Bab 3

METODE PENELITIAN

3.1 Metode Penelitian

Dalam melakukan penelitian ini, tahapan pertama yang dilakukan adalah pengumpulan data. Tahap ini bertujuan untuk memberikan gambaran tentang tahapan-tahapan apa saja yang akan dilakukan pada proses penelitian. Teknik yang digunakan pada tahapan pengumpulan data adalah studi literatur atau biasa disebut dengan studi pustaka. Studi literatur atau studi pustaka merupakan sebuah proses pengumpulan data dan informasi berupa teori-teori yang berhubungan dengan masalah yang diteliti. Studi literatur dilakukan dengan mencari jurnal ilmiah, paper ilmiah, dan beberapa tesis yang tersebar di internet. Adapun data-data yang dibutuhkan untuk dalam penelitian ini, diantaranya teori tentang CNN, teori dan proses tentang *Beat Tracking*, teori tentang *Chromagram* dan cara mengubah *audio* ke bentuk *chromagram*, teori tentang bahasa pemrograman Python versi 3, dan beberapa implementasi tentang *backend* pada sebuah aplikasi.

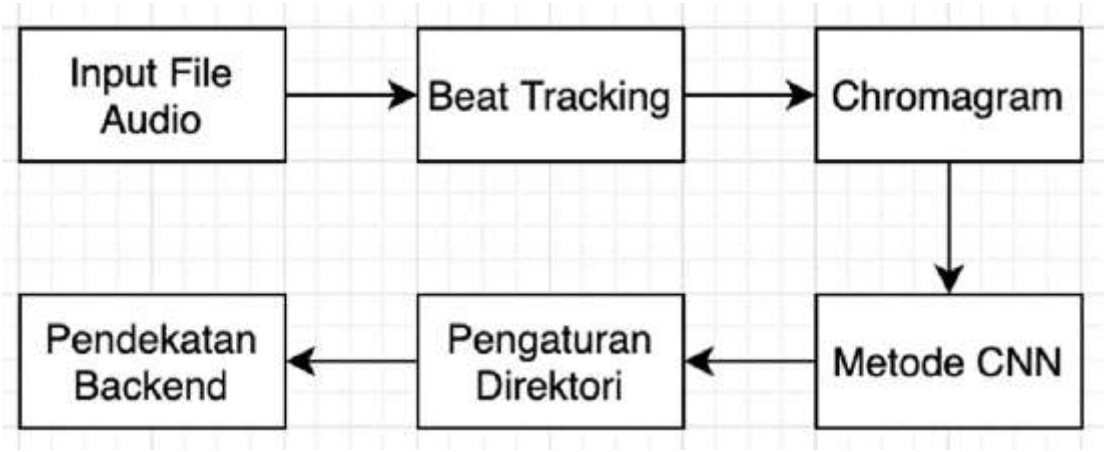
3.1.1 Teori Penelitian

Menurut Kamus Besar Bahasa Indonesia (KBBI), kecerdasan adalah kesempurnaan perkembangan akal budi (seperti kepandaian atau ketajaman pemikiran). Helfi Nasution berpendapat bahwa Kecerdasan Buatan atau dalam bahasa Inggris disebut *Artificial Intelligence* (AI) didefinisikan sebagai kecerdasan yang ditunjukkan oleh suatu entitas buatan (Helfi Nasution, 2012), sedangkan menurut Dedi Nugraha dan Sri Winati (2014), kecerdasan buatan adalah salah satu cabang ilmu pengetahuan yang berhubungan dengan

pemanfaatan mesin untuk memecahkan persoalan yang rumit dengan cara yang lebih manusiawi.

3.2 Kerangka Pemikiran

Kerangka pemikiran dalam penelitian ini memiliki hasil akhir yakni mengetahui akurasi dari metode CNN dalam mengenali sebuah akor dengan memanfaatkan *Chromagram*. Kerangka pemikiran yang akan dilakukan dapat dilihat pada Gambar 3.1.



