BAB 3
METODOLOGI PENELITIAN

Metodologi penelitian merupakan metode yang akan digunakan selama kegiatan penelitian berlangsung. Metode penelitian bertujuan agar penelitian yang sedang berlangsung dapat berjalan sesuai dengan yang diharapkan. Metode tersebut berupa studi pustaka, waktu penelitian, metode pengumpulan data, metode pengambilan sampel, metode perancangan, metode evaluasi, metode dokumentasi, metode penelitian menggunakan data Kaggle, metode terjemahan hasil wicara, metode penggunaan aplikasi, metode penelitian menggunakan data pribadi, dan metode menyimpan & melihat model.

3.1 Studi Pustaka

Studi Pustaka merupakan referensi yang akan digunakan selama proses penelitian ini berlangsung. Proses penelitian akan menggunakan referensi dari buku, artikel, jurnal, internet, Mendeley Reference Manager, dan YouTube.

3.2 Tempat dan Waktu Penelitian

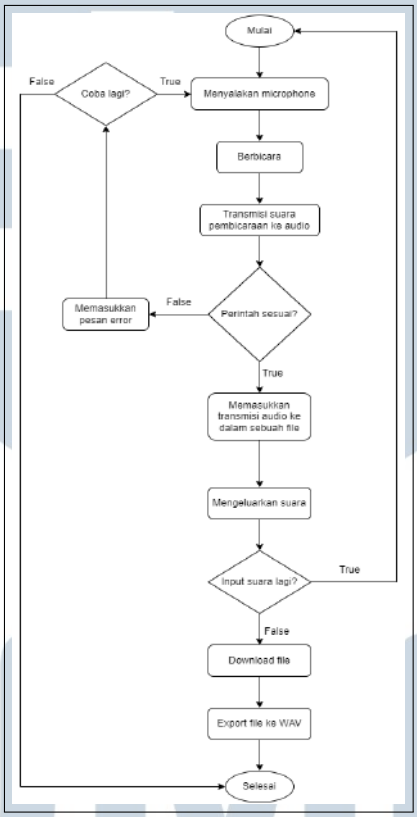
Penelitian yang bertema terjemahan hasil wicara ke teks dilakukan di kampus UMN ruangan laboratorium *Artificial Intelligence* (AI) gedung C lantai lima ruangan 504 kampus UMN dan melalui akses komputer jarak jauh di rumah masing-masing. Penelitian ini berlangsung selama enam bulan mulai bulan April sampai September 2023.

3.3 Metode Pengumpulan Data

Pengumpulan data dilakukan melalui percobaan/ latihan berbicara melalui aplikasi PyCharm. Pengumpulan data dilakukan dengan cara seseorang berbicara dengan maksud untuk memasukkan perintah suara ke audio. Perintah suara yang diucapkan akan mengeluarkan suara pada audio. Jika hasilnya sesuai dengan masukan perintah audio, maka pengumpulan data tersebut sukses. Jika tidak, maka pengumpulan data tersebut gagal. Pengumpulan data yang sukses akan disimpan dalam berkas audio. Percobaan ini dilakukan sebanyak 5.000 sampai 10.000 kali.

3.4 Metode Pengambilan Sampel

Metode pengambilan sampel adalah metode mengambil data yang ada dari pengujian data melalui aplikasi PyCharm. Metode pengambilan sampel yang dilakukan adalah mengambil data suara yang sukses. Sampel yang sukses adalah sampel yang sesuai dengan pembicaraan manusia melalui perintah audio. Proses pengambilan sampel dapat dilihat pada Gambar 3.1.



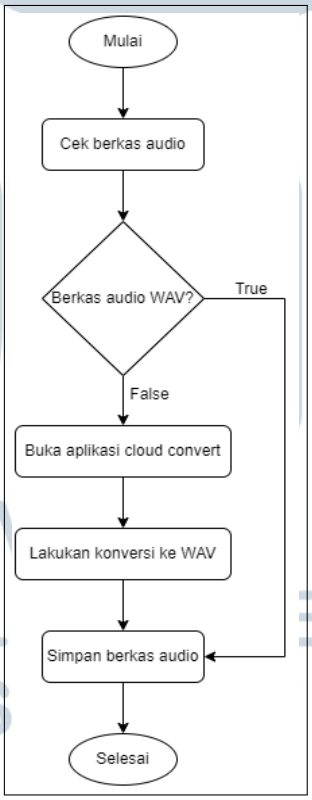
Gambar 3.1. Prosedur translasi hasil wicara ke teks

Pada Gambar 3.1, terlihat rincian prosedur dalam menyampaikan pesan suara ke teks. Prosedur ini merupakan proses awal dalam melakukan proses translasi wicara ke teks. Proses dilakukan menggunakan aplikasi PyCharm. Aplikasi ini akan mengenali suara seseorang dengan menggunakan mikrofon. Mikrofon berguna untuk transmisi suara pembicaraan seseorang ke audio. Suara yang telah dilakukan transmisi melalui audio akan disimpan ke dalam berkas audio. Namun, jika perintah suara yang dimasukkan masih salah atau kurang memuaskan, seseorang dapat melakukan masukan suara ulang. Semua hasil penyampaian pesan ke teks akan diunduh dan disimpan pada berkas audio dengan format WAV.

3.5 Metode Perancangan

Metode perancangan adalah metode yang digunakan pada pembuatan suatu model. Pada awal kegiatan ini, proses mengunduh aplikasi dan mempelajari materi yang berhubungan dengan penelitian ini diperlukan. Proses tersebut ialah mengunduh Python, dan Visual Studio Code, mengunduh PyCharm, mengunduh tensorflow, mengunduh CUDA, dan mengunduh CuDNN. Setelah semua aplikasi diunduh, penelitian translasi wicara ke teks dimulai.

Pada penelitian ini, ada satu aplikasi dan satu situs web utama yang digunakan dalam pembuatan model ini. Aplikasi tersebut adalah PyCharm dengan situs web cloudconvert.com. Aplikasi PyCharm digunakan untuk merancang tensorflow dengan mengunduh tensorflow melalui terminal, serta menerjemahkan pesan suara ke teks dengan menggunakan bahasa pemrograman Python. Situs web cloudconvert.com akan digunakan untuk melakukan konversi berkas audio yang bukan WAV menjadi WAV. Pada penelitian kali ini, ada 5.120 berkas yang memiliki format M4A. Sehingga, proses konversi berkas audio diperlukan, seperti yang terlihat pada Gambar 3.2.

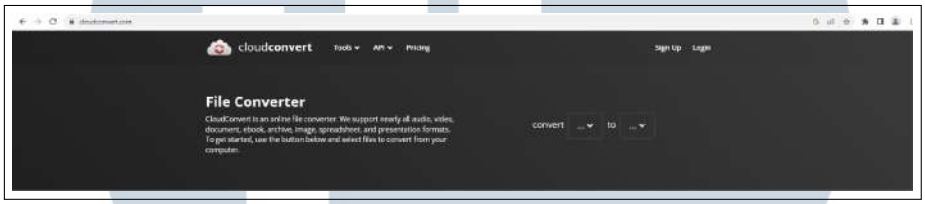


Gambar 3.2. Prosedur konversi berkas audio ke teks

Pada Gambar 3.2, terlihat bahwa dalam merancang suatu model diperlukan

berkas audio yang sesuai. Pada proses perancangan model kali ini, berkas audio yang digunakan harus WAV. Jika berkas audio bukan WAV, maka situs web cloudconvert.com harus digunakan untuk melakukan konversi berkas audio ke teks. Jika berkas audio yang digunakan sudah WAV, maka berkas audio tersebut bisa disimpan dan digunakan dalam proses pembuatan model.

Audio yang memiliki format bukan WAV, seperti yang digunakan dalam proses penelitian ini yaitu M4A, akan dilakukan proses konversi ke WAV. Prosedur konversi dapat dilihat pada Gambar 3.3.



Gambar 3.3. Prosedur konversi berkas audio ke WAV

Pada Gambar 3.3 terlihat bahwa hasil dari konversi berkas audio diperoleh melalui tautan cloudconvert.com. Tautan tersebut mempermudah seseorang dalam melakukan konversi berkas audio ke WAV. Berkas audio yang dilakukan konversi tersebut berupa berkas audio dengan format selain WAV, seperti M4A, MP3, dan MP4.

Pada tautan cloudconvert.com terdapat perintah untuk melakukan konversi berkas audio. Penjelasan lebih rinci dapat dilihat pada Gambar 3.4.

