

BAB I

PENDAHULUAN

A. Latar Belakang

Saat ini, teknologi memiliki peran yang sangat penting bagi kehidupan manusia. Dengan teknologi, sebuah informasi sangat mudah untuk diperoleh. Akan tetapi, informasi maupun pengetahuan ini sangat sulit diperoleh dengan cepat dan akurat bagi kalangan tertentu, khususnya para penyandang tuna netra. Ini disebabkan mereka memerlukan alat bantu untuk membaca maupun menulis, yaitu huruf *braille*. Pola huruf *braille* dengan enam titik yang terbagi dalam tiga baris dan dua kolom, memiliki tingkat kompleksitas yang tinggi untuk dipahami secara cepat. Akibatnya, para penyandang tuna netra lebih lambat dalam memperoleh informasi maupun pengetahuan daripada masyarakat pada umumnya. (Romando, E., Sudaryanto, A. 2018).

Huruf *braille* adalah suatu sistem penulisan yang menggunakan titik-titik yang timbul yang mewakili karakter tertentu. Huruf yang diciptakan oleh Louis *braille* ini diperuntukkan untuk orang tunanetra. Satuan dasar dari sistem tulisan ini disebut sel *braille*, dimana tiap sel terdiri dari enam titik timbul, tiga baris dengan dua titik. Keenam titik tersebut dapat disusun sedemikian rupa hingga menciptakan bermacam kombinasi (Subur, J., 2015).

Cara membaca huruf *braille*, yaitu dengan menempelkan ujung jari-jari tangan pada lembar huruf *braille* dan meraba titik-titik timbul pada huruf *braille*. Perlu dilatih kepekaan tangan dalam membaca huruf *braille* tersebut dan harus

hafal setiap kombinasi titik-titik yang terbentuk. Sehingga proses untuk dapat mengetahui huruf *braille* dibutuhkan waktu yang relatif lama (Subur, J., 2015).

Masalah tersebut dapat diatasi melalui perkembangan teknologi, seperti maraknya penggunaan *smartphone* berbasis android yang saat ini menjadi salah satu *platform* paling populer di kalangan masyarakat umum.

Android saat ini merupakan salah satu perangkat lunak perangkat seluler yang paling populer. Android dikembangkan berdasarkan *linux*, termasuk sistem operasi, *middleware*, dan aplikasi android. Hingga saat ini, ada sembilan versi android 1.0, android 1.1, android 1.5 (*cupcake*), android 1.6 (*donut*), 2.0-2.1 (*eclair*), 2.2-2.3 (*froyo*), 2.3-versi 2.3.7 (*gingerbread*) hingga versi android terbaru saat ini adalah android 11 (Anwar, Y., 2021).

Berdasarkan latar belakang yang telah penulis uraikan maka penulis melakukan penelitian yang berjudul “Aplikasi Penerjemah Huruf *Braille* Berbasis Android”.

B. Rumusan Masalah

Berdasarkan dari uraian latar belakang diatas, maka rumusan masalah yang akan dibahas adalah bagaimana menerjemahkan huruf *braille* berbasis android?

C. Tujuan Penelitian

Sesuai dengan permasalahan yang telah diuraikan diatas, maka tujuan dari penelitian ini adalah untuk memudahkan orang yang ingin mengetahui huruf *braille* dalam bahasa Indonesia.

D. Batasan Masalah

Penelitian ini memiliki beberapa batasan masalah dengan harapan penelitian terfokus dengan batasan-batasan yang dibuat. Adapun batasan dalam penelitian sebagai berikut :

1. Sistem yang dirancang hanya untuk menerjemahkan huruf *braille* ke dalam huruf abjad.
2. Sistem yang dirancang menggunakan *software Android Studio* dan menggunakan bahasa pemrograman java dan kotlin.
3. Karakter yang diolah adalah A sampai Z.

E. Manfaat Penelitian

Manfaat dari penelitian ini adalah untuk membantu masyarakat awam yang ingin mengetahui dan mempelajari karakter huruf *braille*.

F. Sistematika Penulisan

Secara garis besar sistematika penulisan yang dapat disajikan sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menguraikan tentang teori-teori yang menyangkut tentang penelitian ini.

BAB III METODE PENELITIAN

Bab ini menguraikan mengenai metode metode yang akan digunakan dalam penelitian baik dari jenis penelitiannya, tahapan, bagan alir serta lain sebagainya.

BAB IV HASIL PENELITIAN DAN PEMBAHASAN

Dalam bab ini menjelaskan mengenai rancangan sistem yang telah dibuat, metode dan teknik pengujian.

BAB V KESIMPULAN DAN SARAN

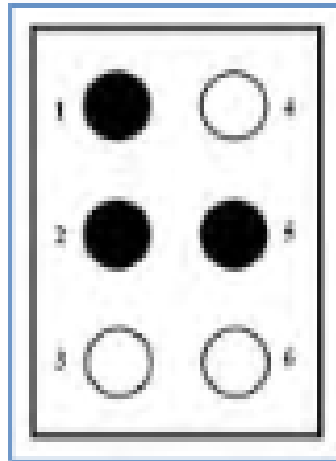
Dalam bab ini menjelaskan mengenai kesimpulan dan saran.

BAB II

TINJAUAN PUSTAKA

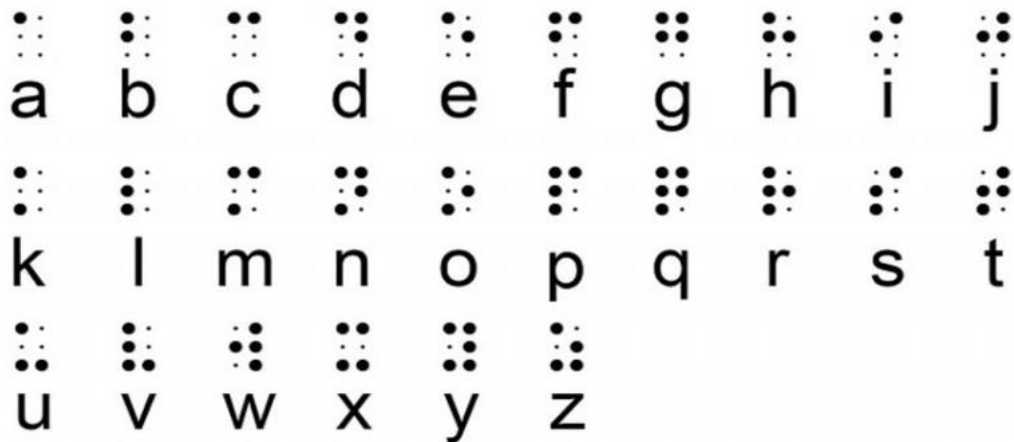
A. Huruf *braille*

Huruf *braille* adalah suatu sistem penulisan yang menggunakan titik-titik yang timbul yang mewakili karakter tertentu. Huruf yang diciptakan oleh Louis Braille ini diperuntukkan untuk orang tunanetra. Satuan dasar dari sistem tulisan ini disebut sel *braille*, dimana tiap sel terdiri dari enam titik timbul, tiga baris dengan dua titik. Keenam titik tersebut dapat disusun sedemikian rupa hingga menciptakan bermacam kombinasi (Subur, J., 2015).



Gambar 2. 1 Susunan titik-titik dot pada huruf *braille*
(sumber: Subur, J, 2015)

Pada gambar 2.1 ditunjukkan adanya warna hitam dan putih, pada gambar tersebut dapat diasumsikan bahwa lingkaran yang berwarna hitam menunjukkan dot yang timbul sedangkan yang putih merupakan dot yang tidak timbul.



Gambar 2. 2 Susunan kombinasi titik-titik timbul pada huruf *braille*
(Sumber: Subur, J, 2015)

Pada Gambar 2.2 menunjukkan susunan kombinasi titik-titik pada huruf *braille* dalam mewakili huruf abjad. Setiap karakter memiliki kombinasi yang berbeda, sehingga dengan 6 titik tersebut maka bisa dibuat sebanyak 64 kombinasi.



Gambar 2. 3 Membaca huruf *braille* dengan diraba
(sumber: Subur, J, 2015)

Sedangkan pada gambar 2.3 ditunjukkan bagaimana cara membaca huruf *braille* pada umumnya yang dilakukan oleh tunanetra, yaitu dengan menempelkan ujung jari-jari tangan pada lembar huruf *braille* dan meraba titik-titik timbul pada huruf *braille*. Perlu dilatih kepekaan tangan dalam membaca huruf *braille*

tersebut dan harus hafal setiap kombinasi titik-titik yang terbentuk. Sehingga proses untuk dapat membaca huruf *braille* dibutuhkan waktu yang relatif lama (Subur, J, 2015).

B. Android

Android adalah sistem operasi seluler yang telah ada selama hampir 15 tahun yang telah dikeluarkan oleh *google*. Anda terutama akan menemukannya sebagai sistem operasi dasar ponsel dan tablet di seluruh dunia. Selain itu, ada sistem operasi lain yang secara *native* mendukung aplikasi android, termasuk *Chrome OS* dan *Windows 11* (Anwar, Y., 2021).

Android dimulai sebagai perangkat lunak untuk kamera. Andy Rubin dan timnya mengembangkan android mulai tahun 2003 dengan menggunakan kode inti dari *linux*, sistem operasi *open-source* lainnya. Idennya adalah untuk membuat sistem operasi universal yang dapat digunakan oleh semua perusahaan kamera.

Android adalah sistem operasi *open source*, dan *google* merilis kode sumber di bawah lisensi *apache*. Kode dan lisensi *open source* android memberi produsen perangkat kebebasan untuk memodifikasi dan mendistribusikan perangkat lunak. Sampai hari ini, ada 18 versi android, dengan 11 rilis utama. Versi stabil terbaru dari sistem operasi adalah android 11. Kami berharap android 12 akan mendarat nanti pada tahun 2021, pada akhir Agustus atau awal September, yang akan menjadikannya versi ke-19 dan rilis utama ke-12,). Berikut daftar versi OS hingga sekarang oleh Gambar 2.2



Gambar 2.2 Versi Android
(Sumber: Anwar, Y., 2021)

C. Pengolahan Citra

Citra (*image*) istilah lain untuk gambar sebagai salah satu komponen multimedia memegang peranan sangat penting sebagai bentuk informasi visual. Citra mempunyai karakteristik yang tidak dimiliki oleh data teks, yaitu citra kaya dengan informasi. Ada sebuah peribahasa yang berbunyi “sebuah gambar bermakna lebih dari seribu kata” (*a picture is more than a thousand words*). Maksudnya tentu sebuah gambar dapat memberikan informasi yang lebih banyak daripada informasi tersebut disajikan dalam bentuk kata-kata (tekstual) (Nurwahida, 2016).