Gambar 3.1: Kerangka Pemikiran

Langkah awal yang akan dilakukan adalah menerima *input file audio* yang memiliki format .wav. Selanjutnya akan dilakukan *beat tracking*. Pada proses *beat tracking*, *file audio* yang telah diunggah akan dipotong tiap detik dan akan dianalisa bentuk gelombangnya sehingga terbentuklah sebuah onset. Setelah mendapatkan onset dari proses *beat tracking*, selanjutnya onset tersebut akan diubah ke bentuk *chromagram*. Pada proses *chromagram*, onset yang telah didapatkan akan dibentuk ke bentuk gambar. Gambar tersebut nantinya akan dilatih dengan menggunakan metode CNN untuk menemukan akor yang sesuai. Setelah menemukan akor yang sesuai, selanjutnya melakukan proses pengaturan direktori. Proses ini dibutuhkan agar penempatan *parenthesis* dan *child package* bagian *backend* dan *frontend* dapat berkomunikasi satu sama lain. Dan terakhir adalah ada proses pendekatan *backend*. Pendekatan *backend* dibutuhkan agar memudahkan dalam mengakses fungsi yang ada pada pada script Python.

3.2.1 *Audio*

Audio adalah sebuah deret waktu, dimana sumbu y adalah amplitudo arus yang sesuai dengan sebuah membran loudspeaker dan sumbu x adalah sumbu yang sesuai dengan satuan waktu data tersebut. Telinga manusia hanya dapat mendengar bunyi dengan rentang frekuensi antara 20 Hz hingga 20 KHz (20.000Hz). Angka 20 Hz sebagai frekuensi suara terendah yang dapat didengar, sedangkan 20 KHz merupakan frekuensi tertinggi yang dapat didengar. Gelombang suara mengandung sejumlah komponen penting, seperti amplitudo, panjang gelombang, dan frekuensi. Komponen-komponen tersebut mampu membuat suara yang satu berbeda dengan suara yang lain.

Audio yang digunakan dalam penelitian ini adalah audio yang berisi suara piano dan *bass* serta *audio* yang berbentuk format *wav. *Audio* tersebut nantinya akan diolah melalui beberapa tahapan, seperti *Preprocessing* dan *Modeling*. Proses *preprocessing* terbagi atas 2 tahap, yakni *Beat Tracking* dan membentuk gambar *chroma*. Sedangkan untuk proses *modeling* akan langsung dengan menggunakan metode CNN.

3.2.2 *Beat Tracking*

Beat tracking digunakan untuk menentukan contoh waktu dalam rekaman *audio*, di mana pendengar manusia cenderung mengetuk kakinya ke musik. *Beat tracking* pada rancangan menggunakan metode *dynamic programming*. *Dynamic programming* adalah kombinasi pemikiran terstruktur dan pola pikir analitis untuk menyelesaikan sebuah permasalahan. Proses *beat tracking* ini dibutuhkan untuk membentuk ketukan dalam setiap detik dari *file audio* yang diunggah.

3.2.3 *Chromagram*

Chromagram adalah transformasi properti frekuensi-waktu sinyal menjadi *prekursor pitch* yang berubah-ubah untuk sementara waktu. Transformasi ini didasarkan pada pengamatan persepsi tentang sistem pendengaran dan telah terbukti memiliki beberapa sifat matematika yang menarik. *Chromagram* memperluas konsep *chroma* untuk memasukkan dimensi waktu.

Seperti halnya kita menggunakan *spektrogram* untuk menyimpulkan properti tentang distribusi energi sinyal dari frekuensi dan waktu, *chromagram* dapat digunakan untuk menyimpulkan properti tentang distribusi energi sinyal terhadap kroma dan waktu.

3.2.4 *Convolutional Neural Network* (CNN)

Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data *image*. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah *image*. Metode CNN ini digunakan untuk melatih sebuah sistem dalam mengenali akor pada setiap bar.

3.2.5 Pengaturan Direktori

Tujuan dari pengaturan direktori adalah agar penempatan *parenthesis* dan *child package* bagian *backend* dan *frontend* dapat berkomunikasi satu sama lain. Semua *direktori package* dibuat di dalam direktori berjenis Git. Direktori aplikasi ini dapat diunduh melalui link yang terdapat halaman website Git.

Direktori tersebut akan memiliki tiga buah *direktori child* (sub direktori) dengan nama “app”, “app_settings”, dan “backend”. Selain ketiga sub direktori tersebut, terdapat enam buah file yaitu, “.env”, “.flaskenv”, “.gitignore”, “README.md”, “requirements.txt”, dan “run.py”.