Gambar 3.4. Konversi berkas audio M4A ke WAV

Pada Gambar 3.4 dijelaskan bahwa berkas audio yang dibuat adalah berkas audio dengan format M4A. Berkas audio tersebut tidak bisa dilakukan pengujian data suara ke teks. Berkas audio yang dapat dilakukan pengujian pesan suara ke teks adalah berkas audio yang memiliki format WAV. Sehingga proses konversi berkas audio ke WAV seperti pada Gambar 3.4 harus dilakukan sebanyak jumlah data yang memiliki format M4A yaitu 5.120 sampel data.

3.6 Metode Evaluasi

Evaluasi adalah penilaian suatu aplikasi melalui proses pengujian. Pada penelitian kali ini, proses pengujian dilakukan dengan menjalankan perintah melalui terminal. Terminal yang digunakan adalah terminal yang ada pada aplikasi PyCharm. Pada terminal tersebut, seseorang akan melatih kemampuan berbicara selama satu sampai tiga detik. Hasil dari pembicaraan tersebut akan menampilkan hasil dari pembicaraan yang dilakukan oleh seseorang.

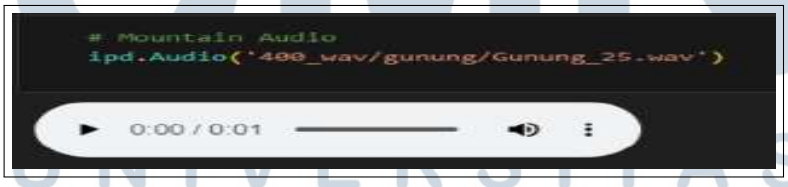
Proses evaluasi yang dilakukan adalah mengambil data audio yang benar. Data audio yang salah tidak akan digunakan. Data audio yang benar akan disimpan ke dalam berkas audio dengan format WAV. Audio tersebut akan dijalankan pada aplikasi Windows Media Player.

3.7 Metode Dokumentasi

Pada penelitian kali ini, semua hasil dari pembicaraan yang dilakukan oleh seseorang akan dibuat dokumentasi dalam satu berkas. Proses dokumentasi ini dilakukan dengan menyimpan semua berkas audio ke dalam satu selebaran yang besar.

3.8 Metode Menampilkan Hasil Wicara

Metode ini adalah metode di mana hasil wicara yang telah dibuat akan ditampilkan. Hasil wicara tersebut berupa pesan suara dalam bentuk audio. Proses penampilan hasil wicara dapat dilihat pada Gambar 3.5.



Gambar 3.5. Gambar untuk melihat hasil wicara

Pada Gambar 3.5, terdapat gambar segitiga berwarna hitam pada bulatan yang berwarna putih. Bulatan tersebut adalah tombol yang digunakan untuk memulai sebuah audio. Pada bagian kanan segitiga tersebut, terdapat durasi sebuah audio. Pada metode ini, durasi audio yang ditampilkan adalah satu detik. Format audio yang digunakan dalam menampilkan hasil wicara ini adalah WAV.

3.9 Metode Penggunaan Aplikasi

Pada penelitian kali ini, ada tiga aplikasi yang digunakan. Aplikasi tersebut berupa PyCharm, Windows Media Player, dan Visual Studio Code. Aplikasi ini akan digunakan untuk mendukung kemampuan berbicara ke teks. Selain itu, ada beberapa aplikasi pendukung yang harus diunduh. Tanpa aplikasi pendukung tersebut, pesan suara dan pengujian model tidak bisa berjalan sesuai dengan fungsinya dan pembicaraan seseorang ke teks menjadi terhambat. Aplikasi lain yang akan digunakan adalah Kaggle, CUDA, CuDNN, Tensorflow, dan Keras.

3.9.1 Visual Studio Code

Pada penelitian kali ini, aplikasi Visual Studio Code akan digunakan dalam membuat kode program dengan bahasa pemrograman Python. Kode ini digunakan untuk pengenalan suara, pengenalan audio, pengenalan mikrofon, dan pengenalan sistem operasi. Kode yang digunakan tersebut akan digunakan dalam pembuatan aplikasi menerjemahkan pesan suara ke teks. Pesan suara tersebut akan diuji pada aplikasi Visual Studio Code.

3.9.2 PyCharm

Pada penelitian ini, PyCharm digunakan untuk membuat kode program dengan bahasa pemrograman Python. Aplikasi ini digunakan untuk membuat kode program pada konteks mengenal suara seseorang melalui gelombang audio. Seseorang dibebaskan untuk berbicara kata apa pun selama beberapa detik dengan total durasi diam selama maksimal dua detik. Penjelasan tersebut dapat dilihat pada Kode 3.1

```
from gtts import gTTS
import speech_recognition as sr
import os

engine = sr.Recognizer()
mic = sr.Microphone()
engine.pause_threshold = 2

print("Selamat datang di aplikasi perekam suara")
with mic as source:
    print("Coba berbicara: ")
    audio = engine.listen(source)
    print("Tunggu sebentar")
```

```
14
15      try:
16          kalimat = engine.recognize_google(audio, language='id')
17          print("Kamu mengatakan: ", kalimat)
18
19          file = gTTS(text=kalimat)
20          cont = input("Apakah Anda mau menyimpan suara Anda(Y/N)?")
21          if (cont == 'Y' or cont == 'y'):
22              simpan = input("Tulis nama file: ")
23              file.save(simpan)
24              print("File tersimpan")
25          elif (cont == 'N' or cont == 'n'):
26              print("Terima kasih Anda telah melatih kemampuan
    berbicara Anda, sampai jumpa")
27          else:
28              print("Maaf input Anda salah")
29      except:
30          print("Maaf suara Anda kurang jelas, mohon coba lagi")
```

Kode 3.1: Aplikasi melatih kemampuan berbicara

Kode 3.1 merupakan kode yang digunakan untuk melatih kemampuan berbicara. Pada baris pertama sampai baris ketiga Kode 3.1, pustaka yang dibutuhkan akan dilakukan inisiasi terlebih dahulu. Setelah itu, pengenalan suara dan mikrofon dilakukan proses inisiasi pada baris kelima dan baris keenam Kode 3.1.

Setelah proses inisiasi dilakukan, waktu seseorang berhenti berbicara bisa diatur, atau bisa dihilangkan. Pada baris ketujuh Kode 3.1, waktu berhenti berbicara diatur selama maksimal dua detik. Sehingga jika seseorang berbicara selama dua detik, lalu berhenti, lalu berbicara lagi, maka pesan suara tersebut masih tetap terdeteksi oleh sistem, asalkan tidak berhenti selama lebih dari dua detik. Namun, jika kode baris ketujuh tersebut dihilangkan, waktu berhenti berbicara seseorang maksimal 0,8 detik.

Kode 3.1 ini bisa dijalankan melalui terminal. Terminal tersebut akan mencetak tulisan mulai dari kode baris kesembilan sampai kode baris ke-17. Pada baris ke-12, kode tersebut mendengarkan tentang apa yang seseorang bicarakan. Jika seseorang berhenti berbicara, waktu berhenti bicara adalah maksimal dua detik, seperti terlihat pada baris ketujuh kode. Jika seseorang telah selesai berbicara atau seseorang diam selama lebih dari dua detik, aplikasi akan menampilkan semua kata yang dibicarakan oleh seseorang seperti terlihat pada kode baris ke-17.

Setelah semua kata ditampilkan, kode baris ke-20 akan tampil pada terminal. Pada perintah yang tampil pada terminal, seseorang bisa menulis berdasarkan perintah. Hasil yang dituliskan akan menampilkan keluaran berdasarkan baris ke-21, baris ke-25, atau baris ke-27 pada Kode 3.1. Kondisi ini sangat tergantung dengan apa yang dikatakan oleh seseorang. Jika seseorang menulis huruf Y atau huruf y, maka berkas audio akan tersimpan. Jika seseorang menulis huruf N atau huruf n, maka berkas audio tidak akan tersimpan. Jika seseorang menulis selain huruf Y, y, N, atau n, maka berkas audio tidak akan tersimpan, dan akan ada keluaran berdasarkan Kode 3.1 baris ke-28. Namun, jika seseorang tidak berbicara selama lebih dari dua detik setelah terminal dijalankan, maka akan muncul keluaran seperti Kode 3.1 baris ke-30.

3.9.3 Windows Media Player

Windows Media Player adalah aplikasi yang digunakan untuk memutar berkas yang memiliki format berupa audio. Pada penelitian kali ini, Windows Media Player digunakan untuk memutar hasil data suara yang direkam. Data suara yang direkam memiliki durasi rata-rata satu sampai tiga detik. Format audio yang diputar adalah WAV.

3.9.4 Tensorflow

Pada penelitian ini, tensorflow digunakan untuk menjalankan program yang memiliki tipe data tensorflow. Tensorflow sangat berguna pada teknik pemisahan dan pengujian data. Data yang memiliki tipe selain tensorflow harus dilakukan proses konversi ke tensorflow. Setelah data tersebut dilakukan proses konversi ke tensorflow, data tersebut dapat dipisahkan. Data yang telah dipisahkan akan dilakukan pengujian.

