Bab 3

3.1 Pengumpulan Data

Data dikoleksi dengan dua cara, dataset pertama diambil dari rekaman langsung melalui *smartphone* dengan konfigurasi 16kHz dan *mono channel*. Data rekaman suara dikondisikan tanpa ada gangguan / *noise*. Data sudah terkumpul sebanyak 105 kalimat dari 5 orang dengan total waktu suara 7 menit. Dataset kedua diambil dari rekaman luar seperti radio, video *Youtube*, suara diseleksi dengan memilih bagian - bagian suara yang bersih tanpa ada gangguan / *noise*. Dataset yang sudah terkumpul sebanyak 4 orang sumber dengan total waktu suara 7 menit. Dataset asli kedua memiliki konfigurasi standar yaitu 44.1kHz sehingga dilakukan konversi dengan menggunakan Audacity menjadi 16kHz. Dataset yang terkumpul akan dibagi menjadi proporsi training 80%, validation 10%, testing 10%.

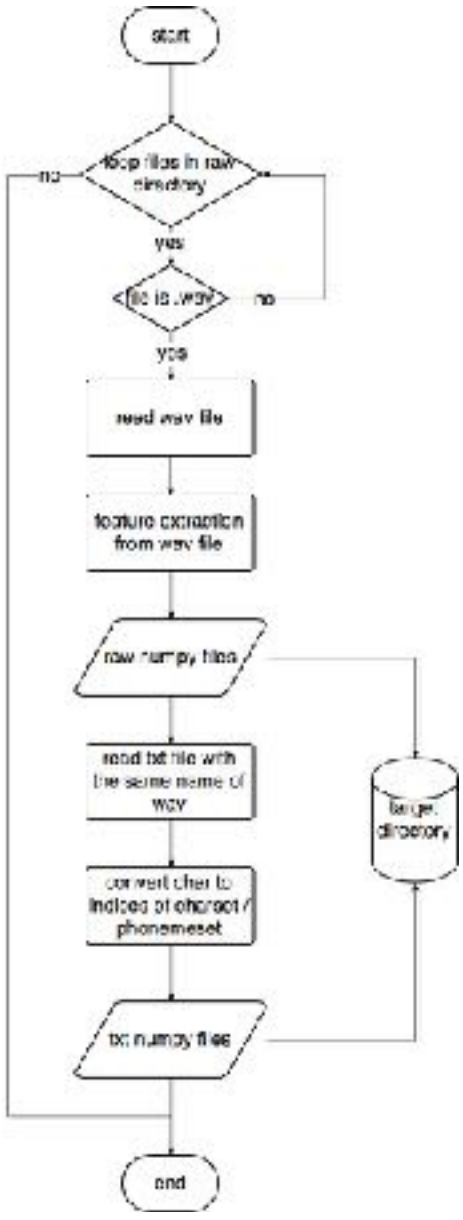
3.2 Representasi Data

Dataset disimpan dengan kode unik nama dengan dua ekstensi *file* berbeda .WAV dan .TXT, dimana .WAV adalah dataset suara, .TXT adalah target label suara. Charset / set karater yang akan menjadi target label suara dapat berupa grafem / fonem, dimana semua .TXT harus konsisten mengikuti charset / set karakter. Karena pemodelan grafem sama seperti huruf biasa sehingga charset / set karakter adalah {a,b,c,d…,z,SPACE}, namun untuk pemodelan fonem tidak bisa mengikuti huruf latin biasa, ada beberapa fonem yang disuarakan berbeda dengan penulisan huruf latin-nya, misal fonem <ng> dimisalkan ŋ, <ny> dimisalkan ñ, fonem <e> dapat berbunyi dua macam yaitu kata menang di simbolkan mənaŋ, sate disimbolkan sate.