Direktori “app” merupakan bagian dari pendekatan *frontend* yang bertujuan untuk menyediakan tampilan kepada pengguna dengan basis web. Sementara itu, direktori “*backend*” merupakan pendekatan *backend* yang bertujuan untuk menyediakan *model* dan *preprocessing* data masukan. Direktori “app_settings” bertujuan untuk melakukan inisialisasi pengaturan Redis Server dan *session id* dengan tujuan agar kedua pendekatan tersebut dapat berkomunikasi satu sama lain.

File “README.md” dibuat dengan tujuan memberikan penjelasan aplikasi secara singkat pada situs Github. File “requirements.txt” berfungsi untuk menuliskan pustaka Python yang dibutuhkan aplikasi ini. File “requirements.txt” dapat digunakan pengguna untuk melakukan pemasangan pustaka yang dibutuhkan aplikasi dengan memasukan perintah “pip install -r requirements.txt” setelah mengunduh direktori aplikasi dari Github. File terakhir yang ada pada direktori “web” adalah “run.py”. File ini berfungsi untuk menjalankan keseluruhan aplikasi melalui perintah “flask run”.

3.2.6 Pendekatan *Backend*

Pendekatan *backend* pada aplikasi ini ditempatkan pada direktori "*backend*" dan dilengkapi dengan file "__init__.py" di dalamnya. Hal ini dilakukan agar direktori "app" milik pendekatan *frontend* dapat mengakses fungsi yang ada pada pendekatan *backend* dengan melakukan perintah *import backend* pada script Python. Model CNN pada aplikasi menggunakan 1655 citra *chromagram* sebagai data latih, dan 347 citra *chromagram* sebagai data validasi.

Pembuatan model menggunakan pustaka Jupyter, Librosa, dan Keras. Data latih berbentuk wav dan mp3 yang dipisah dan dikelompokan setiap dua ketukan menggunakan pustaka Librosa. Setiap ketukan yang terbentuk berupa detik dengan tipe data *float*. Setiap detik ketukan kemudian disimpan ke dalam file berbentuk ".txt". Setelah membentuk file ".txt", dilakukan *labelling* akor pada masing-masing pasangan ketukan yang dibantu oleh Anastashia Aurelia, pelajar piano dari *Associated Board of the Royal School of Music* (ABRSM) yang sudah berada pada Grade 7.

Akor pada aplikasi berjumlah 24 yang terdiri dari 12 akor major dan 12 akor minor. Setelah *labelling* dilakukan, dilakukan visualisasi citra *spectogram* MFCC berdasarkan detik ketukan, dan akor yang ada pada file ".txt". Visualisasi tersebut kemudian diletakkan di dalam direktori masing-masing kelas untuk dijadikan data latih. Sebanyak 20 persen dari jumlah masing-masing kelas dijadikan data validasi. Data latih dimasukan ke dalam direktori "data_train" dan data validasi ke dalam direktori "data_test". Banyak citra data latih dan data validasi masing-masing kelas dapat dilihat pada Tabel 3.1.

Tabel 3.1: Citra Data Latih Dan Data Validasi

Kelas	Data Latih	Data Validasi
A	48	12
A#	50	9
A#m	248	15
Am	84	27
B	61	14
Bm	53	13
C	76	16
C#	44	11
C#m	77	16
D	79	25
D#	57	13
D#m	24	8
Dm	48	12
Em	56	13
F	56	13
F#	67	13
F#m	54	13
Fm	74	17
G	131	17
G#	54	19
G#m	54	13
Gm	75	16
Jumlah	1655	347

Setelah data latih dan data validasi ditempatkan pada direktori dan kelasnya masing-masing, tahap selanjutnya membuat model CNN menggunakan pustaka Keras. Proses pelatihan menggunakan Jupyter Notebook. Model dilakukan proses kompilasi menggunakan *optimizer* RMSprop yang ada pada pustaka Keras dengan nilai *learning rate* sebesar 0.00001. Tahap selanjutnya dalam pembuatan model adalah dengan memasukan *path* direktori data latih dan data validasi ke dalam Keras. Dimensi citra yang digunakan sebagai data masukan ke dalam model memiliki dimensi 200x200 dan berbentuk tiga dimensi (RGB).