Pada proses pengujian data, versi tensorflow yang digunakan harus sesuai dengan versi Python yang digunakan. Selain itu, versi CUDA dan CuDNN harus menyesuaikan versi Python dan tensorflow. Pada penelitian ini, versi Python yang digunakan adalah 3.11.3. Versi tensorflow yang digunakan adalah 2.12.0. Versi CUDA yang digunakan adalah 11.8. Versi CuDNN yang digunakan adalah 8.6.

3.10 Himpunan data yang digunakan

Himpunan data adalah kumpulan data, baik data audio, data video, atau pun data dokumen yang jumlahnya banyak. Pada konteks penelitian kali ini, himpunan data yang digunakan adalah himpunan data yang berasal dari internet. Himpunan data tersebut diambil melalui situs web Kaggle. Jumlah data tersebut rata-rata di atas seribu. Himpunan data tersebut akan divisualisasikan satu-persatu dengan metode iterasi. Metode iterasi tersebut berjumlah 40 periode.

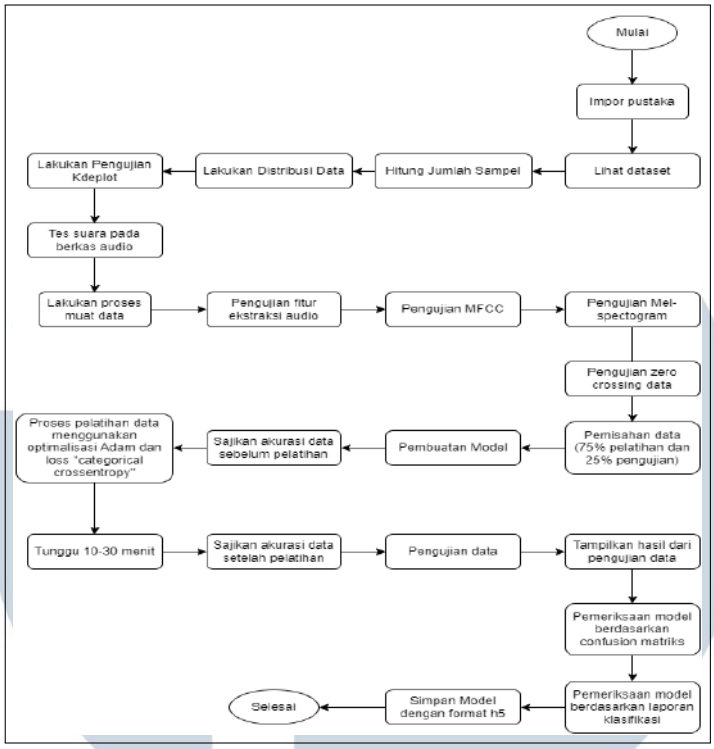
Pada penelitian kali ini, himpunan data pribadi juga digunakan. Hal tersebut berkaitan dengan topik penelitian kemampuan berbicara ke teks. Tujuannya agar seseorang juga dapat melatih kemampuan berbicara sendiri melalui aplikasi PyCharm. Data suara yang dibicarakan akan ditampilkan pada aplikasi PyCharm. Jika data suara tersebut jelas, maka akan keluar hasilnya melalui terminal. Jika tidak, maka akan muncul perintah **"Mohon maaf suara Anda kurang jelas, mohon coba lagi."**. Proses ini akan dilakukan sebanyak berulang-ulang. Jumlah himpunan data yang dihasilkan berada pada kisaran 5.000-10.000 himpunan data. Semua himpunan data tersebut akan dibagi menjadi beberapa sampel. Pada penelitian kali ini, semua himpunan data tersebut akan dibagi menjadi 40 sampel data.

Himpunan data yang digunakan akan diuji dengan algoritma *Convolutional Neural Network* (CNN). Himpunan data yang diuji berupa himpunan data individu/ pribadi. Hasilnya akan diuji tingkat ketepatan. Proses pengujian tersebut akan berlangsung selama 10-30 menit. Kondisi ini bersifat tentatif, tergantung banyak data dan ukuran data tersebut.

3.11 Proses implementasi data Kaggle

Proses implementasi data adalah tahapan yang dilakukan untuk merancang basis data. Proses implementasi data akan menggunakan model pra-pelatihan. Model pra-pelatihan tersebut dilakukan dengan mengambil salah satu model pada Kaggle. Model tersebut berupa model yang digunakan untuk menerjemahkan hasil wicara ke teks.

Model pra-pelatihan dari Kaggle akan dilakukan implementasi. Proses implementasi tersebut akan dilakukan secara bertahap. Tahapan tersebut akan dijabarkan pada Gambar 3.6.



Gambar 3.6. Prosedur Implementasi Data Kaggle

Pada Gambar 3.6, terdapat beberapa tahapan dalam menyajikan data Kaggle. Tahapan tersebut dimulai dari impor pustaka yang diperlukan, pengambilan sampel himpunan data, pengujian data audio, pengujian gelombang audio, pengujian MFCC, pengujian spektogram, pengujian *zero crossing* data, pembagian data untuk model pengujian & pelatihan, pengujian data dalam beberapa periode, pengujian nilai akurasi, pengujian model, dan penyimpanan model. Penjelasan lebih rinci akan dijelaskan pada bagian di bawah ini.

3.11.1 Melihat semua berkas audio

Pada bagian ini terdapat bagian untuk melihat semua daftar berkas yang ada pada himpunan data yang digunakan. Kode tersebut dapat lihat pada Kode 5.1 pada halaman lampiran. Kode tersebut akan menghasilkan keluaran berkas yang terdapat pada himpunan data tersebut bisa dilihat dengan menjalankan Kode 5.1 pada pustaka Python 3.11.3. Hasil keluaran data akan menampilkan semua data yang akan dilakukan pengujian model.

3.11.2 Impor pustaka

Setelah semua berkas audio yang tersedia ditunjukkan, proses impor pustaka akan dilakukan. Skenario dalam impor pustaka dapat dilihat pada Kode 5.2 melalui halaman lampiran.

3.11.3 Proses menggunakan himpunan data pribadi

Pada penelitian ini, pengambilan himpunan data dilakukan dengan cara seseorang berbicara melalui aplikasi PyCharm. Seseorang berbicara sebanyak berkali-kali. Jumlah himpunan data yang digunakan berjumlah 5.120. Setelah itu, himpunan data tersebut dilakukan proses kompresi ke WAV melalui situs web cloudconvert.com dan dijalankan melalui aplikasi Visual Studio Code.

Penulisan pengambilan himpunan data melalui Visual Studio Code dapat dilihat pada Kode 3.2.

```
dataset='400_wav'
pd.DataFrame(os.listdir(dataset),columns=['Files'])
```

Kode 3.2: Mengambil dataset (himpunan data) untuk pengujian model pada Visual Studio Code

Pada Kode 3.2 dijelaskan bahwa pengambilan himpunan data pada baris pertama kode tersebut dengan cara mengambil himpunan data dari Kaggle. Namun, himpunan data dari Kaggle telah disimpan ke dalam suatu penyimpanan pada data komputer. Sehingga penulisan pengambilan himpunan data tersebut dilakukan dengan menulis letak berkas pada halaman yang dituju. Hasil dari pengambilan dapat dilihat berdasarkan nomor dan nama berkas pada Tabel 4.1 pada halaman hasil dan diskusi.

3.11.4 Menghitung jumlah sampel

Setelah melakukan proses mengambil himpunan data pada Visual Studio Code, proses perhitungan jumlah sampel dilakukan. Proses tersebut dapat dilihat pada Kode 3.3.

```
def count(path):
    size=[]
    for file in os.listdir(path):
        size.append(len(os.listdir(os.path.join(path,file))))
    return pd.DataFrame(size,columns=['Number of Sample'],index=os
    .listdir(path))
```

```
tr=count(dataset)
tr
```

Kode 3.3: Menghitung jumlah sampel data

Kode 3.3 merupakan kode yang akan menghasilkan jenis sampel dan jumlah sampel setiap kosakata pada kode baris kelima. Jenis dan jumlah sampel kosakata tersebut dapat dilihat pada Tabel 4.2 pada hasil dan diskusi.

