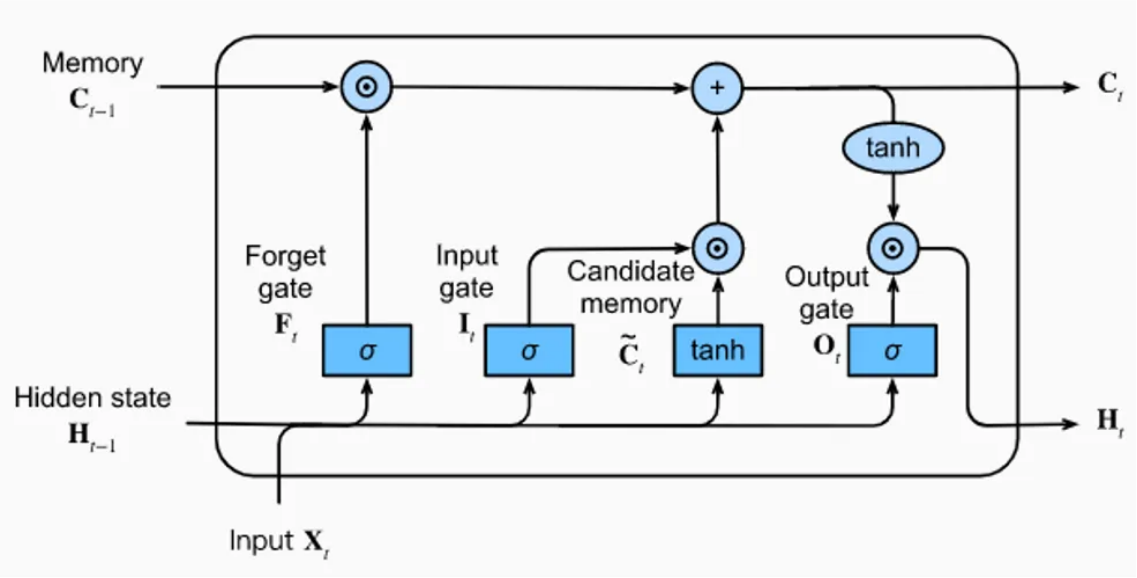
Deep learning mampu untuk mempelajari secara otomatis dan mengekstrak fitur dari data mentah tanpa perlu untuk menggunakan fitur yang dibuat secara manual. Hal ini dicapai dengan menggunakan *Multiple layers of nonlinear transformation*, dimana setiap lapisan belajar representasi data yang lebih abstrak.

Aspek penting dari *deep learning adalah* jumlah data dalam skala besar yang akan digunakan untuk *training* dari model *deep learning*. Selain itu juga dengan ketersedian *computing resources* seperti *Graphic Processing Units* (GPU), dapat mempercepat waktu *training* dari model *deep learning*. *Deep learning* sendiri masih merupakan bidang penelitian yang terbilang baru dan masih akan diteliti lebih jauh untuk diaplikasikan pada bidang-bidang lainnya.

2.6 Long Short Term Memory (LSTM) dan Bi-LSTM

LSTM merupakan model *Neural Network* yang dikembangkan setelah RNN, RNN merupakan sebuah model neural network, *recurrent* yang berarti pada strukturnya ada pengulangan. Output yang dihasilkan terpengaruh dari kondisi awal dan juga kondisi sebelumnya. RNN ini sering digunakan pada data *time series* ataupun data yang mempunyai ketergantungan pada tiap-tiap data yang bersifat sekuensial (Rezk et al., 2020).

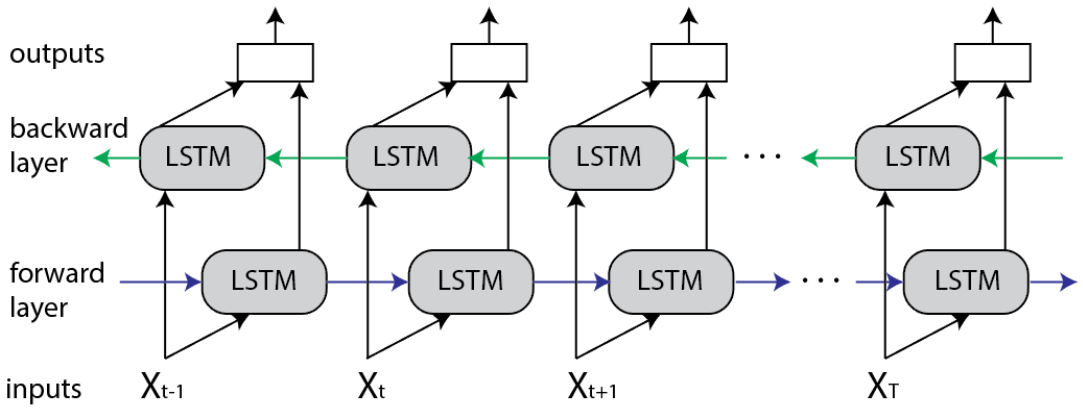
Namun terdapat kekurangan dalam RNN, yaitu adanya permasalahan bernama *Vanishing Gradient Problem*, dimana LSTM menyelesaikan permasalahan tersebut (Sherstinsky, 2020). *Vanishing Gradient Problem* ini disebabkan oleh permasalahan *backpropagation through time*, yaitu kondisi dimana *state* paling jauh dari saat awal mempunyai *gradient error* dengan nilai mendekati nol, hal ini menyebabkan parameter pada *state-state* akhir tidak mengalami perbaikan dengan benar. LSTM memberikan solusi untuk *Vanishing Gradient Problem* dengan memperkenalkan *constant error flow* melalui *internal states* dari *special cells* (Staudemeyer & Morris, 2019). LSTM terdiri dari *memory cell* dan *gates unit*, dengan 3 *gate* utama yaitu: *Forget gate, input gate, output gate* yang bisa dilihat pada Gambar 2.5.