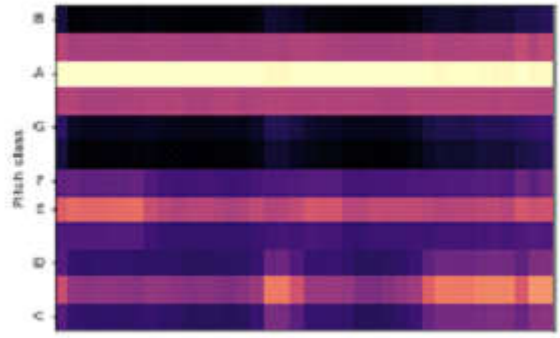
Ukuran *batch* data latih dan data validasi masing-masing adalah sebesar 32 data dan 257 data dengan mode *shuffle* bernilai true, sehingga data latih diproses sebanyak 32 data yang dipilih secara acak oleh Keras dengan jumlah pengulangan sebanyak banyak data latih dibagi besar *batch* yang disebut sebagai *steps per epoch*. Sementara itu, *steps per epoch* yang dipilih untuk data validasi adalah sebanyak dua kali dengan mode *shuffle* bernilai true.

Langkah selanjutnya dalam proses pembuatan model adalah dengan melakukan proses pelatihan. Model yang dilatih disimpan di dalam file berbentuk ".h5" agar dapat digunakan kembali. Proses pelatihan berhasil mendapatkan nilai *accuracy* 87 persen, *loss accuracy* 31 persen, *validation loss* 1.7 persen, dan *validation accuracy* sebanyak 96 persen.

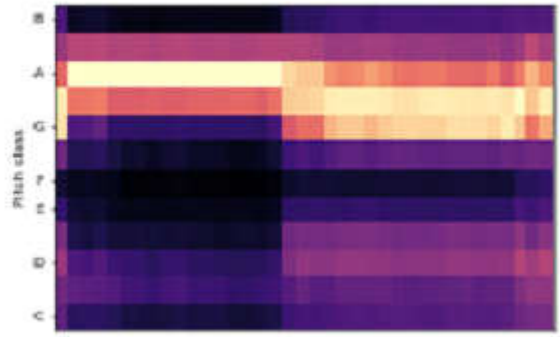
Selanjutnya pengaturan session id. Session id dibentuk pada saat pengguna pertama kali mengunjungi halaman web aplikasi dan juga setelah pengguna menggunakan fitur pengenalan akor. Session id yang dibentuk berguna pada penamaan data masukan pengguna. Penamaan data masukan yang unik berdasarkan session id diperlukan agar data pada saat aplikasi diakses oleh pengguna lebih dari satu tidak saling tumpang tindih.

Redis Server diinisialisasi oleh *backend* terlebih dahulu pada saat aplikasi dijalankan pertama kali. Redis Server pada aplikasi berguna untuk menyimpan session id yang sedang aktif agar dapat diakses baik dari segi *backend* maupun *frontend*. Selain untuk menyimpan session id, Redis Server juga berfungsi sebagai media komunikasi antara *frontend* dengan *backend* pada saat pengguna memilih fitur. Fitur-fitur yang dapat dipilih oleh pengguna adalah memilih data masukan dari direktori pengguna atau melakukan rekaman secara langsung. Perbedaan fitur ini diperlukan suatu variabel pembeda agar fungsi yang dijalankan oleh *backend* tepat sesuai pilihan pengguna.

Model yang sudah dilatih dapat diakses melalui file ".h5" yang sudah disimpan terlebih dahulu pada saat proses pelatihan model, sehingga pada saat pengguna melakukan fitur pengenalan dalam aplikasi, model dapat digunakan kembali. Data masukan pengguna pada saat menggunakan fitur pengenalan pertama-tama dibentuk ".txt" berdasarkan pasangan detik setiap ketukan pada data. Setiap pasangan detik tersebut dibentuk citra *spectogram*. Masing-masing dari citra yang dibentuk kemudian dilakukan proses pengenalan oleh model. Contoh data latih dan data validasi dapat dilihat pada Gambar 3.2 dan Gambar 3.3. Untuk mengetahui akor pada gambar *chromagram*, yang perlu dilihat adalah baris yang paling terang (berwarna putih kekuning-kuningan). Pada data latih, dapat dilihat bahwa akor A yang paling terang dan diikuti oleh akor E dan akor Dm. Setelah dilakukan validasi, ternyata benar bahwa akor yang dilatih adalah akor A. Hal tersebut dapat diketahui dari baris yang paling terang pada data validasi yakni akor A.



Gambar 3.2: Contoh Data Latih Akor A



Gambar 3.3: Contoh Data Validasi Akor A