3.11.5 Proses Distribusi Data

Setelah menghitung jumlah sampel, proses selanjutnya adalah distribusi data. Proses tersebut dapat dilihat pada Kode 3.4.

```
plt.figure(figsize=(10,10))
plt.pie(x='Number Of Sample',labels=os.listdir(dataset),autopct ='
    %1.1f%%',data=tr)
plt.title('Distribution Of Data In Train',fontsize=14)
plt.show()
```

Kode 3.4: Distribusi data menjadi beberapa bagian

Pada Kode 3.4 baris keempat dijelaskan bahwa proses distribusi adalah proses membagi himpunan data menjadi beberapa bagian. Proses membagi himpunan data tersebut dilakukan dengan mengelompokkan data dalam satuan %. Proses distribusi data kali ini dilakukan dengan mengelompokkan 40 jenis kosakata data yang berbeda dalam satu diagram lingkaran.

3.11.6 Proses Pengujian Kdeplot

Pada sampel yang diuji, ukuran sampel dapat digambarkan pada pengujian Kdeplot. Penjelasan lebih rinci bisa dilihat pada Kode 3.5.

```
sns.kdeplot(tr['Number Of Sample'], color='red')
plt.title('Kdeplot For Train', fontsize=20)
plt.xlabel('Number', fontsize=15)
plt.ylabel('Count', fontsize=15)
```

Kode 3.5: Menggambarkan sampel dengan grafik

Pada Kode 3.5 dijelaskan bahwa terdapat grafik yang berwarna merah. Grafik tersebut berwarna merah dibuktikan dengan kode program yang terdapat pada baris pertama. Grafik tersebut juga menampilkan angka sumbu x berdasarkan Kode 3.5 baris ketiga, dan jumlah hitungan pada sumbu y berdasarkan Kode 3.5 baris keempat.

3.11.7 Proses Perhitungan Sampel

Proses ini merujuk pada proses perhitungan jumlah sampel data. Proses tersebut bisa dilihat pada Kode 3.6.

```
def load(path):
    data=[]
    label=[]
    sample=[]
    for file in os.listdir(path):
        path_=os.path.join(path,file)
        for fil in os.listdir(path_):
            data_contain,sample_rate=lr.load(os.path.join(path_,
    fil))
            data.append(data_contain[:15360])
            sample.append(sample_rate)
            label.append(file)
    return data,label,sample
```

Kode 3.6: Proses menghitung jumlah sampel mandiri

Berdasarkan Kode 3.6, baris kedua sampai baris keempat merupakan kode yang digunakan untuk pengenalan data, label, dan sampel. Ketiga baris tersebut digunakan untuk mengetahui nomor data, label data, dan *sample rate*. *Sample rate* atau disingkat SR merupakan jumlah sampel per detik saat merekam atau memainkan suara digital. Jumlah sampel per detik akan memengaruhi grafik dalam visualisasi data dan tingkat akurasi data. Pada penelitian kali ini jumlah sampel yang digunakan adalah 5.120. Selain itu, ukuran data juga dipotong berdasarkan jumlah sampel paling pendek. Pada penelitian kali ini, jumlah sampel paling pendek adalah 15.360 dan dilakukan pemotongan pada Kode 3.6 baris kesembilan. Pemotongan dilakukan agar ukuran semua data sama.

Ukuran sampel yang telah dibuat, bisa dilakukan pengujian. Pengujian tersebut tergantung dengan jumlah data yang ada. Pengujian tersebut dilakukan distribusi dalam bentuk tabel dengan kolom label dan sampel. Kode program tersebut dapat dilihat pada Kode 3.7.

```
data,label,sample=load(dataset)
df=pd.DataFrame()
df['Label'],df['sample']=label,sample
df
```

Kode 3.7: Menggambarkan ukuran sampel mandiri

Berdasarkan Kode 3.7, hasil dari distribusi akan dijabarkan dalam bentuk tabel berdasarkan pengujian model yang terdapat pada baris ketiga dan baris

keempat. Hasil dari distribusi tersebut akan dilampirkan pada Tabel 4.3 pada hasil dan diskusi.

3.11.8 Pengujian fitur ekstraksi audio

Setelah proses impor pustaka dan pengecekan suara audio, proses ekstraksi audio dilakukan. Proses pertama yang dilakukan dapat dilihat pada Kode 5.3 di halaman lampiran.

Kode 5.3 menjelaskan tentang fitur ekstraksi audio menggunakan algoritma CNN. Fitur tersebut terdiri dari pengenalan fitur ekstraksi audio, mel-spektogram, MFCC, dan ukuran gelombang audio. Fitur yang dilakukan pengenalan tersebut akan dilakukan implementasi dengan menggunakan kode program. Kode program tersebut dapat dilihat melalui Kode 3.8, Kode 3.9, dan Kode 3.10, Kode 3.11, dan Kode 3.12.

Proses pengujian fitur ekstraksi audio yang pertama kali dilakukan adalah proses pengujian gelombang audio. Proses tersebut dilakukan implementasi berdasarkan Kode 3.8.

```
#waveform data[0]
waveform(data[0],sample[0],label[0])
plt.legend()
```

Kode 3.8: Pengujian gelombang audio mandiri

Pada Kode 3.8 terdapat hasil dari pengujian gelombang audio pada Gambar 4.3 pada hasil dan diskusi. Pengujian gelombang audio kali ini dilakukan dengan menggunakan array nol, sampel data pertama. Setelah dilakukan pengujian gelombang audio, dilakukan proses pengujian MFCC. Proses pengujian MFCC dilakukan untuk mengetahui seberapa bagus sinyal suara yang diperoleh. Proses pengujian MFCC dilakukan pada Kode 3.9.

```
#MFCC data[0]
mfccs_mean, mfccs = mfcc(data[0], sample[0])
print('MFCCs Mean:', mfccs_mean)
print('MFCCs shape:', mfccs.shape)
mfcc_v(mfccs,label[0])
```

Kode 3.9: Pengujian MFCC mandiri

Pada Kode 3.9 diperoleh hasil dari pengujian MFCC berupa nilai rata-rata dari MFCC pada baris ketiga dan ukuran data dari MFCC pada baris keempat. Hasil pengujian MFCC kali ini dilakukan dengan menggunakan array nol, sampel data pertama. Ukuran data dari MFCC pada baris keempat akan dilakukan proses implementasi menggunakan grafik dengan menuliskan Kode 3.10.

```
#MFCC data[0]
mfccs_mean, mfccs = mfcc(data[0], sample[0])

# Create a heatmap of the MFCCs
plt.imshow(mfccs, cmap='viridis', origin='lower', aspect='auto')

# Add a text annotation for the mean value
plt.text(10, 18.5, f'Mean: {mfccs_mean:.6f}', color='black',
    fontsize=10)

plt.colorbar(format='%+2.0f dB')
plt.title('MFCCs')
plt.xlabel('Time Frames')
plt.ylabel('MFCC Coefficients')
plt.show()
```

Kode 3.10: Implementasi pengujian MFCC mandiri

Kode 3.10 akan menyajikan hasil dari data dan sampel pada array nol berdasarkan kode baris kedua. Ukuran MFCC akan menyajikan nilai rata-rata dari sinyal suara pada kode baris kedelapan, interval waktu pada sumbu x berdasarkan kode baris ke-12, dan koefisien dari MFCC pada sumbu y berdasarkan kode baris ke-13.

Setelah proses implementasi pengujian MFCC, dilakukan proses pengujian mel-spektogram. Pengujian mel-spektogram akan dilakukan untuk mengetahui ukuran frekuensi suatu gelombang suara. Proses pengujian tersebut dapat dilihat pada Kode 3.11.

```
##Mel-spectrogram data[0]
mel_mean,mel=Mel(data[0],sample[0])
print('Mel Mean:',mel_mean)
print('Mel :',mel.shape)
mel_v(mel,label[0],sample[0])
```

Kode 3.11: Pengujian mel-spectrogram mandiri

Kode 3.11 akan menyajikan hasil dari data dan sampel pada array nol, himpunan data pertama yang berupa nilai rata-rata pada baris ketiga dan ukuran mel-spektogram pada baris keempat. Ukuran data dari Mel-spektogram pada baris keempat akan dilakukan proses implementasi menggunakan grafik dengan menuliskan Kode 3.12.

```
##Mel-spectrogram data[0]
mel_mean,mel=Mel(data[0],sample[0])

# Create a heatmap of the Mel spectrogram
```

```
plt.imshow(mel, cmap='viridis', origin='lower', aspect='auto')

# Add a text annotation for the mean value
plt.text(10, 120, f'Mean: {mel_mean:.7f}', color='white', fontsize
    =10)

plt.colorbar(format='%+.7f')
plt.title('Mel Spectrogram')
plt.xlabel('Time Frames')
plt.ylabel('Frequency Bins')
plt.show()
```

Kode 3.12: Implementasi pengujian mel-spectrogram mandiri

Kode 3.12 akan menyajikan hasil dari data dan sampel pada array nol berdasarkan kode baris kedua. Ukuran mel-spektogram akan menyajikan rata-rata frekuensi pada kode baris kedelapan, interval waktu pada sumbu x berdasarkan kode baris ke-12, dan rentang frekuensi pada sumbu y berdasarkan kode baris ke-13.