Gambar 2.5 Arsitektur dari Unit LSTM

Sumber: Medium. (2022.). An Intuitive Explanation of LSTM. Retrieved March 29, 2023, From https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c

Bidirectional-LSTM (Bi-LSTM) merupakan model yang dikembangkan dari model tradisional LSTM, dimana proses yang seharusnya berjalan satu arah, akan berjalan dua arah seperti pada Gambar 2.6. Bi-LSTM menambahkan 1 *layer* LSTM dimana akan membalik arah dari informasi. Kemudian akan informasi akan dikumpulkan dari hasil *output* dari *layer* LSTM yang ada.



Gambar 2.6 Arsitektur dari Bi-LSTM

Sumber: DagsHub. (2023.). Complete Guide to RNN, LSTM, and Bidirectional LSTM. Retrieved October 29, 2023, From https://dagshub.com/blog/rnn-lstm-bidirectional-lstm

2.7 Connectionist Temporal Classification (CTC)

CTC merupakan sebuah arsitektur *neural network* klasifikasi untuk melabelkan data berurutan / sekuensial menggunakan RNN yang tidak memerlukan training data yang telah disegmentasikan terlebih dahulu (pre-segmented training data), post-processed outputs, dan memodelkan keseluruhan aspek dari sekuens dalam 1 arsitektur jaringan. Menurut (Wang et al., 2019) keuntungan dalam menggunakan CTC adalah dapat menghilangkan kebutuhan untuk mensegmentasikan data terlebih dahulu, sehingga RNN bisa langsung memproses data secara langsung.

Ide dasar dari CTC adalah untuk menginterpretasikan hasil output dari jaringan sebagai distribusi probabilitas dari semua kemungkinan sekuensial label, berdasarkan dari input sequence (Li, 2022). dengan begitu objective function dapat diturunkan dan memaksimalkan probabilitas dari pelabelan yang benar. CTC di *train* untuk memprediksi distribusi probabilitas dari semua kemungkinan *output sequence*, termasuk label berulang dan juga label kosong yang bisa keluar kapan saja dalam *output sequence*.

Selama model di *train*, CTC *loss function* akan digunakan untuk mengukur perbedaan antara distribusi probabilitas dengan sekuens dari label sesungguhnya. *Loss function* menghitung total dari probabilitas dari seluruh sekuens input terhadap sekuens output, termasuk label perulangan dan label kosong. Kemudian nilai total tersebut akan dipakai untuk meng-*update weight* dari model.

Ringkasnya, CTC merupakan arsitektur *neural network* yang memungkinkan output secara langsung dari sekuens label dari panjang data. CTC sering digunakan dalam Speech Recognition karena memiliki performa bagus (Graves et al., 2006).

2.8 N-Gram

N-Gram merupakan sebuah *language model* yang sangat sering dipakai untuk *Natural Language Processing*, dimana N-Gram memperkirakan probabilitas dari sebuah kata berdasarkan kata sebelumnya atau bahkan lebih. Aplikasi paling umum dari N-Gram dapat dilihat pada kebanyakan *search engines* yang menawarkan fitur *auto-completion* berdasarkan kata yang ditulis, dan tentu dapat mengurangi kesalahan *spelling error* (Avasthi et al., 2021).

N-Gram digunakan untuk memodelkan distribusi probabilitas *word sequences* berdasarkan *acoustic features* dari sinyal suara. Sebagai contoh, model bigram digunakan untuk memperkirakan probabilitas sebuah kata berdasarkan kata sebelumnya, sedangkan model

trigram digunakan untuk memperkirakan probabilitas dari sebuah kata berdasarkan 2 kata sebelumnya.

2.9 *Context Based Learning* (CBL)

CBL merupakan metodologi pembelajaran yang merujuk pada lingkungan dari pada proses belajar. Berdasarkan (Rose, 2012) pendekatan dari CBL berasal dari kepercayaan bahwa belajar merupakan kegiatan sosial yang terpengaruh dari kondisi ruang kelas, sehingga kondisi tersebut dapat mempengaruhi proses berpikir, memproduksi, serta keberhasilan dari belajar. CBL ini dapat dilakukan dengan cara membuat skenario, mereplikasi kondisi belajar. Kemudian CBL bisa disebut sebagai metode yang baik dikarenakan proses belajar dapat berjalan dengan sangat praktikal dengan lingkungan belajar.

CBL memberikan pendekatan yang unik dalam pembelajaran karena memprioritaskan konteks dari setiap pembelajaran yang dilakukan, sehingga dalam *deep learning*, model mendapatkan model *training* berdasarkan situasi / konteks dari data yang akan dikelola. Konsep CBL dalam deep learning sendiri masih terbilang cukup baru, namun sudah memberikan hasil yang cukup efektif di bidang *computer vision* yang dilakukan pada penelitian (Mandal, R. et al., 2021).