Citra bergerak (*moving images*) adalah rangkaian citra diam yang ditampilkan secara beruntun (sekuensial) sehingga memberi kesan pada mata kita sebagai gambar yang bergerak. Setiap citra di dalam rangkaian itu disebut *frame*.

Gambar-gambar yang tampak pada film layar lebar atau televisi pada hakikatnya terdiri atas ratusan sampai ribuan *frame*. Agar citra yang mengalami gangguan mudah diinterpretasi (baik oleh manusia maupun mesin), maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik. Bidang studi yang menyangkut hal ini adalah pengolahan citra (Munir, R. 2004).

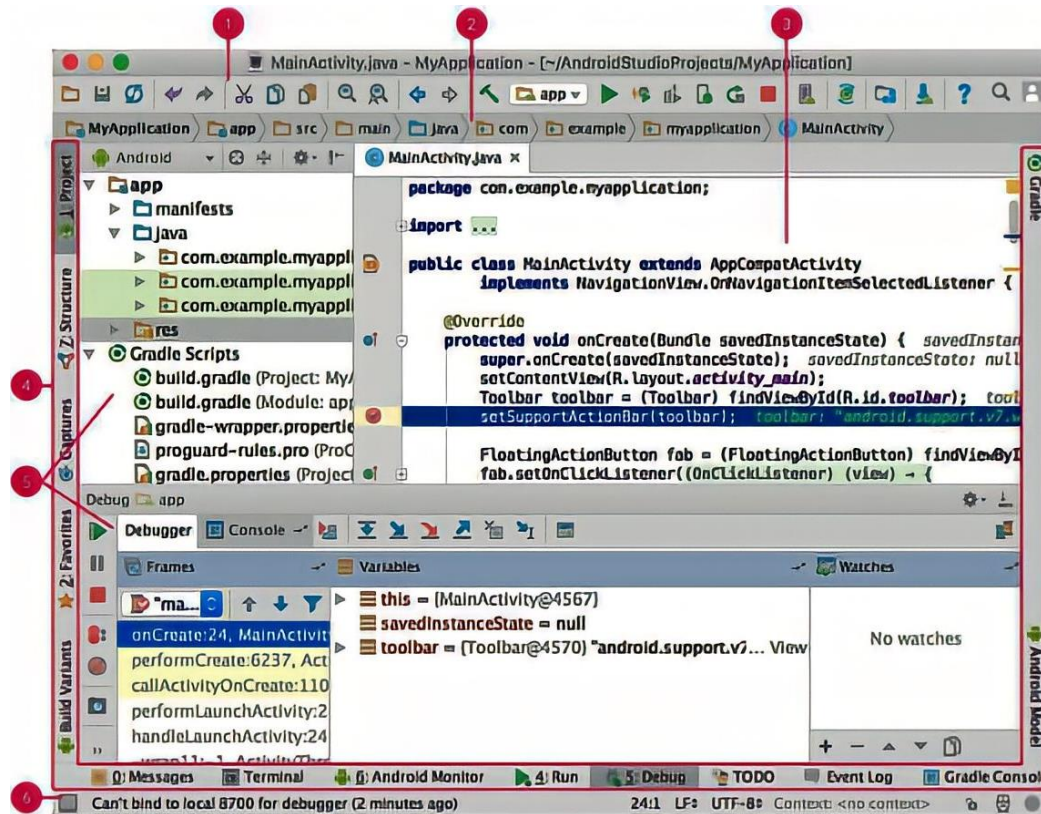
Beberapa operasi pemrosesan gambar antara lain, *image enhancement* untuk meningkatkan kualitas gambar dengan memanipulasi parameter gambar, *image restoration* untuk menghilangkan/ meminimalkan kesalahan gambar, *image compression* untuk mendapatkan representasi dalam bentuk yang lebih ringkas, lebih sedikit memori yang dibutuhkan. Segmentasi citra adalah penguraian citra menjadi segmen-segmen, Analisis citra adalah penghitungan kuantitas (Putra, D. 2010).

D. Android studio

Android Software Development Kit (SDK) merupakan alat yang bisa digunakan oleh para pengembang *mobile* untuk mengembangkan aplikasi berbasis *mobile Android*. Di dalamnya, terdapat beberapa *tools* seperti *emulator*, *software documentation*, *debugger*, *libraries*, *sample code* dan tutorial (Anwar, Y., 2021).

Dengan menggunakan *android studio*, pengembang dapat membuat aplikasi dari awal dan kemudian mempublikasikannya di toko. *android studio* juga memiliki beberapa fitur bawaan yang benar-benar dapat membantu pengembang mendapatkan hasil maksimal dari proses pengembangan aplikasi. Fitur-fitur ini termasuk *gradle*, penyelesaian kode, dan berbagai integrasi dengan

layanan *google* seperti *firebase* (Hansun, S., Kristanda, M. B. & Saputra, M. W., 2018).



Gambar 2.4 Android Studio Workspace

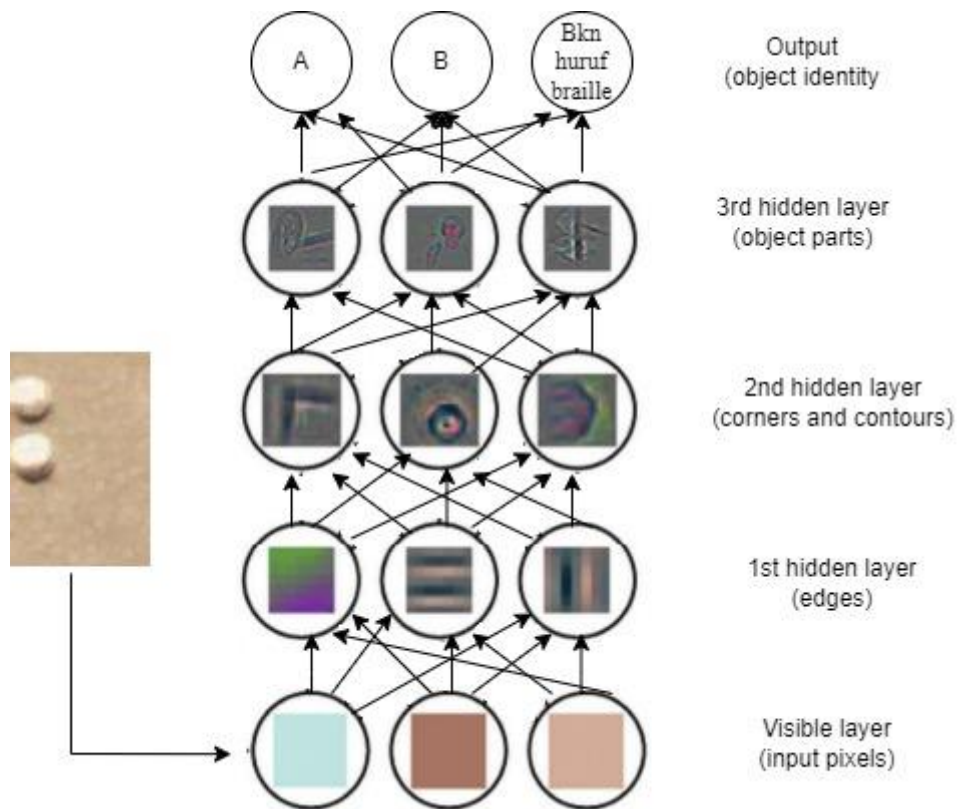
Keterangan :

1. Sebuah *tools* yang memungkinkan anda untuk jalankan aplikasi dan mulai alat android.
2. Sebuah navigasi untuk membantu anda dalam *explore* dalam membuka *file* untuk diedit antar *proyek*. Alat ini memberikan tampilan struktur yang lebih ringkas di jendela *proyek*.
3. Editor atau *workspace* adalah wadah untuk menulis sebuah kode untuk membuat suatu program.

4. *Tool* yang nampak di luar *IDE* dan berisi tombol-tombol yang memungkinkan anda untuk meluaskan atau mengecilkan jendela.
5. Sebuah alat untuk memberikan anda akses ke tugas tertentu berupa pengelolaan proyek, penelusuran *file*, *version control*, dan masih banyak lagi. anda bisa meluaskan dan juga menciutkan jendelanya.
6. *Window* status untuk menampilkan sebuah status pada *proyek* yang sedang anda kerjakan dan *IDE* itu sendiri, serta setiap peringatan atau *message* pada *proyek*.

E. Deep Learning

Deep Learning adalah bagian dari kecerdasan buatan dan pembelajaran mesin, yang merupakan pengembangan jaringan saraf *multi-layer* untuk menyediakan tugas-tugas presisi tinggi seperti pengenalan objek, pengenalan ucapan, dan terjemahan bahasa. Pembelajaran mendalam berbeda dari teknik pembelajaran mesin tradisional karena secara otomatis mewakili data seperti 16 gambar, video, dan teks tanpa memperkenalkan aturan kode atau pengetahuan tentang domain manusia. Masalah yang dulunya sulit diselesaikan oleh manusia kini menjadi masalah bagi kecerdasan buatan. dapat dengan mudah dipecahkan. AI mengalami kesulitan menerapkan intuisi tertentu pada pengetahuannya. Untuk mengatasi masalah ini, konsep deep learning digunakan. Pembelajaran mendalam menggunakan representasi sederhana, tetapi konsep ini memungkinkan komputer untuk membuat konsep kompleks seperti gambar. Model pembelajaran mendalam tumbuh seiring dengan perkembangan komputer, perangkat keras, dan perangkat lunak.



Gambar 2.5 *Ilustrasi deep learning*

Deep Learning dibangun pada tahun 1950 namun baru pada tahun 1990 dapat diterapkan dengan sukses. Hal terpenting tentang mengembangkan model pembelajaran mendalam saat ini adalah kami dapat memberi mereka sumber daya yang mereka butuhkan untuk berhasil. Evolusi terus menerus dari kumpulan data telah menghasilkan kumpulan data terpusat yang membuatnya lebih mudah untuk dikelola.