Setelah proses pengujian gelombang audio, MFCC, dan mel-spektogram, dilakukan pengujian *zero crossing* data pada kondisi di mana fungsi menyentuh titik nol. Pengujian *zero crossing* data dilakukan untuk memperoleh amplitudo dari gelombang suara yang dibuat. Penjelasan lebih lanjut bisa dilihat melalui Kode 3.13.

```
#zero_crossing data[0]
zcr=zero_crossing(data[0],sample[0])
print('Zcr:',zcr.shape)
zero_crossing_v(zcr,label[0],data[0],sample[0])
```

Kode 3.13: Pengujian zero-crossing data mandiri

Pada Kode 3.13 diperoleh hasil dari pengujian *zero crossing* data pada Gambar 4.8 pada hasil dan diskusi. Hasil pengujian *zero crossing* data kali ini dilakukan dengan menggunakan array nol, sampel data pertama.

Pada Kode 3.8, Kode 3.9, Kode 3.10, Kode 3.11, Kode 3.12, dan Kode 3.13 ditunjukkan bahwa hasil dari pengujian ini dilakukan untuk melakukan pengujian array nol yang berarti himpunan data pertama dari sampel yang dimasukkan. Pengujian tersebut bisa dilakukan dengan menggunakan himpunan data yang kedua, himpunan data yang ke-1.000, atau pun himpunan data paling terakhir. Pengujian kali ini akan dilakukan dengan menggunakan data array ke-1.000, himpunan data ke-1.001. Kode program tersebut dapat dilihat pada Kode 3.14, Kode 3.15, Kode 3.16, Kode 3.17, Kode 3.18, dan Kode 3.19.

Proses pengujian fitur ekstraksi audio yang pertama kali dilakukan adalah proses pengujian gelombang audio. Proses tersebut dilakukan implementasi

berdasarkan Kode 3.14.

```
#waveform data[1000]
waveform(data[1000],sample[1000],label[1000])
plt.legend()
```

Kode 3.14: Pengujian gelombang audio array 1000

Pada kode 3.14 terdapat hasil dari pengujian gelombang audio pada array ke-1.000 pada Gambar 4.9 pada hasil dan diskusi. Pengujian gelombang audio array ke-1.000 akan menampilkan hasil pengujian gelombang audio pada himpunan data ke-1.001. Himpunan data tersebut dilakukan pengujian dengan menghasilkan himpunan data secara acak.

Setelah dilakukan pengujian gelombang audio, dilakukan proses pengujian MFCC. Proses pengujian MFCC dilakukan pada Kode 3.15.

```
#MFCC data[1000]
mfccs_mean, mfccs = mfcc(data[1000], sample[1000])
print('MFCCs Mean:', mfccs_mean)
print('MFCCs shape:', mfccs.shape)
mfcc_v(mfccs,label[1000])
```

Kode 3.15: Pengujian MFCC array 1000

Pada Kode 3.15 diperoleh hasil dari pengujian MFCC pada array ke-1.000 dengan nilai rata-rata MFCC pada baris ketiga dan ukuran MFCC pada baris keempat. Pengujian MFCC array ke-1.000 akan menampilkan hasil pengujian MFCC pada himpunan data ke-1.001. Himpunan data tersebut dilakukan pengujian dengan menghasilkan himpunan data secara acak. Ukuran data dari MFCC pada baris keempat akan dilakukan proses implementasi menggunakan grafik dengan menuliskan Kode 3.16.

```
#MFCC data[1000]
mfccs_mean, mfccs = mfcc(data[1000], sample[1000])

# Create a heatmap of the MFCCs
plt.imshow(mfccs, cmap='viridis', origin='lower', aspect='auto')

# Add a text annotation for the mean value
plt.text(10, 18.5, f'Mean: {mfccs_mean:.6f}', color='black',
    fontsize=10)

plt.colorbar(format='%+2.0f dB')
plt.title('MFCCs')
plt.xlabel('Time Frames')
plt.ylabel('MFCC Coefficients')
```

```
plt.show()
```

Kode 3.16: Implementasi pengujian MFCC array 1000

Kode 3.16 akan menyajikan hasil dari data dan sampel pada array ke-1.000 berdasarkan kode baris kedua. Ukuran MFCC akan menyajikan nilai rata-rata dari sinyal MFCC pada kode baris kedelapan, interval waktu pada sumbu x berdasarkan kode baris ke-12, dan koefisien dari MFCC pada sumbu y berdasarkan kode baris ke-13.

Setelah dilakukan pengujian MFCC, proses pengujian mel-spektogram dilakukan. Proses pengujian mel-spektogram dapat dilihat pada Kode 3.17.

```
##Mel-spectrogram data[1000]
mel_mean,mel=Mel(data[1000],sample[1000])
print('Mel Mean:',mel_mean)
print('Mel :',mel.shape)
mel_v(mel,label[1000],sample[1000])
```

Kode 3.17: Pengujian mel-spektogram array 1000

Pada Kode 3.17 terdapat hasil dari pengujian mel-spektogram pada array ke-1.000 dengan nilai rata-rata dari mel-spektogram pada baris ketiga dan ukuran dari mel-spektogram pada baris keempat. Pengujian mel-spektogram array ke-1.000 akan menampilkan hasil pengujian mel-spektogram pada himpunan data ke-1.001. Himpunan data tersebut dilakukan pengujian dengan menghasilkan himpunan data secara acak. Ukuran data dari Mel-spektogram pada baris keempat akan dilakukan proses implementasi menggunakan grafik dengan menuliskan Kode 3.18.

```
##Mel-spectrogram data[0]
mel_mean,mel=Mel(data[0],sample[0])

# Create a heatmap of the Mel spectrogram
plt.imshow(mel, cmap='viridis', origin='lower', aspect='auto')

# Add a text annotation for the mean value
plt.text(10, 120, f'Mean: {mel_mean:.7f}', color='white', fontsize
    =10)

plt.colorbar(format='%+.7f')
plt.title('Mel Spectrogram')
plt.xlabel('Time Frames')
plt.ylabel('Frequency Bins')
plt.show()
```

Kode 3.18: Implementasi pengujian mel-spektogram array 1000

Kode 3.18 akan menyajikan hasil dari data dan sampel pada array nol berdasarkan kode baris kedua. Ukuran mel-spektogram akan menyajikan nilai rata-

rata frekuensi pada kode baris kedelapan, interval waktu pada sumbu x berdasarkan kode baris ke-12, dan rentang frekuensi pada sumbu y berdasarkan kode baris ke-13.

Setelah proses pengujian mel-spektogram dilakukan, proses pengujian *zero crossing* data dilakukan pada array ke-1.000. Kode pengujian tersebut dapat dilihat pada Kode 3.19.

```
#zero_crossing data[1000]
zcr=zero_crossing(data[1000],sample[1000])
print('Zcr:',zcr.shape)
zero_crossing_v(zcr,label[1000],data[1000],sample[1000])
```

Kode 3.19: Pengujian zero-crossing data array 1000

Pada Kode 3.19 terdapat hasil dari pengujian *zero crossing* data pada array ke-1.000 pada Gambar 4.14 pada hasil dan diskusi. Pengujian *zero crossing* data array ke-1.000 akan menampilkan hasil pengujian *zero crossing* data pada himpunan data ke-1.001. Himpunan data tersebut dilakukan pengujian dengan menghasilkan himpunan data secara acak.

3.11.9 Pengujian semua himpunan data

Setelah proses pengujian gelombang audio, pengujian MFCC, pengujian mel-spektogram, dan pengujian *zero crossing* data, dilakukan pengujian semua himpunan data dengan penggunaan label. Penjelasan lebih rinci bisa dilihat pada Kode 3.20 dan Kode 3.21.

```
code={}
x=0
for i in pd.unique(label):
    code[i]=x
    x+=1
pd.DataFrame(code.values(),columns=['Value'],index=code.keys())
```

Kode 3.20: Pengujian value data

Kode 3.20 akan menghasilkan keluaran nilai dari data yang dimasukkan berdasarkan kode baris keenam. Hasil keluaran tersebut berupa nomor array yang dilakukan deklarasi menggunakan huruf i dan nomor kosakata yang dilakukan deklarasi menggunakan huruf x.

```
def get_Name(N):
    for x,y in code.items():
        if y==N:
            return x
for i in range(len(label)):
```

```
    label[i]=code[label[i]]
pd.DataFrame(label,columns=['Labels'])
```

Kode 3.21: Pengujian label data

Kode 3.21 akan menghasilkan hasil keluaran pada kode baris kelima sampai kode baris ketujuh. Hasil keluaran tersebut akan menunjukkan hasil dari nomor array data pada kolom bagian kiri dan nomor array dari sampel data pada kolom kanan.