3.4 Preprocessing Data

Dataset pertama memiliki perbedaan channel dengan dataset kedua. Pengambilan fitur / ekstraksi fitur tidak dapat dilakukan ketika dataset pertama dan kedua digabungkan, sehingga perlu dilakukan pengambilan 1 kolom

saja untuk dataset suara yang *stereo channel* sehingga dihasilkan keduanya mono channel. Proses preprocessing data juga melakukan konversi data dari .WAV menjadi .NPY *(numpy file),* agar memudahkan saat training, validation, dan testing. *Numpy file* berisi *file* biner yang menyimpan nilai matriks hasil dari ekstraksi fitur. Data yang dihasilkan ada dua *file* dengan format berikut, yaitu <filename>.NPY dan _<filename>.NPY. Karakter <_> pada awal nama *file* menunjukan sebagai dataset target label suara.



Gambar 3.4.a flowchart / diagram alur mekanisme preprocessing data yang dilakukan

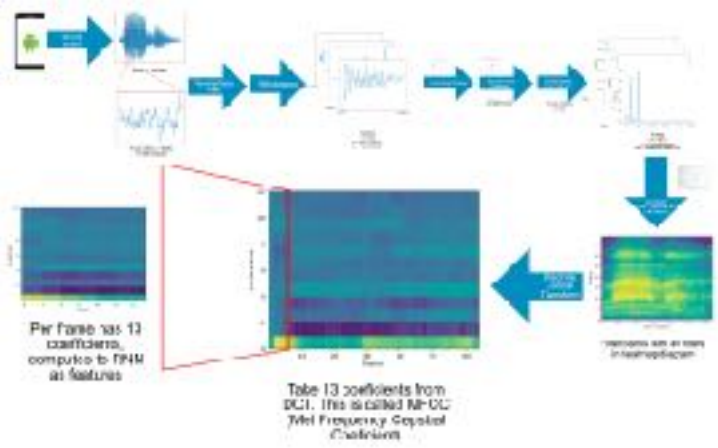
Flowchart / diagram alur diatas dimulai dengan memberikan argumen sebagai berikut :

1. *Raw directory*, sebagai tempat *directory* / *folder* yang akan dibaca dan berisi file .wav dan .txt.
2. *Target directory*, sebagai tempat tujuan / *target directory* / *folder* yang akan dituju ketika menuliskan hasil outputnya berupa .npy *(numpy file)*.
3. *Feature type*, definisi tipe fitur yang akan di ekstrak, misalnya *mfcc* / *spectrogram*.

Sehingga ketika dijalankan, proses ini akan secara otomatis membaca semua file yang ada di *raw directory* dengan memfilter .wav saja dengan catatan untuk .txt file harus memiliki nama yang sama dengan .wav. Maka akan dihasilkan dua file output ber-ekstensikan .npy *(numpy file)*.