F. *Convolutional Neural Network (CNN)*

Convolutional neural network (CNN) adalah jenis jaringan saraf tiruan yang dirancang untuk tugas-tugas yang melibatkan data visual, seperti pengenalan gambar dan video, deteksi objek, dan klasifikasi gambar. CNN sangat efektif dalam menangkap pola hierarki dan ketergantungan spasial dalam data. Secara sederhana ada dua bagian dalam operasi CNN yaitu pembelajaran fitur dan klasifikasi. Dalam bagian pembelajaran fitur terdapat tiga proses yang dijelaskan, yaitu:

1. Operasi konvolusi, yang merupakan proses utama dalam CNN. Tujuannya adalah untuk mengestrak fitur dari citra masukan.
2. ReLu (*Rectified Linear Unit*), yaitu fungsi aktivasi yang memiliki output 0 jika input kurang dari 0. Artinya, jika $x \leq 0$ maka $x = 0$ dan jika $x > 0$ maka $x = x$.
3. *Pooling*, yang digunakan untuk mengurangi jumlah parameter pada peta fitur dan mengambil informasi yang paling penting dengan operasi *down-sampling*. Metode *pooling* yang paling umum digunakan adalah *max pooling* dan *average pooling*.

Pada bagian klasifikasi, terdapat tiga proses yaitu:

1. *Flatten*, *layer* ini mengubah output dari hasil *layer* konvolusi menjadi satu kolom *feature vector* untuk digunakan pada *layer* terhubung secara penuh.
2. *Fully connected layer*, *layer* ini mirip dengan *Artificial Neural Network (ANN)* untuk melakukan klasifikasi.

3. *Softmax*, untuk mendapatkan nilai probabilitas dari setiap kelas, nilai probabilitas kelas tertinggi merupakan output prediksi kelas yang diperoleh. Proses-proses tersebut merupakan operasi yang umum digunakan dalam berbagai model CNN.

Metode dengan menggunakan *convolutional neural network* (CNN) merupakan salah satu metode dari *machine learning* yang merupakan pengembangan dari *multi layer perceptron* yang mana dirancang untuk mengolah atau membuat data dari dua dimensi. CNN juga merupakan salah satu jenis metode dari *Deep Neural Network* yang dikarenakan didalamnya memiliki tingkat jaringan dan mempunyai banyak penerapan yang dilakukan di dalam citra (Azmi, K., dkk 2019)

Convolutional neural network telah sangat berhasil dalam berbagai tugas visi komputer, mencapai hasil canggih dalam tantangan klasifikasi gambar seperti *ImageNet*. Mereka juga banyak digunakan dalam pembelajaran transfer, di mana CNN yang telah dilatih sebelumnya disesuaikan untuk tugas-tugas tertentu dengan data terbatas. Adapun algoritma CNN dapat diuraikan sebagai berikut :

1. Membaca dari datasets berupa sebuah citra gambar.
2. Memecah citra gambar menjadi beberapa citra gambar yang berukuran kecil.
3. Melakukan perhitungan dengan algoritma CNN.
4. Mengelompokkan klasifikasi sesuai class.

Cara kerja sebuah algoritma CNN dapat dijelaskan berdasarkan dari 4 langkah, yaitu: *convolution*, *subsampling*, *activation*, dan *fully connected*. Pada langkah pertama, konvolusi, fungsi menerima input (input gambar). Konvolusi

biasanya merupakan operasi antara dua fungsi untuk menghasilkan fungsi ketiga, yang merupakan modifikasi dari dua fungsi aslinya. (Zufar, M., & Setiyono, B. 2016).

Tabel 2.1 Operasi Convolutional

Fungsi	Persamaan
Umum	$f(x, y) * g(x, y) = f(x, y) g(x, y) (1)$
Kontinu	$h(x) = f(x) * g(x)$ $= \int_{-\infty}^{\infty} f(a) g(x - a) da (2)$
Diskret	$h(x) = f(x) * g(x)$ $= \sum_{a=-\infty}^{\infty} f(a) g(x - a) (3)$

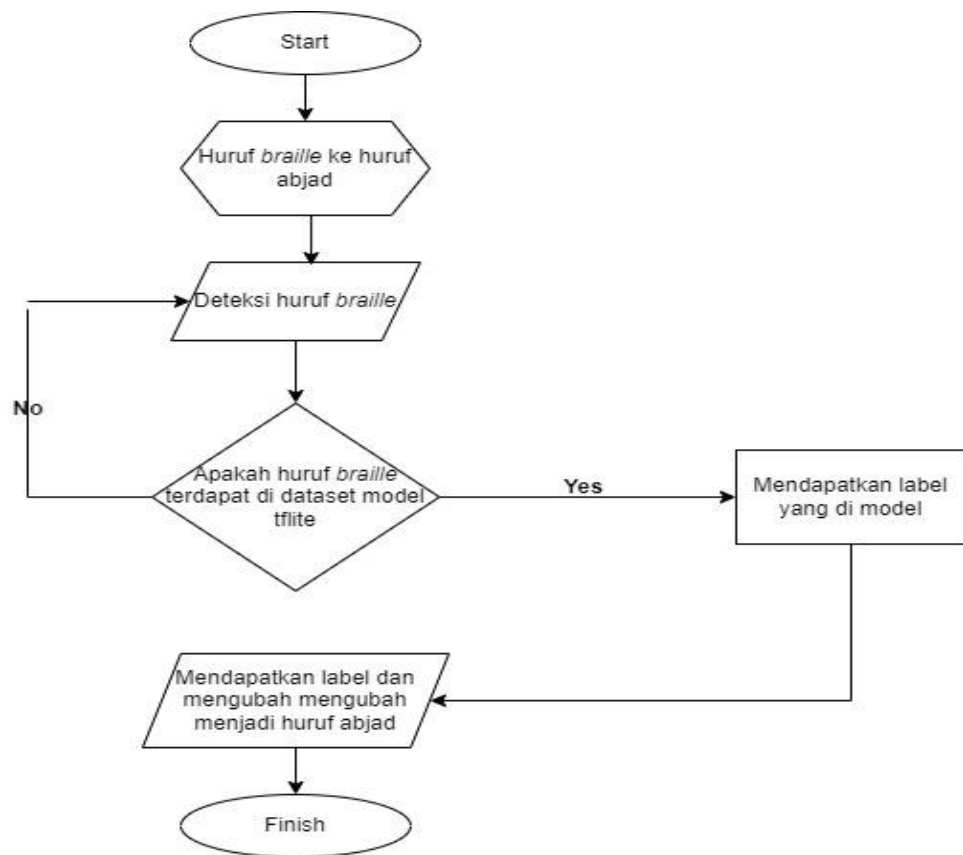
Tahap selanjutnya adalah subsampling, input *convolutional* dapat ditingkatkan untuk tidak berlebihan *sensitivity noise*. Proses ini disebut subsampling, yang dapat diraih dengan mengambil nilai rata-rata atau maksimum dari beberapa signal. Contoh salah satu subsampling adalah mengurangi ukuran gambar atau mengurangi kontras warna (RGB) pada gambar. Kemudian langkah ketiga adalah aktivasi, yang mengarahkan sinyal dari satu lapisan ke lapisan lain tergantung pada bagaimana neuron berfungsi. Sinyal keluaran yang digabungkan

dengan referensi sebelumnya mengaktifkan banyak neuron lain sehingga sinyal dapat ditransmisikan secara efisien. Pada klasifikasi multiclass, terdapat batasan ketika output layer memiliki lebih dari 1 neuron. Dalam hal ini, fungsi aktivasi yang umum digunakan adalah fungsi *softmax*. Langkah terakhir adalah *fully connected layer*. Yang bertujuan dalam langkah ini adalah untuk mengklasifikasikan *image* atau gambar. (Zufar, M., & Setiyono, B. 2016).

G. Tensorflow

Tensorflow lite adalah *library machine learning* yang dibuat oleh google khusus untuk perangkat *mobile*. Ini memungkinkan pada mesin untuk “belajar” di perangkat dengan latensi rendah dan ukuran *binary* yang kecil.

Tensorflow merupakan pustaka matematika simbolis yang dibuat berdasarkan dataflow dan pemrograman diferensial. Pustaka ini digunakan untuk keperluan penelitian dan produksi oleh google *tensorflow lite* terdiri dari runtime di mana dapat menjalankan model yang sudah ada sebelumnya, dan serangkaian alat yang dapat digunakan untuk mempersiapkan model untuk digunakan pada perangkat seluler dan perangkat yang disematkan. Sebelum dirancang untuk model pelatihan. Terlebih kita melatih model pada mesin bertenaga lebih tinggi, lalu mengonversi model tersebut ke format *.TFLITE*, dari mana model tersebut dimuat ke penerjemah seluler. (Widodo, B. 2016).



Gambar 2.5 Flowchart tensorflow lite

Flowchart pada gambar diatas menjelaskan tentang *image braille* ke abjad yang dimana inputannya berupa gambar huruf *braille* setelah itu huruf *braille* tersebut apakah terdapat di dataset model *tensorflow lite* jika tidak maka akan kembali ke inputan gambar huruf *braille*, jika iya maka akan mendapatkan label yang ada di model kemudian outputnya berupa huruf abjad.

H. Hashmap

Hashmap adalah salah satu kelas yang termasuk dalam kerangka kerja *collections* API yang menyediakan struktur data untuk menyimpan pasangan kunci-nilai. *Hashmap* digunakan *hashmap* untuk menyimpan data dan memberikan kinerja pencarian dan pengambilan yang cepat. *Hashmap* digunakan

untuk menyimpan data dalam bentuk pasangan kunci-nilai. Setiap kunci dan nilai dalam *hashmap* dapat menjadi objek dari jenis apa pun.