3.11.10 Pemisahan data

Pengujian data mandiri ini dilakukan dengan pemisahan data menggunakan *One Hot Encoder* dan *Label Encoder*. Kode program yang akan digunakan dapat dilihat melalui Kode 3.22 dan Kode 3.23.

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
label_encoder = LabelEncoder()
integer_encoded = label_encoder.fit_transform(label)

onehot_encoder = OneHotEncoder(sparse=False)
integer_encoded = integer_encoded.reshape(-1, 1)
one_hot_label = onehot_encoder.fit_transform(integer_encoded)
```

Kode 3.22: Pengenalan data dengan menggunakan Label Encoder dan One Hot Encoder

Proses pada Kode 3.22 dilakukan pembuatan ukuran *encoder* menjadi (-1, 1) pada baris keenam agar semua data dapat dilakukan pengujian dengan baik. Setelah proses pada Kode 3.22 dilakukan, proses pemisahan data dilakukan menggunakan *train test split*. Pemisahan data tersebut akan membentuk data dengan ukuran 15.360. Penjelasan lebih rinci dapat dilihat pada Kode 3.23.

```
data = np.array(data).reshape(-1, 15360, 1)
label = np.array(label)

# Replace 'y' with 'label' for stratification
X_train, X_test, y_train, y_test = train_test_split(data,
    one_hot_label, test_size=0.25, stratify=label, shuffle=True,
    random_state=44)
print('X_train shape is ', X_train.shape)
print('X_test shape is ', X_test.shape)
print('y_train shape is ', y_train.shape)
print('y_test shape is ', y_test.shape)

X_train = tf.convert_to_tensor(X_train.astype(np.float32))
X_test = tf.convert_to_tensor(X_test.astype(np.float32))
```

```
13 y_train = tf.convert_to_tensor(y_train.astype(np.float32))
14 y_test = tf.convert_to_tensor(y_test.astype(np.float32))
```

Kode 3.23: Pemisahan data menggunakan train test split

Kode 3.23 baris kelima adalah kode yang menentukan seberapa banyak data yang digunakan untuk proses pelatihan dan seberapa banyak data yang digunakan untuk proses pengujian. Pada kode yang dibuat, terdapat ukuran data yang dilakukan pengujian adalah 0,25 atau setara dengan 25% dari jumlah sampel yang seluruhnya. Namun, ukuran pengujian tersebut bisa diubah-ubah sesuai kemauan karena secara umum jumlah data yang dilakukan pengujian biasanya 20% dan sisanya dilakukan untuk proses pelatihan. Pada penelitian kali ini, ada 75% data yang digunakan untuk pelatihan dan 25% data yang digunakan untuk pengujian. Semua data tersebut akan dilakukan konversi ke tensorflow berdasarkan Kode 3.23 baris ke-11 sampai baris ke-14.

3.12 Proses Pembuatan Model

Proses pembuatan model adalah proses yang dilakukan untuk pemeriksaan semua data yang telah dibuat. Proses tersebut berupa pengujian dan pelatihan data dengan menggunakan label sebenarnya, label prediksi, *confusion* matriks, dan laporan klasifikasi.

3.12.1 Pelatihan Data

Proses pelatihan data ada banyak tahapan. Tahapan tersebut diawali dengan pembentukan beberapa lapisan. Lapisan tersebut berupa lapisan konvolusi satu dimensi, parameter total, parameter untuk pelatihan, dan parameter yang tidak digunakan dalam pelatihan. Proses pembentukan lapisan tersebut dapat dilihat pada Kode 3.24 dan Kode 3.25.

```
1 num_class=len(pd.unique(label))
2 model=keras.Sequential()
3 model.add(keras.layers.Conv1D(filters=8, kernel_size=13,activation
    =tf.nn.relu,input_shape=(15360,1)))
4 model.add(keras.layers.MaxPooling1D(3))
5 model.add(keras.layers.Dropout(.3))
6 model.add(keras.layers.Conv1D(filters=16, kernel_size=11,
    activation=tf.nn.relu))
7 model.add(keras.layers.MaxPooling1D(3))
8 model.add(keras.layers.Dropout(.3))
```

```
model.add(keras.layers.Conv1D(filters=32, kernel_size=9,activation
    =tf.nn.relu))
model.add(keras.layers.MaxPooling1D(3))
model.add(keras.layers.Dropout(.3))
model.add(keras.layers.Conv1D(filters=64, kernel_size=7,activation
    =tf.nn.relu))
model.add(keras.layers.MaxPooling1D(3))
model.add(keras.layers.Dropout(.3))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(256,activation=tf.nn.relu))
model.add(keras.layers.Dropout(.3))
model.add(keras.layers.Dense(128,activation=tf.nn.relu))
model.add(keras.layers.Dropout(.3))
model.add(keras.layers.Dense(num_class,activation=tf.nn.softmax))
```

Kode 3.24: Proses pembentukan model

```
model.summary()
```

Kode 3.25: Menunjukkan model yang ada

Kode 3.24 baris ketiga akan membagi data menjadi berukuran 15.360. Kode 3.24 baris keempat sampai baris ke-20 akan membagi data berdasarkan matriks konvolusi satu dimensi berdasarkan penggunaan lapisan *pooling*. Hasil keluaran terdapat pada Kode 3.25. Hasil dari keluaran tersebut ditampilkan pada Gambar 4.16 pada hasil dan diskusi. Setelah proses tersebut dilakukan, proses penglihatan model akan dilakukan. Proses tersebut akan ditulis melalui Kode 3.26.

```
tf.keras.utils.plot_model(model, to_file='model.png', show_shapes=
    True, show_layer_names=True,show_dtype=True,dpi=120)
```

Kode 3.26: Menunjukkan grafik model

Hasil dari Kode 3.26 akan disimpan dengan format gambar. Format gambar tersebut adalah PNG seperti yang tertulis pada baris pertama Kode 3.26. Hasilnya akan menampilkan grafik model yang akan ditampilkan pada Gambar 5.8 pada halaman lampiran.

Setelah grafik tersebut tampil, proses pengujian model dilakukan. Proses tersebut dilakukan dengan iterasi sebanyak jumlah ragam kosakata yang digunakan dengan menggunakan himpunan data sendiri. Pengujian model dilakukan dengan melakukan percobaan melalui model pra-pelatihan melalui penggunaan model yang terdapat pada Kaggle. Namun, sebelum proses pengujian, nilai akurasi data akan diperiksa terlebih dahulu. Akurasi data akan diperiksa melalui Kode 3.27.

```
loss,acc=model.evaluate(X_test,y_test)
print('Loss is :',loss)
print('ACC is :',acc)
```

Kode 3.27: Menampilkan rata-rata akurasi data

Pada Kode 3.27 diperoleh hasil kerugian berdasarkan Kode 3.27 baris kedua dan akurasi data pada Kode 3.27 baris ketiga. Setelah itu, pengujian model dilakukan dengan menggunakan himpunan data sendiri pada aplikasi Visual Studio Code. Kode dari proses pengujian model dapat dilihat pada Kode 3.28.

```
model.compile(optimizer ='adam',loss='categorical_crossentropy',
    metrics=['accuracy'])
hist=model.fit(X_train,y_train,epochs=40)
```

Kode 3.28: Proses pelatihan data

Kode 3.28 akan menjalankan proses pelatihan data. Proses tersebut akan menggunakan optimalisasi Adam dan kategori kerugian *categorical crossentropy* pada baris pertama Kode 3.28. Kode tersebut akan dilakukan proses iterasi sebanyak 40 iterasi pada Kode 3.28 baris kedua. Proses setiap iterasi akan berlangsung selama 15-40 detik. Sehingga total waktu yang dibutuhkan untuk melakukan proses iterasi rata-rata 10-30 menit.

Setelah menunggu proses iterasi berakhir, akurasi data akan tampil. Akurasi data ditampilkan dengan menjalankan Kode 3.27 setelah proses pelatihan data selesai. Kode 3.27 baris kedua akan menampilkan kerugian. Kode 3.27 baris ketiga akan menampilkan hasil dari akurasi data. Kedua baris tersebut merupakan proses pengujian data yang dilakukan sebanyak jumlah iterasi yang dihitung secara rata-rata. Hasil akurasi data pada proses iterasi juga dapat ditampilkan secara satu persatu. Hasil akurasi tersebut akan ditulis melalui Kode 3.29.