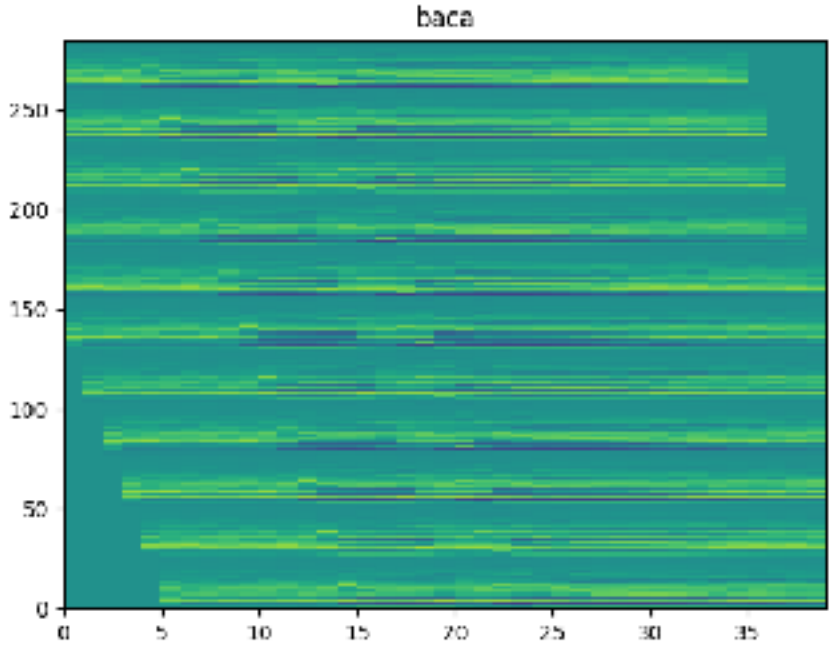
3.5 Pengambilan Fitur

Pengambilan fitur / ekstraksi fitur yang digunakan adalah MFCC (Mel-Frequency Cepstral Coefficients). MFCC adalah koefisien dari koleksi yang dihasilkan oleh MFC (Mel-Frequency Cepstrum). Jumlah koefisien yang digunakan adalah 13. Ukuran *frame* yang digunakan setiap waktunya adalah 0.002 detik / 20 milidetik dengan *striding* / pergeseran sebanyak 0.001 detik / 10 milidetik. Berikut *flowchart* / diagram pengambilan fitur MFCC dari WAV *audio file :*



Gambar 3.5.a flowchart / diagram alur pengambilan fitur MFCC dari WAV audio file

Data yang dihasilkan berupa data matriks / *array* 2 dimensi dengan asumsi baris adalah banyak frame *T (time),* kolom adalah jumlah koefisien *(depth)*. MFCC akan dieksten / diperpanjang dengan menambah n frame sesudah dan sebelumnya pada kolom matriks. Sehingga dihasilkan