Hashmap adalah sebuah konsep abstrak dari sebuah kelas. *Hashmap* memetakan atribut dan *value* dalam sebuah *array*. Secara teknis *hashmap* menyimpan pemetaan antara teks dan teks terenkrip, dalam bentuk *key* : *value*, dimana *key* adalah teks dan *value* adalah nilai dari teks yang terenkrip. Dengan mendefinisikan *key* dan *value*, tidak diperlukan lagi kelas atau objek untuk memasukkan data ke dalam sebuah list. Tidak perlu untuk menulis tipe data, karena yang perlu diperhatikan adalah *key* dan *value* saja. Hal ini memberi dampak positif karena tidak perlu lagi membuat kelas tambahan. Dalam *list* yang tidak standar pun penggunaan *hashmap* bisa dilakukan. Dalam tulisan ini didapatkan bahwa dengan menggunakan *hashmap* ukuran *file installer* untuk android menjadi lebih kecil. Selain itu dalam penulisan kode juga lebih sederhana karena tidak memerlukan adapter dan kelas baru. (Wibowo, A, 2017)

Berikut adalah beberapa karakteristik *hashmap* dalam pemrograman java:

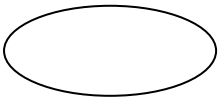
1. Pasangan kunci-nilai: *hashmap* menyimpan data dalam bentuk pasangan kunci-nilai. Kunci digunakan untuk mengakses nilai terkait.
2. *Unordered*: *hashmap* tidak menjamin urutan elemen-elemen yang disimpan. Jadi, ketika anda mengambil elemen dari *hashmap*, anda tidak akan mendapatkan elemen tersebut dalam urutan tertentu.

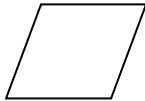
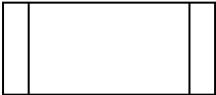
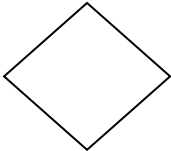
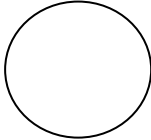
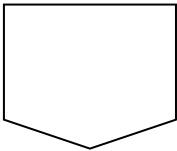
3. Tidak memperbolehkan kunci duplikat: hashmap tidak memperbolehkan kunci duplikat. Jika Anda mencoba menambahkan elemen dengan kunci yang sudah ada, nilai yang lama akan digantikan oleh nilai yang baru.
4. *Null values and null keys*: hashmap memungkinkan satu kunci bernilai *null* dan beberapa nilai bernilai *null*.
5. Implementasi hashmap: hashmap menggunakan hashmap untuk menyimpan elemen-elemen. Ini memungkinkan pencarian dan pengambilan elemen dalam waktu konstan ($O(1)$), dengan asumsi fungsi *hash* baik dan tidak terjadi tabrakan (*collision*) berlebihan.

I. Flowchart

Flowchart adalah adalah sebuah diagram dengan simbol-simbol tertentu yang merinci urutan proses dan hubungan antara satu proses (instruksi) dan proses lainnya ke dalam suatu program. Berikut ini adalah beberapa contoh simbol yang digunakan dalam suatu *flowchart* :

Tabel 2. 1 Simbol *Flowchart*

Simbol	Nama	Fungsi
	<i>Terminator</i>	Awal dan akhir sebuah sistem
	<i>Flow Line</i>	Aliran suatu sistem
	<i>Preparation</i>	Proses inisialisasi/pemberian pada variabel
	<i>Process</i>	Proses yang terjadi

	<i>Input/output data</i>	Proses input/output data
	<i>Predefined process (sub program)</i>	Subprogram
	<i>Decision</i>	Percabangan yang memberikan pilihan untuk langkah selanjutnya
	<i>On page connector</i>	Menghubungkan <i>flowchart</i> yang berada pada satu halaman
	<i>Off page connector</i>	Menghubungkan flowchart yang berada pada halaman berbeda

J. UML (*Unified Modelling Language*)

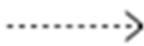
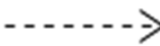
Unified Modeling Language(UML) adalah sebuah bahasa pemodelan perangkat lunak yang telah distandardisasi sebagai media penulisan cetak biru (*blueprints*) perangkat lunak (*Pressman*). UML bisa saja digunakan untuk visualisasi, spesifikasi, konstruksi dan dokumentasi beberapabagian-bagian dari system yang ada dalam perangkat lunak. Dalam kata lain, seperti halnya seorang arsitek dalam membuat dokumen cetak biru yang digunakan oleh perusahaan konstruksi untuk membangun sebuah bangunan, arsitek perangkat lunak membuat diagram-diagram UML untuk membantu programmer/ develop rmembangun perangkat lunak. Untuk selanjutnya, semakin kita mengetahui beberapa kosakata yang digunakan UML, kita

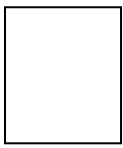
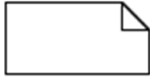
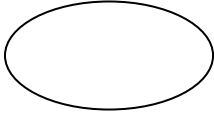
akan semakin mudah dalam memahami spesifik. Sumiati, M., Abdillah, R., Cahyo, A. (2021).

1. Use-case diagram

Use-case diagram menggambarkan secara grafis perilaku *use-case* diagram menggambarkan secara grafis perilaku perangkat lunak. Diagram ini memberikan gambaran menurut perspektif pengguna perangkat lunak. Sebuah *use-case* diagram mengandung *actor*, *use-case* dan interaksi antara *actor* dengan *use-case*.

Tabel 2. 2 Simbol *use case/ UML* diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Actor</i>	Sebagai peran yang <i>user</i> mainkan ketika berkomunikasi dengan <i>use case</i> .
2		<i>Generalization</i>	Relasi dimana objek anak (<i>descendent</i>) berbagi sikap dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
3		<i>Dependency</i>	Relasi dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen-elemen yang bergantung padanya elemen yang tidak mandiri (<i>independent</i>).
4		<i>Extend</i>	Proses bahwa <i>use case</i> target memperluas sikap dari <i>use case</i> sumber pada suatu titik yang diberikan.
5		<i>Include</i>	Proses bahwa <i>use case</i> sumber secara eksplisit.
6		<i>Association</i>	Yang menghubungkan objek dengan objek yang lain.

7		<i>System</i>	Proses paket yang menampilkan sistem secara terbatas.
8		<i>Note</i>	Elemen fisik yang ada saat aplikasi berjalan dan mewakili sumber daya komputasi.
9		<i>Collaboration</i>	Menggambarkan dari aturan-aturan yang saling bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		<i>Use Case</i>	Menjelaskan dari tahapan aksi yang dimunculkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor.

2. Activity diagram

Activity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktifitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena kita dapat memodelkan sebuah alur kerja dari satu aktifitas ke aktifitas lainnya atau dari satu aktifitas ke dalam keadaan sesaat (*state*). *Activity diagram* menggambarkan aliran aktifitas dari sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Diagram ini juga dapat menggambarkan proses *parallel* yang mungkin terjadi pada beberapa eksekusi (Ayu, S., 2022).

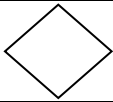
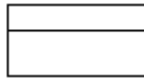
Tabel 2.3 Simbol *activity* diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antar muka saling berinteraksi satu sama lain.
2		<i>Action</i>	<i>State</i> dari sistem yang mencerminkan eksekusi dari suatu aksi.
3		<i>Initial Node</i>	Bagaimana <i>objek</i> dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana <i>objek</i> dibentuk dan dihancurkan.
5		<i>Fork Node</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran.

3. Class diagram

Class diagram merupakan struktur kelas-kelas dari suatu sistem yang memperlihatkan hubungan antar kelas dan penjelasan tiap-tiap kelas. Pada diagram ini terdapat nama kelas, atribut dan operasi kelas tersebut. Selama proses analisis, *class* diagram memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap desain, diagram ini berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat (Anwar, Y., 2021).

Tabel 2.4 Simbol *class* pada diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Generalization</i>	Hubungan dimana objek anak (objek anak) berbagi perilaku <i>objek</i> dan struktur data dengan <i>objek</i> induk (leluhur).
2		<i>Nary Association</i>	Mencoba untuk menghindari asosiasi dengan lebih dari 2 <i>objek</i> .
3		<i>Class</i>	Kumpulan dari objek-objek yang saling berbagi atribut serta operasi yang mirip.
4		<i>Collaboration</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang

			terukur bagi suatu <i>actor</i> .
5		<i>Realization</i>	Operasi yang benar-benar dilakukan oleh suatu <i>objek</i> .
6		<i>Dependency</i>	Hubungan di mana perubahan yang terjadi pada elemen independen mempengaruhi elemen yang bergantung pada elemen dependen.
7		<i>Association</i>	Apa yang menghubungkan antaran objek satu dengan objek lainnya.