```
hist_=hist.history
pd.DataFrame(hist_)
```

Kode 3.29: Menampilkan akurasi data satu-persatu

Kode 3.29 sepanjang dua baris akan menampilkan secara lengkap riwayat akurasi data mulai dari iterasi pertama sampai iterasi ke-40.

Setelah proses menampilkan akurasi data dilakukan secara rata-rata dan dilakukan secara satu-persatu, proses perhitungan akurasi data juga dilakukan dengan menjabarkan grafik. Proses tersebut dituliskan pada Kode 3.30.

```
plt.figure(figsize=(15,5))
plt.subplot(1,2,1)
plt.plot(hist_['loss'],c='r',marker='*',label='Loss')
plt.title('Overall Loss',fontsize=20)
plt.xlabel('Sample number', fontsize=12)
plt.ylabel('Loss history count', fontsize=12)
plt.legend()
plt.subplot(1,2,2)
plt.plot(hist_['accuracy'],label='Accuracy')
plt.title('Overall Accuracy',fontsize=20)
```

```
plt.xlabel('Sample number', fontsize=12)
plt.ylabel('Acccuracy history count', fontsize=12)
plt.legend()
```

Kode 3.30: Menampilkan akurasi data dalam grafik

Kode 3.30 merupakan kode untuk menampilkan keluaran grafik berdasarkan kerugian dan akurasi data. Keluaran grafik kerugian dapat dilihat pada Kode 3.30 baris ketiga dan keluaran grafik akurasi data dapat dilihat pada Kode 3.30 baris kesembilan. Kedua grafik tersebut digunakan untuk menampilkan nomor sampel dan riwayat kerugian & akurasi dari proses pelatihan data sebanyak 40 kali iterasi.

3.12.2 Pengujian Data

Proses pengujian data dilakukan dengan menggunakan X_*test* dan y_*test*. Namun, pada penelitian ini, pengujian data berpedoman pada X_*test*. Pengujian X_*test* dapat dilihat pada Kode 3.31.

```
predict=model.predict(X_test)
predict[0]
```

Kode 3.31: Pengujian data dengan menggunakan X test

Kode 3.31 akan menampilkan keluaran berupa array yang memiliki tipe tensorflow berdasarkan Kode 3.31 baris pertama. Array data dari X_*test* dilakukan pada array nol sesuai dengan Kode 3.31 baris kedua.

Setelah proses X_*test* dilakukan, proses pengujian data dilakukan dengan menggunakan model prediksi. Kode untuk pelatihan dengan model prediksi dapat dilihat pada Kode 3.32.

```
preN=[]
prename=[]
for row in predict:
    N=np.argmax(row)
    preN.append(N)
    prename.append(get_Name(N))
pd.DataFrame(prename,columns=['Predictions'])
```

Kode 3.32: Proses pelatihan menggunakan model prediksi

Kode 3.32 baris keempat digunakan untuk menampilkan model kosakata yang digunakan untuk prediksi yang dimulai dari N terkecil sampai N paling maksimal. Model prediksi ini akan menggunakan fungsi get_Name pada Kode 3.32 baris keenam. Hasil dari pengujian data akan tampil pada Kode 3.32 baris ketujuh. Hasil dari pengujian data akan menampilkan sebanyak jumlah data yang dilakukan proses pengujian.

Setelah pengujian data melalui model prediksi dilakukan, proses pengujian data dilakukan berdasarkan nilai y sebenarnya dan model prediksi. Kode pada proses pengujian tersebut dijelaskan pada Kode 3.33.

```
def get_Name(N):
    # Assuming code.values() and N are both TensorFlow tensors
    N = np.argmax(N)
    matches = tf.equal(list(code.values()), N)
    indices = tf.where(matches)
    if tf.size(indices) > 0:
        return list(code.keys())[indices[0, 0].numpy()]
```

Kode 3.33: Inisiasi fungsi getName

Pada Kode 3.33 baris pertama, terdapat fungsi get_Name. Fungsi get_Name digunakan untuk mengubah nilai N dari *One Hot Encoder* menjadi integer. Pada baris ketiga Kode 3.33 juga terdapat fungsi yang bertujuan agar nilai N yang dilakukan proses pengujian data dilakukan mulai dari nilai N0 sampai nilai N maksimal. Setelah itu, kode untuk proses pengujian selanjutnya akan ditulis pada Kode 3.34.

```
predict=[]
y_act=[]
for p in range(1280):
    y_act.append(get_Name(y_test[p]))
    predict.append(prename[p])
pd_p=pd.DataFrame(y_act,columns=['y_act'])
pd_p['predict']=predict
pd_p
```

Kode 3.34: Pengujian data berdasarkan nilai y sebenarnya dan model prediksi

Kode 3.34 akan menampilkan nilai dari y sebenarnya pada baris pertama, dan nilai dari model prediksi pada baris kedua. Pada baris kedelapan Kode 3.34, hasil keluaran dari y sebenarnya dan model prediksi akan tampil sebanyak 1.280 baris berdasarkan Kode 3.34 baris ketiga. Nilai y sebenarnya akan memiliki N ragam nama sebenarnya dan model prediksi akan memiliki N ragam nama prediksi yang sebenarnya.

3.12.3 Pemeriksaan Model

Pada penelitian kali ini, pemeriksaan model dilakukan sebanyak dua tahap. Tahapan tersebut dimulai dari pemeriksaan berdasarkan *confusion* matriks. Setelah itu dilanjutkan dengan proses pemeriksaan berdasarkan laporan klasifikasi.

Proses pemeriksaan berdasarkan *confusion* matriks dapat dilihat pada Kode 3.35.

```
# Convert one-hot encoded true labels to integer labels
y_test_int = np.argmax(y_test, axis=1)

plt.figure(figsize=(15, 15))
ax = plt.subplot()
CM = confusion_matrix(y_test_int, preN)
sns.heatmap(CM, annot=True, fmt='g', ax=ax, cbar=False, cmap='RdBu')
ax.set_xlabel('Predicted labels')
ax.set_ylabel('True labels')
ax.set_title('Confusion Matrix')
ax.xaxis.set_ticklabels(code.keys())
ax.yaxis.set_ticklabels(code.keys())
print('Confusion matrix is: ')
plt.show()
CM
```

Kode 3.35: Proses pemeriksaan menggunakan matriks confusion

Kode 3.35, hasil dari *confusion* matriks akan ditampilkan berdasarkan label prediksi di sumbu x berdasarkan Kode 3.35 baris kedelapan dan label sebenarnya di sumbu y berdasarkan Kode 3.35 baris kesembilan. Hasil keseluruhan dari *confusion* matriks, ditunjukkan pada baris ke-14 Kode 3.35. Hasil tersebut ditampilkan dalam kotak kecil berukuran persegi yang berukuran jumlah iterasi kali jumlah iterasi.

Selain berdasarkan *confusion* matriks, proses pemeriksaan model dilakukan berdasarkan laporan klasifikasi. Proses tersebut dilakukan untuk memastikan hasil dari proses pengujian data. Proses tersebut dicantumkan pada Kode 3.36.

```
# Classification report testing

# Print the classification report
print('Classification report is: ')
print(classification_report(y_act, predict))
```

Kode 3.36: Proses pemeriksaan berdasarkan laporan klasifikasi

Kode 3.36 menampilkan hasil dari laporan klasifikasi pada kode baris keempat dan kode baris kelima. Hasil tersebut akan memperoleh nilai presisi, nilai akurasi, nilai *recall*, dan nilai F1. Selain itu, rata-rata makro dan rata-rata berbobot juga terdapat pada laporan klasifikasi.

3.13 Menyimpan Model

Proses menyimpan model merupakan hal yang penting untuk dilakukan. Proses tersebut merupakan bagian terpenting agar model tersebut tidak hilang dan

tidak rusak. Kode untuk penyimpanan model dapat dilihat pada Kode 3.37 dan Kode 3.38.

```
model.save('test1.h5')
```

Kode 3.37: Menyimpan hasil pengujian model

```
savedmodel=keras.models.load_model('test1.h5')
```

Kode 3.38: Memuat hasil pengujian model

Model yang telah dibuat akan disimpan pada Kode 3.37. Setelah model tersebut disimpan, model tersebut akan terdapat pada berkas yang ada pada komputer. Kemudian model tersebut akan dimuat dengan menggunakan Kode 3.38.

3.14 Metode Melihat Model

Model pelatihan dan pengujian yang telah dilakukan sangat penting. Pada konteks kali ini, model yang dilihat adalah model X_*test*, y_*test*, X_*train*, dan y_*train*. Model yang dilihat telah disimpan pada berkas dengan format h5. Dengan menyimpan berkas dengan format h5 tersebut, proses menjalankan model secara berulang kali tidak perlu dilakukan lagi. Proses melihat model pada penelitian ini dilakukan dengan proses melihat model melalui banyak himpunan data yang tersedia, jenis himpunan data yang tersedia, ukuran himpunan data, isi dari himpunan data, kunci yang ada pada himpunan data, dan perbandingan akurasi antara pengujian pra-pelatihan dan pengujian menggunakan data sendiri.

3.14.1 Himpunan data

Proses melihat model dilakukan dengan membuat halaman kode yang baru. Halaman kode tersebut memiliki format Python. Melihat model ini merupakan tahapan pertama yang dilakukan setelah proses pengujian model selesai dilakukan. Melihat model pada penelitian kali ini dilakukan dengan melihat model berdasarkan himpunan data yang tersedia. Proses penglihatan model tersebut dapat dilihat pada Kode 3.39.