data MFCC yang memiliki fitur n frame sebelum dan sesudahnya *(past-future context)*. Misalkan $n = 5$ dengan banyak frame $T = 1000$, sehingga pada frame $t \geq 5$ akan ditambahkan $0 \leq t \leq 4$ dan $6 \leq t \leq 10$ pada kolom matriks frame $t = 5$, jika $t - n < 0$ atau $t + n > T - 1$ maka akan diisi dengan matriks 0 dengan ukuran yang tetap sama.



Gambar 3.5.b contoh MFCC past-future context dengan mengambil n = 5

3.6 Model Neural Network

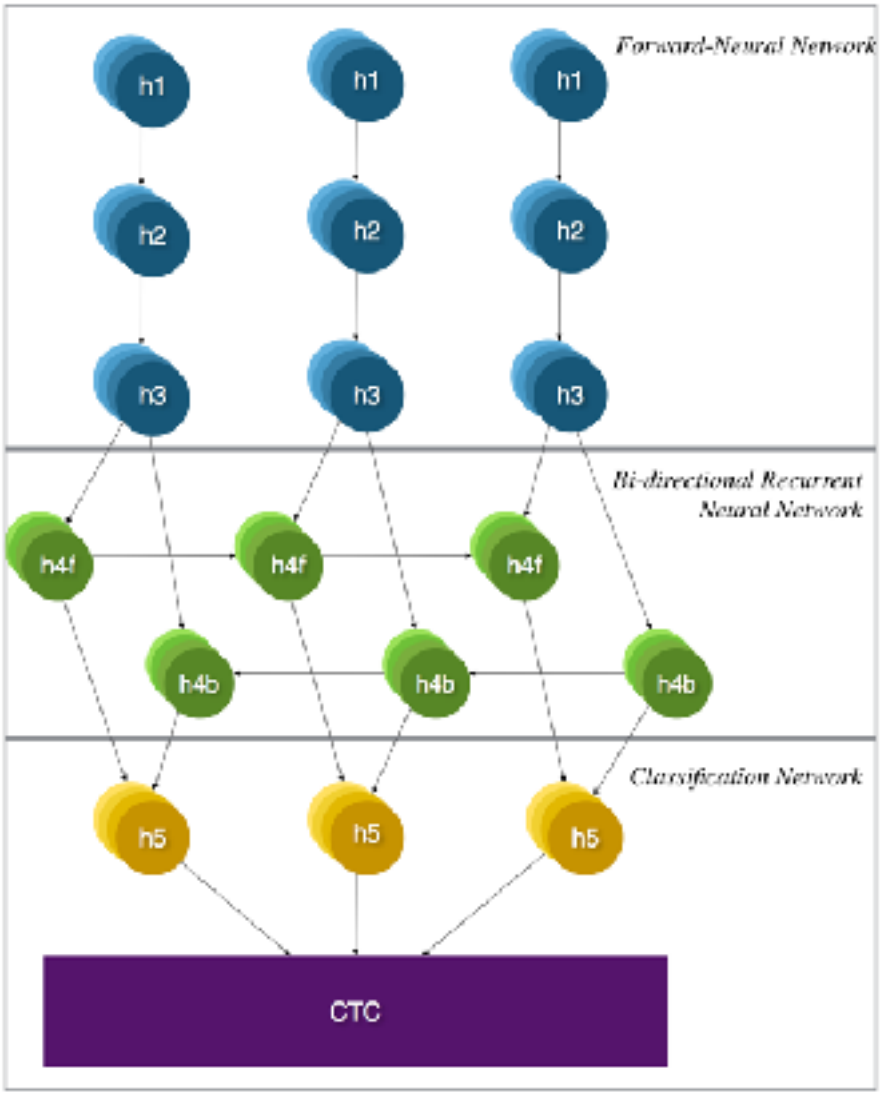
Dataset $x = \{x_{t=0}, x_{t=1}, .. x_T\}$ adalah fitur matriks yang dihasilkan oleh tahap preprocessing data. Konsep sederhana dari *neural network* adalah memasukan inputan dataset *x* ke dalam 3 layer, yaitu input, hidden, dan output. Dimana semua layer diberikan *weight*, *bias* dan *activation function*. Secara umum dapat dirumuskan secara matematika sebagai berikut :