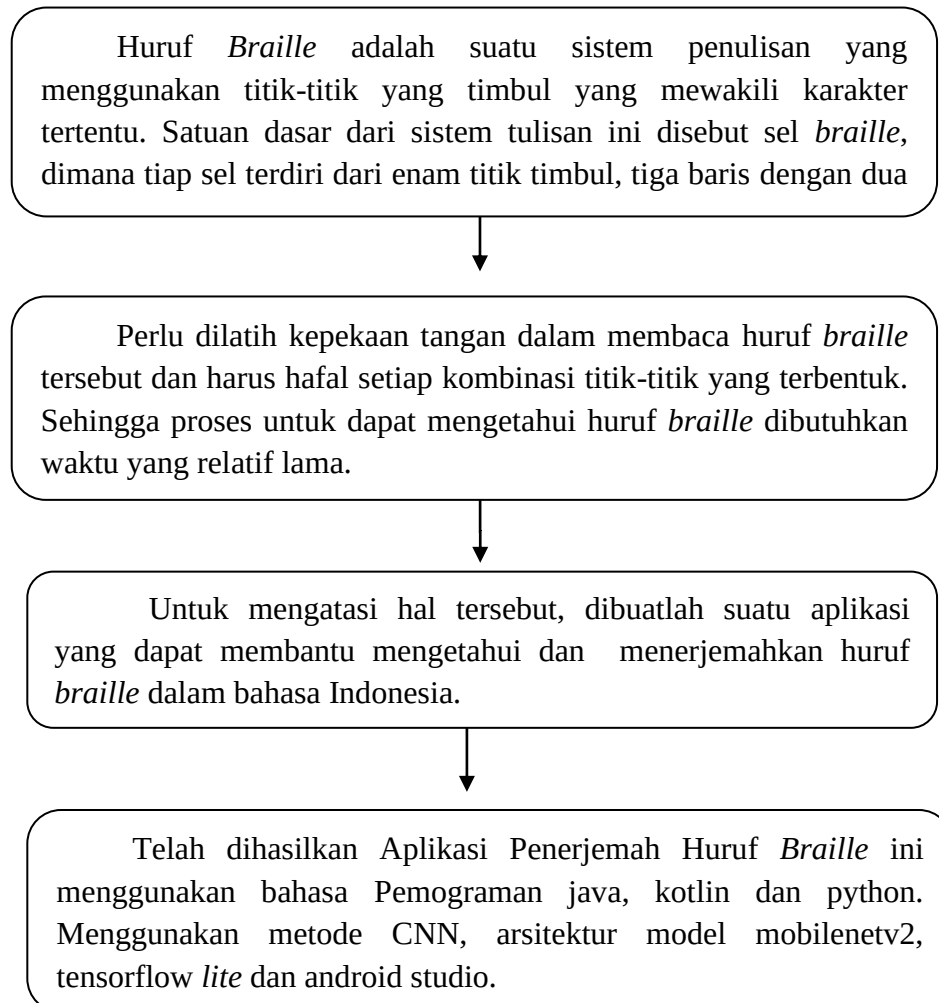
K. Penelitian Terdahulu

1. Herlambang, F. M., Hermana, N. A., & Putra, R. K. (2020) penelitiannya yang berjudul “*Pengenalan Karakter Braille dengan Metode Cellular Neural Network*”. Perbedaan pada penelitian saya yaitu tidak menggunakan arsitektur model mobilenetv2, jumlah dataset dan *class* yang di training, dan tingkat akurasi. Persamaan pada penelitian saya yaitu metode yang digunakan CNN dan huruf *braille*.
2. Ramiati., Aulia, S., & Lifwarda. (2020) penelitiannya yang berjudul “*Aplikasi Identifikasi Huruf Braille Menggunakan Computer Vision Berbasis Raspberry Pi*”. Perbedaan pada penelitian saya yaitu, menggunakan metode Jaringan Saraf Tiruan (JST) sedangkan metode yang saya gunakan metode CNN, berbasis raspberry pi sedangkan aplikasi saya berbasis android, outputnya berupa teks tulisan dan suara menggunakan *scanner* dan kamera berbasis *Raspberry Pi* sedangkan penelitian saya hanya berupa huruf abjad. Persamaan pada penelitian saya yaitu huruf yang diteliti huruf *braille*.
3. Sulviyani, I., Novamizanti. L. and Atmaja, D. R.(2017) Penelitiannya yang berjudul “*Perancangan Konversi Braille ke Teks Berbasis Android*”.

Perbedaan dengan penelitian saya yaitu metode yang digunakan metode *K-nearest neighbor* (KNN) sedangkan metode yang saya gunakan metode CNN .
Persamaan dengan penelitian saya yaitu berbasis android, huruf yang diteliti huruf *braille*.

4. Anwar, Y. (2021) penelitiannya yang berjudul “*Aplikasi Penerjemah Bahasa Isyarat Indonesia ke dalam Huruf Abjad*”. Perbedaan dengan penelitian saya yaitu *object* yang diteliti bahasa isyarat sedangkan pada penelitian saya berupa huruf *braille*. Persamaan dengan penelitian saya yaitu menggunakan metode CNN, menggunakan arsitektur model mobilenetv2 dan berbasis android.

L. Kerangka Pikir



Gambar 2.6 Kerangka piker

BAB III

METODOLOGI PENELITIAN

A. Jenis Penelitian

Jenis penelitian yang digunakan yaitu pendekatan deskriptif yang menyajikan fakta dan informasi secara berkesinambungan, jujur dan akurat. Penelitian dilakukan melalui sumber data informasi dan pengetahuan tentang sistem yang diteliti, yang kemudian dicocokkan dengan suatu kemungkinan yang akan muncul dalam upaya pemecahan masalah.

B. Waktu dan Tempat Penelitian

Penelitian ini dilakukan di kota Parepare, adapun waktu yang dipergunakan untuk pelaksanaan penelitian adalah ± 5 bulan.

C. Alat dan Bahan Penelitian

1. Alat penelitian

- a. *Reglet* dan *stylus* (alat tulis *braille*)
- b. *Hardware* (Perangkat keras)

Perangkat keras yang digunakan dalam pembuatan aplikasi terjemahan bahasa isyarat ke dalam huruf abjad dapat dilihat dari tabel berikut :

Tabel 3. 1 Spesifikasi *hardware*

Spesifikasi <i>hardware</i>	
Laptop/PC	Toshiba Satelite l40-A
Processor laptop	Intel(R) CPU 1037U @1.80GHz (2 CPUs), ~1.8 GHz
RAM laptop	8 GB

c. *Software* (Perangkat lunak)

Perangkat lunak yang digunakan dalam pembuatan aplikasi terjemahan *braille* ke dalam teks bahasa indonesia dapat dilihat dari tabel sebagai berikut :

Tabel 3. 2 Spesifikasi *software*

Spesifikasi <i>software</i>	
Sistem operasi laptop	Windows 10
<i>Tool IDE</i>	Android studio
Bahasa pemrograman	Java & Kotlin

2. Bahan penelitian

Berupa data-data yang didapat dari perpustakaan, internet maupun referensi-referensi lain mengenai huruf *braille*.

D. Desain Sistem

Perancangan sistem adalah teknik untuk memecahkan masalah yang melengkapi (melalui analisis sistem) dengan cara merakit kembali komponen-komponen menjadi suatu sistem yang lengkap (Whitten et al. 2004).

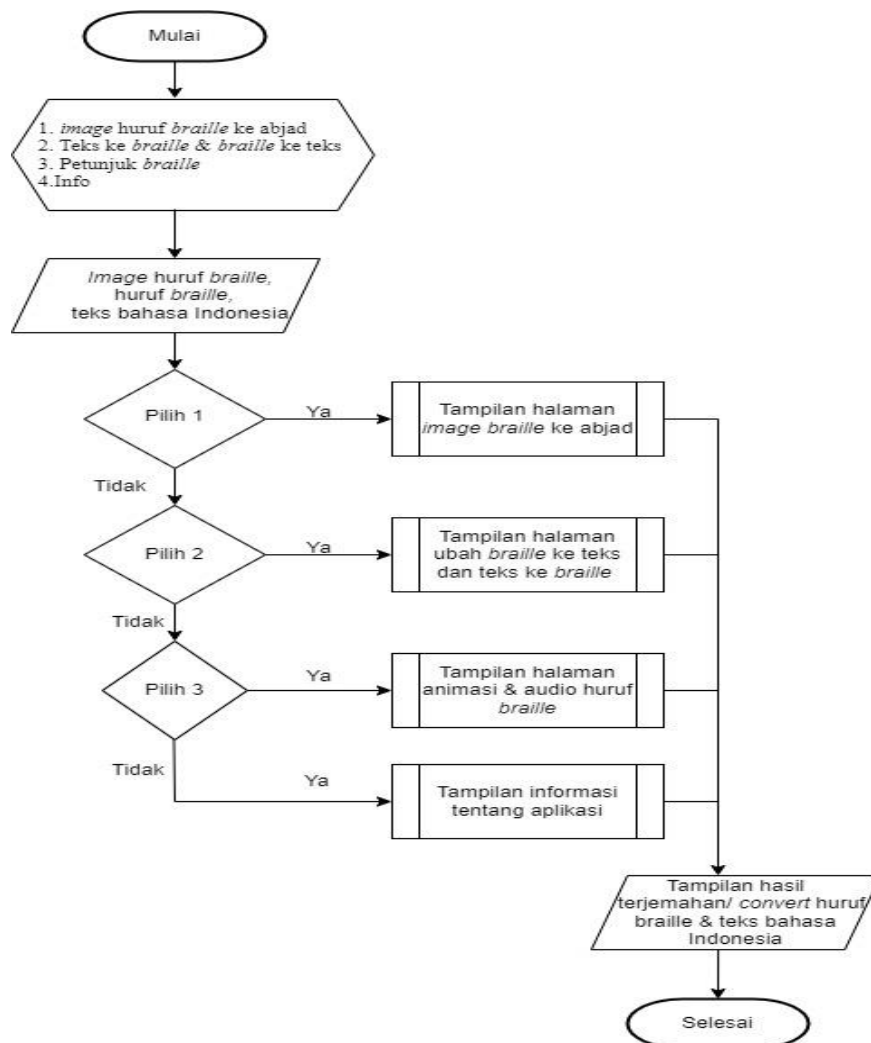
Perancangan sistem dilakukan untuk memberikan gambaran dan mempermudah dalam melakukan implementasi ataupun evaluasi terhadap sistem yang akan dibangun. Pada perancangan sistem terdapat *flowchart*, *use-case diagram*, *activity diagram*, perancangan antarmuka (*interface*), dan sistem yang berjalan serta yang diusulkan (Whitten et al, 2004).

1. Flowchart

Flowchart adalah diagram dengan simbol-simbol tertentu yang merinci urutan proses dan hubungan antara satu proses dan proses lainnya dalam suatu program.

Perancangan sistem secara umum digambarkan dengan *flowchart* seperti pada gambar 3.1.

a. Flowchart sistem



Gambar 3.1 Flowchart Sistem

Gambar *flowchart* di atas menjelaskan variabelnya yaitu *image braille* ke abjad, teks ke *braille* & *braille* ke teks, petunjuk *braille*, dan info setelah itu, inputannya berupa *image* huruf *braille*, huruf *braille*, dan teks bahasa Indonesia jika pilih menu 1 akan menampilkan tampilan halaman *image braille* ke abjad, jika pilih menu 2 maka akan menampilkan halaman ubah huruf *braille* ke teks dan teks ke *braille*, jika pilih 3 maka akan menampilkan halaman animasi & audio huruf *braille*, jika tidak pilih 3 maka akan menampilkan halaman informasi tentang aplikasi.

E. Tahap Penelitian

Tahap penelitian dilakukan dalam enam tahap, yaitu tahap penelitian, pengumpulan data, analisis, perancangan, pengujian, dan implementasi. Uraian dari keenam tahapan tersebut adalah sebagai berikut :

1. Pengumpulan data

Pada tahap ini dilakukan pengumpulan data-data yang menunjang keberhasilan dari penelitian, seperti melakukan studi literatur tentang aplikasi penerjemah huruf *braille*.