```
# Open the H5 file
with h5py.File('test1.h5', 'r') as file:
    # List all dataset names within the file
    dataset_names = list(file.keys())
    print("Available dataset names:", dataset_names)
```

Kode 3.39: Melihat himpunan data yang ada pada model pelatihan

Pada Kode 3.39 dikatakan bahwa himpunan data yang tersedia pada model pelatihan dan pengujian dapat dilampirkan dengan cepat. Himpunan data tersebut akan dibuka melalui penulisan Kode 3.39 baris kedua. Lalu, himpunan data tersebut akan dilampirkan berdasarkan Kode 3.39 baris kelima.

3.14.2 Jenis himpunan data

Setelah proses melihat model berdasarkan himpunan data yang tersedia, proses kedua adalah proses melihat model berdasarkan jenis himpunan data yang tersedia. Untuk melakukan proses penglihatan tersebut, diperlukan kode program dengan menuliskan Kode 3.40.

```
#Open the HDF5 file
with h5py.File('test1.h5', 'r') as hdf5_file:
    # Access a group named 'model_weights'
    group = hdf5_file['model_weights']

    # Now you can work with the group's contents as needed
    # For example, you can list the datasets within the group
    for dataset_name in group:
        print(dataset_name)
```

Kode 3.40: Melihat jenis himpunan data pada model pelatihan

Pada Kode 3.40, jenis himpunan data yang tersedia dapat dilampirkan dengan membuka berkas h5 pada kode baris kedua. Setelah itu, kelas dari berkas tersebut dilakukan pengaturan berdasarkan Kode 3.40 baris keempat. Hasilnya dari himpunan data tersebut terdapat pada baris kesembilan Kode 3.40.

3.14.3 Ukuran dari himpunan data

Proses ketiga dalam melihat model adalah proses melihat model berdasarkan ukuran dari himpunan data. Pada penelitian kali ini, ada dua jenis himpunan data yang digunakan. Proses melihat model berdasarkan ukuran dari himpunan data dapat dilihat pada Kode 3.41 dan Kode 3.42.

```
with h5py.File('test1.h5', 'r') as hdf:
    data = hdf.get('model_weights')
    dataset1 = np.array(data)
    print('Shape of dataset1: \n', dataset1.shape)
```

Kode 3.41: Menampilkan ukuran dari dataset1

Kode 3.41 digunakan untuk menampilkan ukuran dari himpunan data pertama. Himpunan data pertama berupa *model_weights* yang terdapat pada

baris kedua Kode 3.41. Ukuran dari himpunan data tersebut akan ditampilkan berdasarkan baris keempat Kode 3.41.

```
with h5py.File('test1.h5', 'r') as hdf:
    data = hdf.get('optimizer_weights')
    dataset2 = np.array(data)
    print('Shape of dataset2: \n', dataset2.shape)
```

Kode 3.42: Menampilkan ukuran dari dataset2

Kode 3.42 digunakan untuk menampilkan ukuran dari himpunan data kedua. Himpunan data pertama berupa *optimizer_weights* yang terdapat pada baris kedua Kode 3.42. Ukuran dari himpunan data tersebut akan ditampilkan berdasarkan baris keempat Kode 3.42.

3.14.4 Isi dari himpunan data

Isi dari himpunan data dapat dilihat. Isi dari himpunan data dilakukan setelah ukuran dari himpunan data diketahui. Setelah ukuran dari himpunan data diketahui, proses melihat isi dari banyaknya himpunan data tersebut dilakukan. Hasilnya akan menampilkan keluaran yang berupa array.

3.14.5 Kunci pada himpunan data

Proses keempat dalam melakukan penglihatan model adalah proses melihat model berdasarkan kunci pada himpunan data. Proses ini merupakan proses melihat kunci yang ada berdasarkan himpunan data yang tersedia dan melihat kunci yang ada berdasarkan isi dari himpunan data yang tersedia. Kode program pada metode penglihatan ini dapat dilihat pada Kode 3.43.

```
# Open the H5 file for reading
h5_file = h5py.File("test1.h5", "r")

# Access and inspect "optimizer_weights" group
optimizer_weights_group = h5_file["optimizer_weights"]
print("Keys in 'optimizer_weights' group:", list(
    optimizer_weights_group.keys()))

# Access and inspect the "Adam" group
adam_group = optimizer_weights_group["Adam"]
print("Keys in 'Adam' group:", list(adam_group.keys()))
```

Kode 3.43: Menampilkan kunci utama pada optimizer weights

Pada Kode 3.43, berkas h5 akan dibuka berdasarkan kode baris kedua. Kode 3.43 akan menampilkan kunci utama yang ada pada *optimizer_weights* berdasarkan kode baris keenam dan kunci utama yang ada pada kelompok Adam berdasarkan kode baris ke-10.

Selain *optimizer_weights*, metode penglihatan kunci utama pada himpunan data dapat berdasarkan *model_weights*. Kode untuk penglihatan tersebut dapat dilihat pada Kode 3.44.

```
# Open the H5 file for reading
h5_file = h5py.File("test1.h5", "r")

# Access and inspect "optimizer_weights" group
model_weights_group = h5_file["model_weights"]
print("Keys in 'model_weights' group:", list(model_weights_group.
    keys()))

# Access and inspect the "conv1d_3" group within
    model_weights_group
conv1d_3_group = model_weights_group["conv1d_3"]
print("Keys in 'conv1d_3' group:", list(conv1d_3_group.keys()))
```

Kode 3.44: Menampilkan kunci utama pada model weights

Pada Kode 3.44, berkas h5 akan dibuka berdasarkan kode baris kedua. Kode 3.44 akan menampilkan kunci utama yang ada pada *model_weights* berdasarkan kode baris keenam dengan kunci utama yang ada pada kelompok conv1d_3_group berdasarkan kode baris ke-10.

3.14.6 Membandingkan Akurasi Data

Proses kelima dalam melakukan penglihatan model adalah membandingkan akurasi data. Proses ini membandingkan akurasi data melalui model pelatihan berdasarkan pengujian data menggunakan metode pra-pelatihan melalui Kaggle dan pengujian data menggunakan metode pengujian melalui Visual Studio Code. Kedua proses ini akan menggunakan himpunan data sendiri.

Penelitian ini akan membandingkan hasil akurasi data dengan menggunakan *optimizer* Adam. Jumlah data yang dilakukan pelatihan adalah 75% dan sisanya digunakan untuk pengujian. Hasil dari akurasi data pada ketiga kode tersebut berbeda-beda. Kondisi ini tergantung dengan banyak data yang diuji, dan jumlah sampel setiap data yang digunakan. Pada konteks kali ini, perbandingan akurasi data akan dilakukan. Kode untuk perbandingan akurasi data akan ditulis pada Kode 3.45.

```
# Simpan akurasi ke dalam sebuah file teks
with open('akurasi1.txt', 'w') as file:
    file.write(str(acc))
```

Kode 3.45: Menulis dan menyimpan akurasi data

Pada Kode 3.45, akurasi data akan ditulis dan disimpan berdasarkan huruf w yang terdapat pada baris kedua. Akurasi data tersebut akan disimpan dengan format txt. Setelah akurasi data disimpan, akurasi akan tampil berdasarkan tulisan yang ada pada Kode 3.45 baris ketiga. Setelah itu, akurasi data akan dibaca pada model h5. Kode untuk membaca akurasi data ditulis pada Kode 3.46.

```
# Membaca nilai akurasi data
with open('akurasi1.txt', 'r') as file:
    saved_acc = float(file.read())
    print("Akurasi yang disimpan: {:.2f}%".format(saved_acc * 100)
    )
```

Kode 3.46: Membaca akurasi data

Pada Kode 3.46, akurasi data yang telah simpan, akan ditunjukkan pada model. Akurasi data tersebut akan dibaca oleh model berdasarkan huruf r yang terdapat pada baris kedua. Hasil dari akurasi data akan ditampilkan pada model berdasarkan kode baris keempat. Akurasi tersebut akan dikalikan 100 dan ditampilkan dalam satuan %.