$$h_i = \sum_{t=0}^{T} f(x_t * w_i + b_i)$$

Pada model *neural network* digunakan *random weight* dan *bias* mengikuti *gauss* / *normal distribution* dengan *activation function ReLU (Rectifier Linear Unit) :*

$$f(x) = \begin{cases} x, & if\ x \geq 0 \\ 0, & otherwise \end{cases}$$

Layer untuk neural network dibagi 3 bagian besar, yaitu *Forward-Neural Network (hidden layer), Bi-directional Recurrent Neural Network (hidden layer), Classification Network (output layer):*



Gambar 3.6.a flowchart / diagram alur model network oleh Baidu Research

Pada layer *Forward-Neural Network*, dilakukan *fully connected neural network* dengan *activation function* ReLU. Dengan jumlah hidden neuron yang sama pada masing - masing *network*. Pada layer ini dapat dinotasikan rumus sebagai berikut :

Pada layer *Bi-directional Recurrent Neural Network* digunakan GRUCell (Gated-Recurrent Unit Cell) untuk setiap network *forward* dan *backward*-nya.

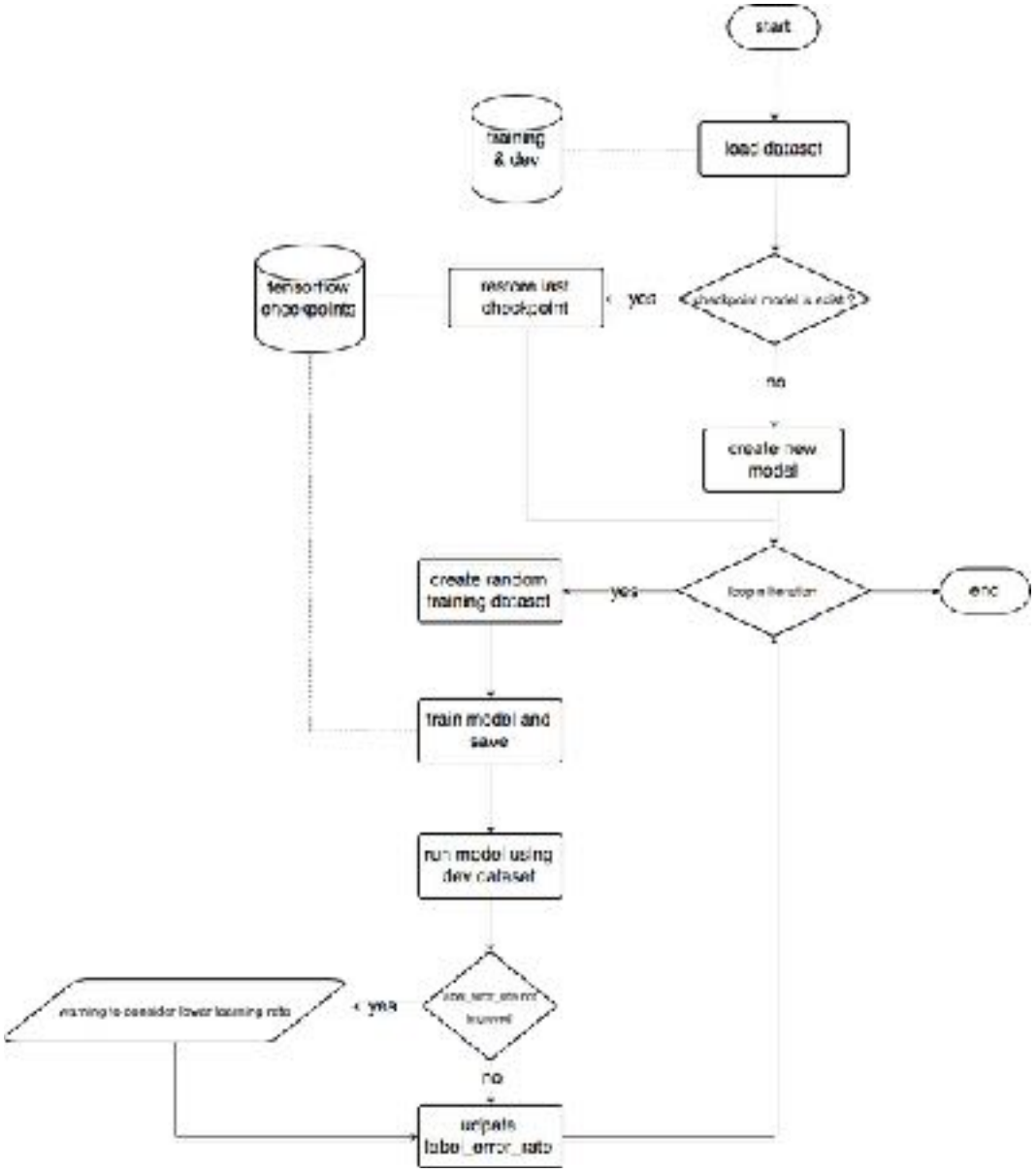
GRUCell dilakukan pada saat layer ini bertujuan untuk mencegah vanishing problem.

Pada layer *Classification Network*, dilakukan *Softmax function* sebagai mapping dari semua network terhadap target k karakter (grafem / fonem) yang diinginkan.

Hasil dari *Softmax function* adalah distribusi probabilitas dari network terhadap k karakter (grafem / fonem). Hasil ini disebut juga dengan *logit*. *Logit* ini digunakan sebagai input pada CTC *(Connectionist Temporal Classification)* untuk memapping-kan urutan karakter tanpa perlu memperhatikan segmentasi data.

3.7 Desain sistem

Sistem dibagi menjadi dua bagian, sistem *training* dan sistem implementasi. Pada sistem *training*, dilakukan proses berulang dengan N iterasi untuk dilakukan backpropagation sehingga didapatkan perubahan weight dan bias yang optimum. Backpropagation yang digunakan adalah Adam Optimizer. Untuk data training dibagi menjadi 2 bagian, *training* dan *dev* dataset. Dengan menggunakan validation untuk dev dataset untuk menghitung performa training model dengan tolak ukur *Edit Distance* / *Label Error Rate*. Flowchart / diagram alur untuk menjelaskan sistem *training* adalah sebagai berikut :