2. Analisis data

Pada tahap analisis, peneliti melakukan analisa terhadap sistem yang di terapkan sekarang berdasarkan kemudian merumuskan masalah yang menjadi pokok penelitian sehingga dapat dibuat alternatif pemecahan masalah.

3. Perancangan aplikasi

Pada tahap ini yang dilakukan perancangan *flowchart*, *UML*, dan desain dari aplikasi.

4. Pembuatan aplikasi

Aplikasi dibangun menggunakan *android studio*.

5. Pengujian aplikasi

Aplikasi yang telah dibangun akan diuji keberhasilannya dalam melakukan penerjemahan huruf *braille* ke dalam huruf abjad.

6. Implementasi

Implementasi akan dilakukan setelah pengujian terhadap aplikasi berhasil dilakukan.

F. Metode Pengujian

1. *White box testing*

White box testing, metode desain *test case* yang memanfaatkan struktur untuk mendapatkan *test case*, jenis pengujian ini digunakan untuk memprediksi detail bagaimana perangkat lunak akan bekerja hingga ke jalur logis, menguji perangkat lunak dengan kondisi fisik dan iterasi. Contoh pengujian kotak putih ini adalah untuk memperingatkan pengguna ketika mereka memasukkan kata sandi pengguna yang salah, dan untuk kesalahan tersebut, itu akan memberi pengguna informasi tentang kesalahan yang dibuat.

2. *Black box*

Semua navigasi dan tombol fasilitas dan proses program lain berjalan tanpa *error* sesuai dengan hasil pengujian yang dilakukan, namun aplikasi telah menetapkan aturan dan harus dipatuhi, karena jika diabaikan, sistem akan menolak perintah kepada pengguna yang tidak mengikuti aturan sistem yang

berjalan persyaratan untuk pemrosesan lebih lanjut yang tidak sesuai seperti kesalahan dalam memberikan informasi kepada pengguna karena data yang tidak lengkap atau tidak sesuai yang akan dimasukkan oleh sistem.

BAB IV

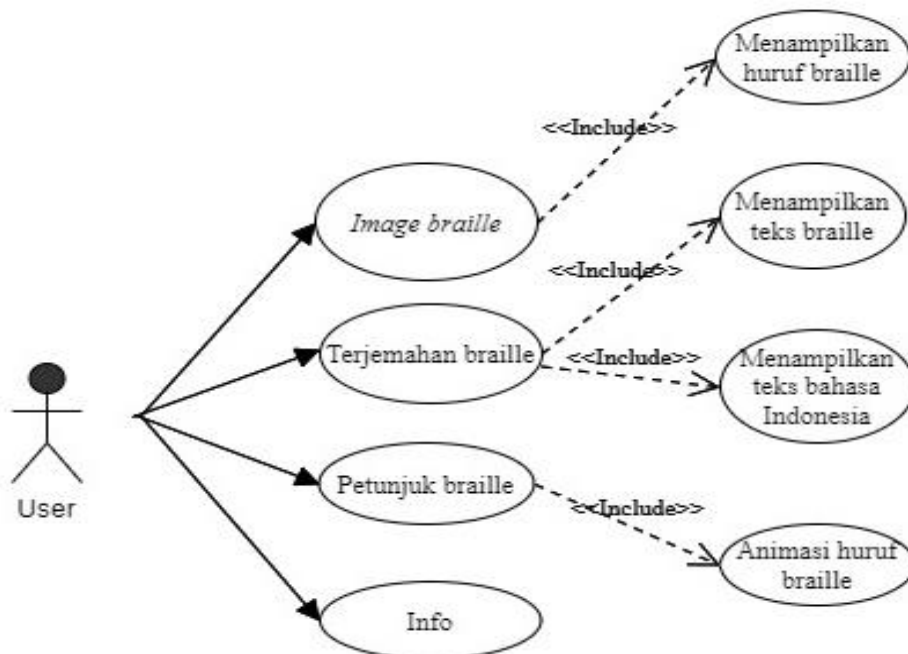
HASIL DAN PERANCANGAN SISTEM

A. Analisis sistem yang diusulkan

Berdasarkan permasalahan yang ada maka penulis membuat sebuah sistem yang akan membantu pengguna untuk mendeteksi huruf *braille*. Skema sistem ini penulis gambarkan menggunakan diagram UML yaitu diagram *use case* dan diagram *activity*.

1. *Use case diagram*

Use case diagram berfungsi melakukan kelebihan sistem jika dilihat dari sudut pandang individu yang berada di luar sistem (*actor*).



Gambar 4.1 *Use case diagram user*

Penjelasan *use case diagram*

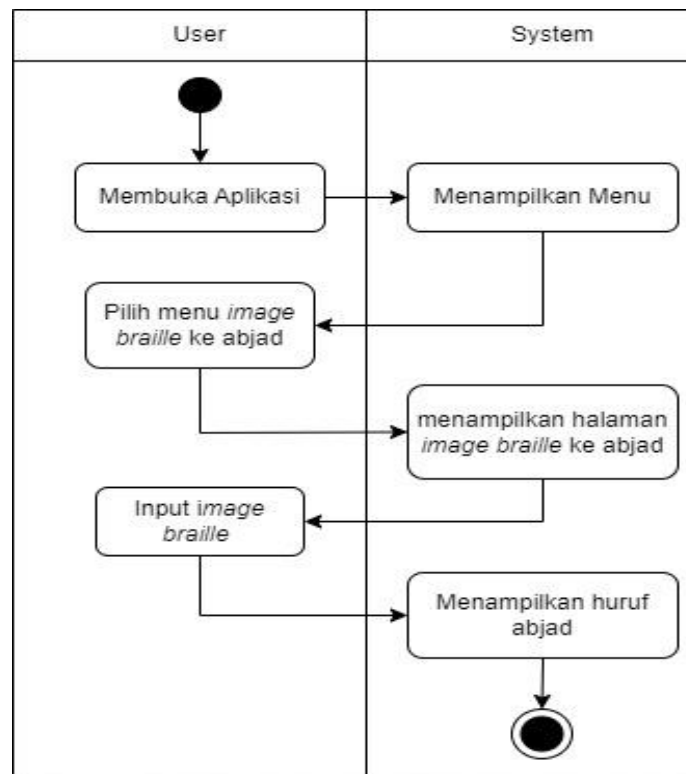
Tabel 4. 1 Penjelasan *use case diagram actor user*

Nama <i>use case</i>	Deskripsi <i>use case</i>
<i>Image</i> huruf <i>braille</i>	<i>Use case</i> ini menjelaskan bagaimana proses untuk membuka suatu aplikasi untuk mengubah gambar huruf <i>braille</i> menjadi huruf abjad .
Petunjuk <i>braille</i>	<i>Use case</i> ini pada bagian ini menampilkan halaman suatu list button huruf abjad dari A-Z.
Informasi aplikasi	<i>Use case</i> ini menjelaskan proses menampilkan tentang aplikasi.
Terjemahan <i>braille</i>	<i>Use case</i> ini menjelaskan huruf <i>braille</i> di ubah ke teks bahasa Indonesia dan mengubah teks bahasa Indonesia menjadi huruf <i>braille</i> .
Animasi huruf <i>braille</i>	<i>Use case</i> ini digunakan untuk menampilkan suatu animasi huruf <i>braille</i> .

2. Activity diagram

Activity Diagram ini menjelaskan tentang aktivitas-aktivitas yang terjadi dalam sebuah aliran proses pada sebuah sistem. Activity ini menjelaskan mengenai proses urutan yang akan dilalui oleh user dalam menjalankan sebuah aplikasi yang mulai dari halaman utama yang terdiri dari beberapa menu yaitu menu *image braille* ke teks, *braille* ke teks dan teks ke *braille*, dan menu informasi aplikasi.

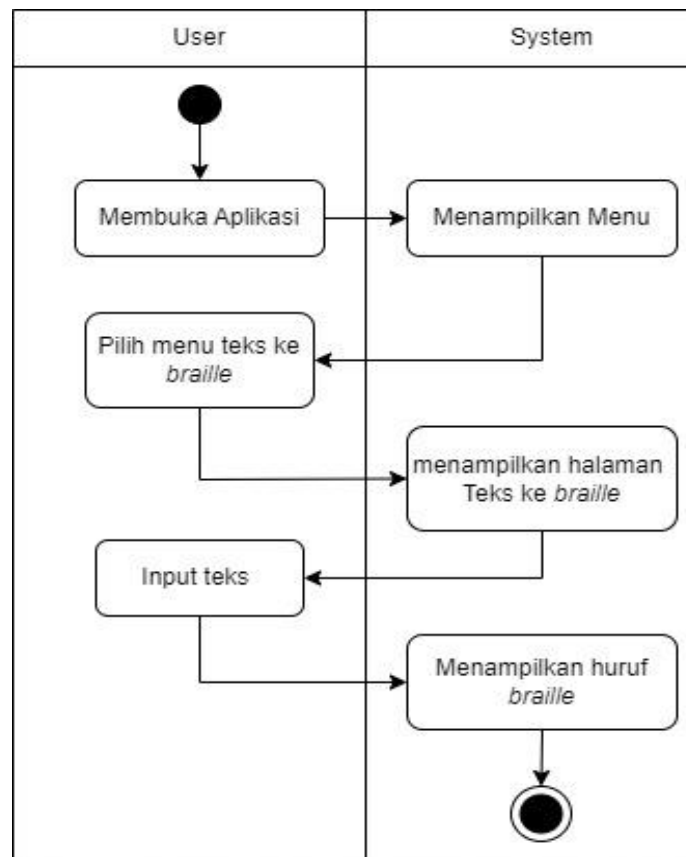
1) Activity diagram *image braille* ke teks



Gambar 4. 2 Activity diagram menu *image braille* ke abjad

Activity diagram di atas menjelaskan bahwa ketika *user* membuka aplikasi ini akan menampilkan menu kemudian user memilih menu *image braille* ke abjad setelah itu menampilkan halaman *image braille* ke abjad kemudian user menginput gambar dari galeri *smartphone* android apabila menekan tombol *load image* maka akan menampilkan huruf abjad.

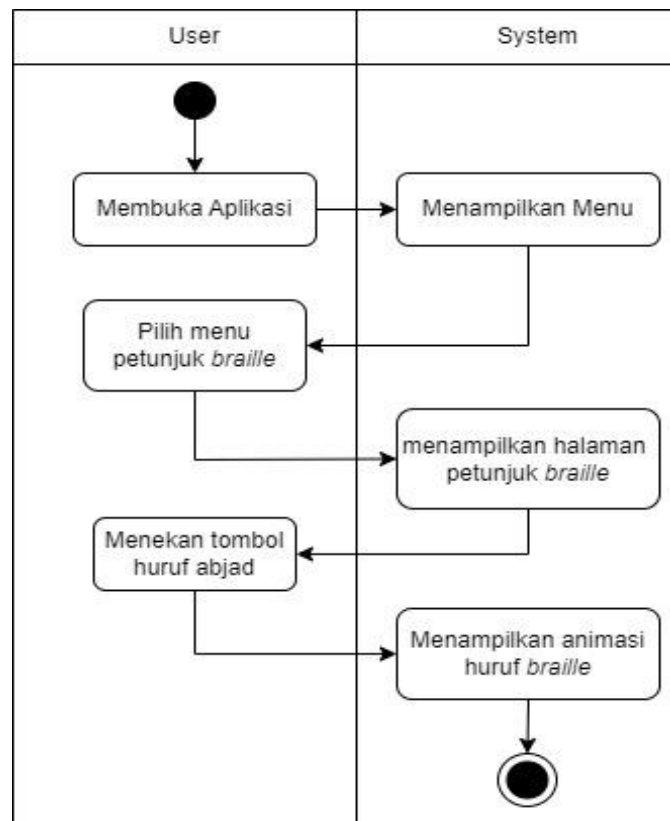
2) Activity diagram teks ke *braille*



Gambar 4.3 Activity diagram teks ke *braille*

Activity diagram di atas menjelaskan bahwa jika *user* membuka aplikasi ini akan menampilkan beberapa menu kemudian *user* memilih menu teks ke *braille* setelah itu *user* menginput teks dan apabila menekan tombol ubah akan menampilkan huruf *braille*.

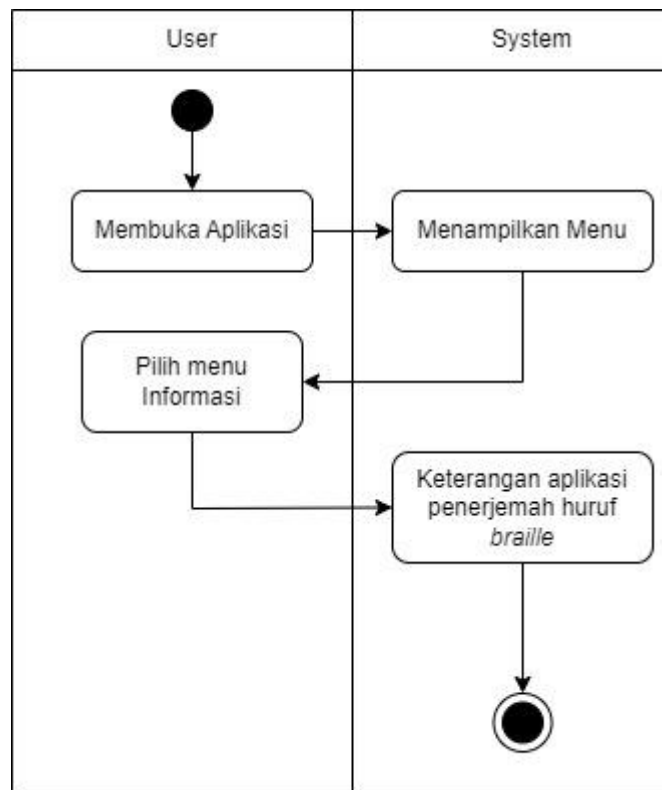
3) Activity diagram petunjuk *braille*



Gambar 4. 4 Activity diagram petunjuk *braille*

Activity diagram di atas menjelaskan bahwa jika *user* membuka aplikasi ini akan menampilkan beberapa menu kemudian *user* memilih menu petunjuk *braille* yang apabila menekan tombol huruf abjad akan menampilkan animasi huruf *braille* beserta audionya.

4) Activity diagram informasi



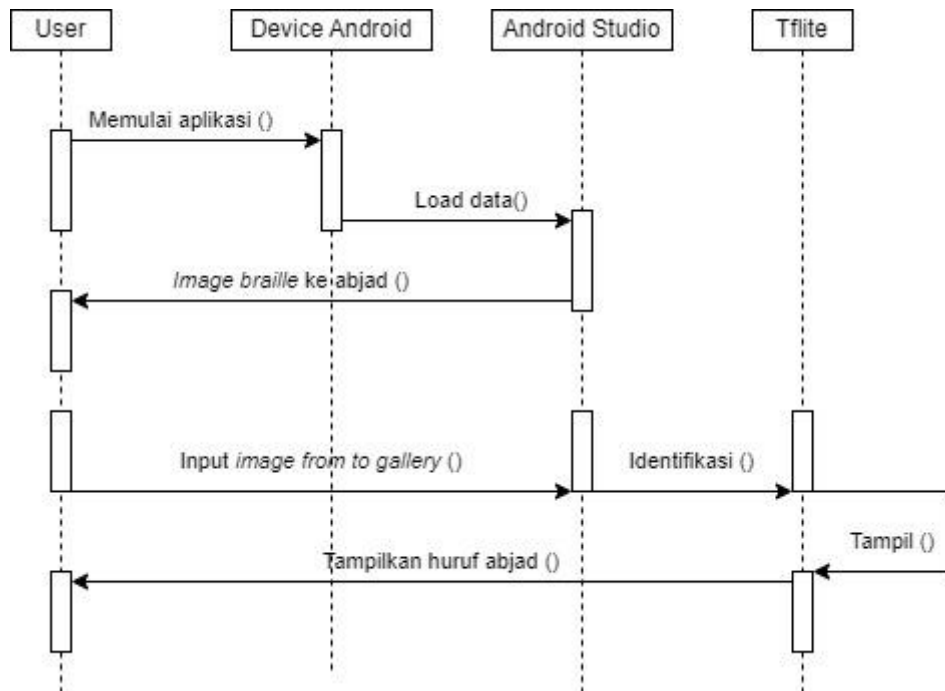
Gambar 4. 5 Activity diagram informasi

Activity diagram di atas menjelaskan bahwa jika *user* membuka aplikasi ini akan menampilkan beberapa menu kemudian *user* memilih menu informasi dan akan menampilkan keterangan aplikasi penerjemah huruf *braille*.

3. Sequence diagram

Sequence diagram adalah salah satu jenis diagram interaksi dalam UML (*Unified Modeling Language*) yang digunakan untuk menggambarkan interaksi antara objek atau komponen dalam sebuah sistem secara kronologis. Diagram ini mengilustrasikan alur pesan atau peristiwa di antara berbagai elemen sistem dalam rentang waktu tertentu.

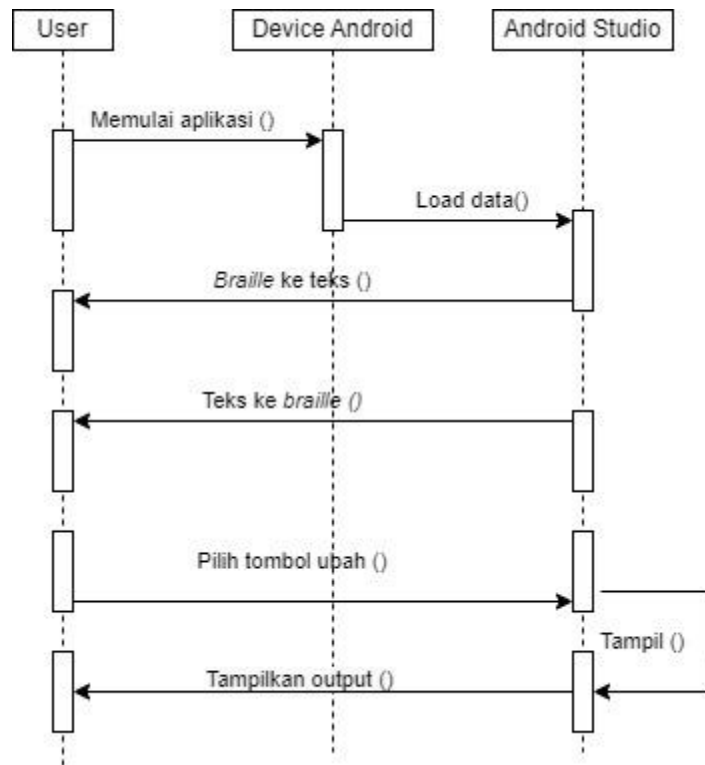
1) *Sequence diagram image braille ke abjad*



Gambar 4. 6 *Sequence diagram image braille ke abjad*

Diagram Sequence di atas menjelaskan bahwa ketika *user* memulai Aplikasi maka *device Android* yang di gunakan akan meload data untuk selanjutnya memulai aplikasi ini, setelah itu akan tampil menu *image braille ke abjad*, setelah itu user menginput *image from to gallery*, android studio mengidentifikasi pada tflite ketika mendapatkan label pada tflite maka akan tampil huruf abjad.

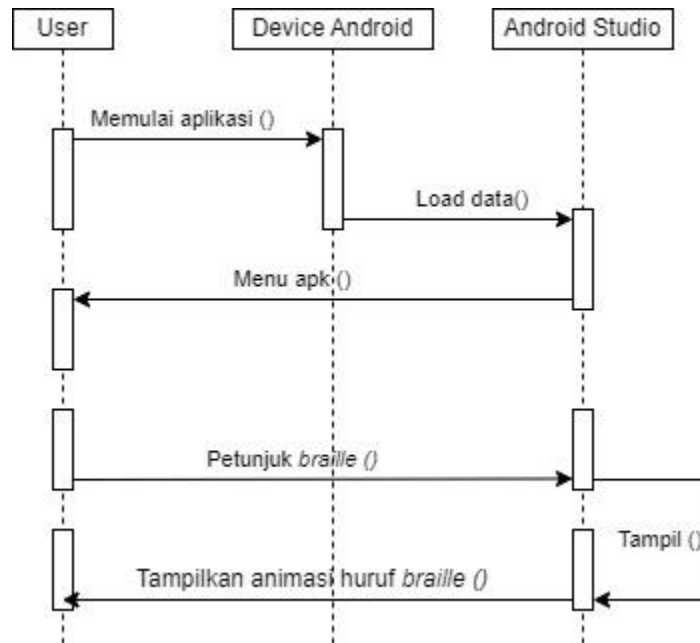
2) Sequence diagram *braille* ke teks



Gambar 4. 7 Sequence diagram *braille* ke teks

Diagram Sequence di atas menjelaskan bahwa ketika *user* memulai Aplikasi maka *device Android* yang di gunakan akan meload data untuk selanjutnya memulai aplikasi ini, setelah itu akan tampil menu *braille* ke teks & teks ke *braille* setelah itu user menginput dan menekan tombol ubah maka akan menampilkan output.