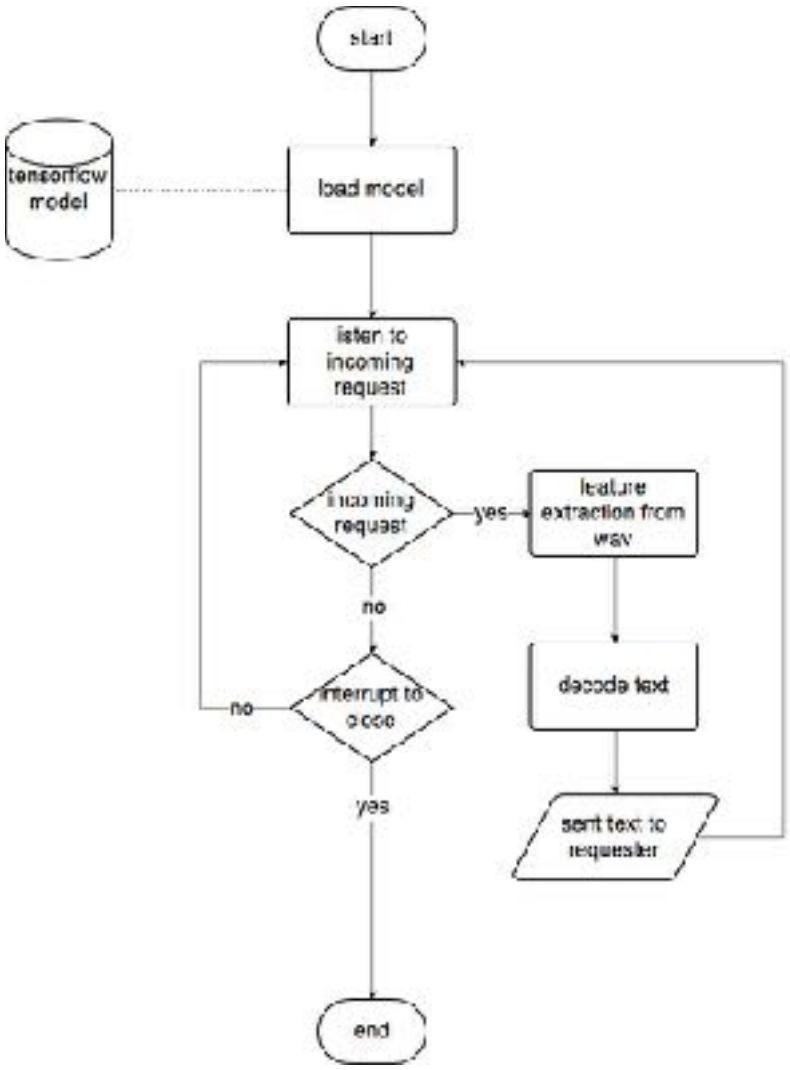
Gambar 3.7.a flowchart / diagram alur training model

Flowchart / diagram alur diatas dimulai dengan memberikan argumen sebagai berikut :

1. *Training directory*, sebagai tempat *directory / folder* berisikan *traning dataset* yang dihasilkan saat preprocessing data.
2. *Dev directory*, sebagai tempat *directory / folder* berisikan *dev dataset* yang dihasilkan saat preprocessing data.
3. *Model directory*, sebagai tempat *directory / folder* yang akan dijadikan tempat penyimpanan hasil model, hasil training, dan *checkpoint* terakhir.

Pada sistem training ini akan dihasilkan sebuah *tensorflow model* yang berisi semua konfigurasi *weight, bias*, dan *graph network* yang telah dibuat saat *training*. *Tensorflow model* ini yang nanti akan digunakan sebagai model untuk menjalankan sistem implementasi di *server*.

Pada sistem implementasi, dilakukan dengan cara mengambil data suara yang direkam melalui *smartphone Android* untuk di kirim ke *server* yang sudah dihasilkan saat sistem *training*. *Server* akan mengolah data .raw audio file yang dikirim oleh *smartphone Android* dan akan menghasilkan prediksi teks untuk ditampilkan di *smartphone Android*. Flowchart / diagram alur yang dilakukan saat sistem implementasi pada *server* adalah sebagai berikut :



Gambar 3.7.b flowchart / diagram alur sistem implementasi pada server