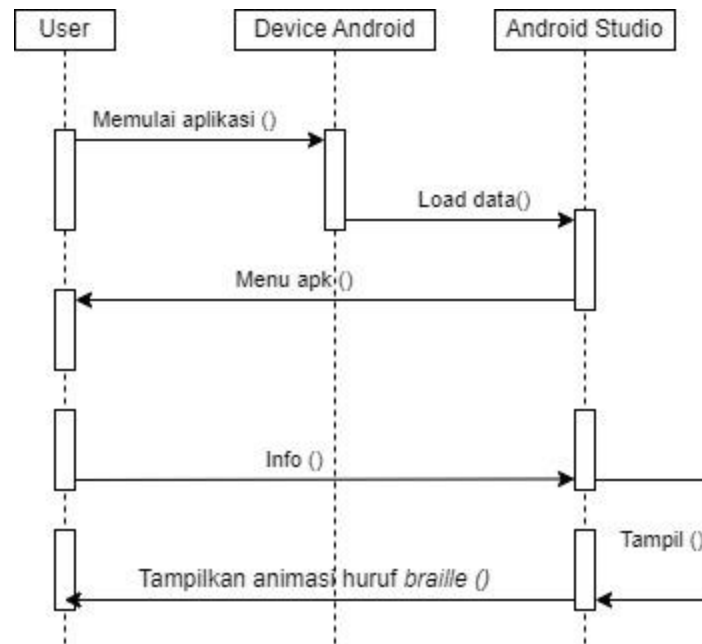
3) Sequence diagram petunjuk braille



Gambar 4.8 Sequence diagram petunjuk braille

Diagram Sequence di atas menjelaskan bahwa ketika user memulai Aplikasi maka *deviceAndroid* yang di gunakan akan meload data untuk selanjutnya memulai aplikasi ini, setelah itu akan tampil menu aplikasi dan user memilih menu petunjuk *braille* dan akan menampilkan animasi huruf *braille*.

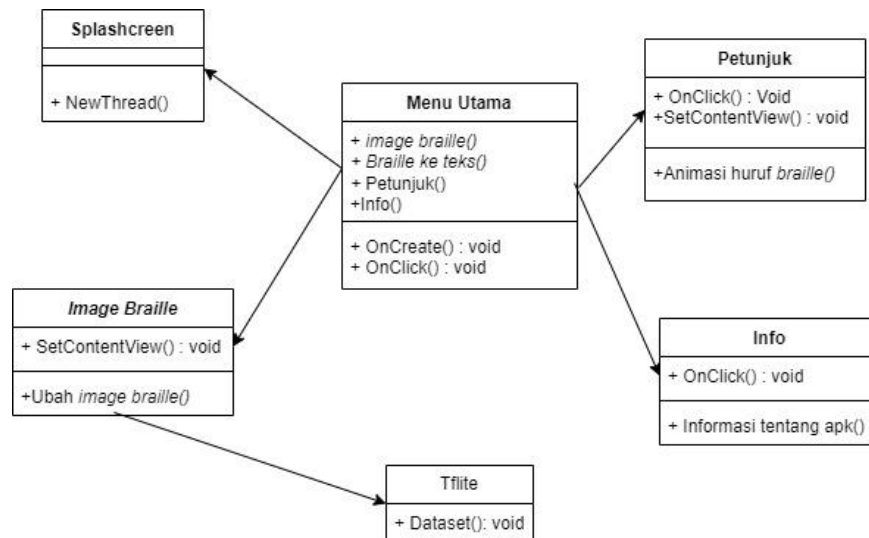
4) Sequence diagram info



Gambar 4.9 Sequence diagram info

Diagram Sequence di atas menjelaskan bahwa ketika *user* memulai Aplikasi maka *deviceAndroid* yang di gunakan akan meload data untuk selanjutnya memulai aplikasi ini, setelah itu akan tampil menu aplikasi dan user memilih menu info dan akan menampilkan informasi tentang aplikasi.

4. Class diagram



Gambar 4. 10 Class diagram sistem

Class Diagram di atas menjelaskan keterkaitan antara setiap form dan fungsi-fungsi yang ada di dalamnya, seperti ketika aplikasi di jalankan pertama kali maka yang akan tampil pertama kali adalah munculnya sebuah splashcren sebelum masuk ke menu utama.

B. Convolutional Neural Network (CNN)

Berikut proses konvolusi dengan memberikan nilai filter pada matrix.

Red							Green						
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	70	35	28	42	0		0	107	59	45	58	0	
0	45	25	30	48	0		0	76	42	45	65	0	
0	31	28	32	49	0		0	51	41	46	65	0	
0	30	30	35	50	0		0	42	41	48	64	0	
0	0	0	0	0	0		0	0	0	0	0	0	
Blue													
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	112	69	51	59	0								
0	79	48	47	63	0								
0	56	43	46	60	0								
0	47	42	46	59	0								
0	0	0	0	0	0								

Gambar4. 11 Nilai pixel RGB image huruf braille Y

Tabel 4.2 kernel 3x3

1	0	-1
1	0	-1
1	0	-1

Langkah selanjutnya dilakukan proses perhitungan disetiap *channel* dengan kernel ukuran 3x3 pada gambar 4.12. Tahapan ini dilakukan berulang dengan pergeseran kernel sebanyak 1 *strides* di setiap *channel*nya sehingga didapatkan perhitungan di setiap channel dengan nilai sebagai berikut:

a. *Channel Red*

Red

0	0	0	0	0	0
0	70	35	28	42	0
0	45	25	30	48	0
0	31	28	32	49	0
0	30	30	35	50	0
0	0	0	0	0	0

Gambar 4.12 Pixel *red*

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*0)+(0*70)+(-1*35)+ \\ &\quad (1*0)+(0*45)+(-1*25)) = -60 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*70)+(0*35)+(-1*28)+ \\ &\quad (1*45)+(0*25)+(-1*30)) = 57 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*35)+(0*28)+(-1*42)+ \\ &\quad (1*25)+(0*30)+(-1*48)) = -30 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*28)+(0*42)+(-1*0)+ \\ &\quad (1*30)+(0*48)+(-1*0)) = 58 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*70)+(-1*35)+ \\ &\quad (1*0)+(0*45)+(-1*25)+ \\ &\quad (1*0)+(0*31)+(-1*28)) = -88 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*70)+(0*35)+(-1*28)+ \\ &\quad (1*45)+(0*25)+(-1*30)+ \\ &\quad (1*31)+(0*28)+(-1*32)) = 56 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*35)+(0*28)+(-1*42)+ \\ &\quad (1*25)+(0*30)+(-1*48)+ \\ &\quad (1*28)+(0*32)+(-1*49)) = -51 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*28)+(0*42)+(-1*0)+ \\ &\quad (1*30)+(0*48)+(-1*0)+ \\ &\quad (1*32)+(0*49)+(-1*0)) = 90 \end{aligned}$$

➤ $((1*0)+(0*45)+(-1*25)+$ $(1*0)+(0*31)+(-1*28)+$ $(1*0)+(0*30)+(-1*30)) = -83$ ➤ $((1*45)+(0*25)+(-1*30)+$ $(1*31)+(0*28)+(-1*32)+$ $(1*30)+(0*30)+(-1*35)) = 9$ ➤ $((1*25)+(0*30)+(-1*48)+$ $(1*28)+(0*32)+(-1*49)+$ $(1*30)+(0*35)+(-1*50)) = -64$ ➤ $((1*30)+(0*48)+(-1*0)+$ $(1*32)+(0*49)+(-1*0)+$ $(1*35)+(0*50)+(-1*0)) = 97$	➤ $((1*0)+(0*31)+(-1*28)+$ $(1*0)+(0*30)+(-1*30)+$ $(1*0)+(0*0)+(-1*0)) = -58$ ➤ $((1*31)+(0*28)+(-1*32)+$ $(1*30)+(0*30)+(-1*35)+$ $(1*0)+(0*0)+(-1*0)) = -6$ ➤ $((1*28)+(0*32)+(-1*49)+$ $(1*30)+(0*35)+(-1*50)+$ $(1*0)+(0*0)+(-1*0)) = -41$ ➤ $((1*32)+(0*49)+(-1*0)+$ $(1*35)+(0*50)+(-1*0)+$ $(1*0)+(0*0)+(-1*0)) = 67$
--	---

Setelah perhitungan di *channel red* maka didapatkanlah hasil pada tabel dibawah ini:

Tabel 4. 3 Hasil tahapan perhitungan konvolusi *channel red*

-60	57	-30	58
-88	56	-51	90
-83	9	-64	97
-58	-6	-41	67

b. *Channel green*

0	0	0	0	0	0
0	107	59	45	58	0
0	76	42	45	65	0
0	51	41	46	65	0
0	42	41	48	64	0
0	0	0	0	0	0

Gambar4.13 Pixel *green*

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*0)+(0*107)+(-1*59)+ \\ &\quad (1*0)+(0*76)+(-1*42)) = -101 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*107)+(0*59)+(-1*45)+ \\ &\quad (1*76)+(0*42)+(-1*45)) = 93 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*59)+(0*45)+(-1*58)+ \\ &\quad (1*42)+(0*45)+(-1*65)) = -22 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*45)+(0*58)+(-1*0)+ \\ &\quad (1*45)+(0*65)+(-1*0)) = 90 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*107)+(-1*59)+ \\ &\quad (1*0)+(0*76)+(-1*42)+ \\ &\quad (1*0)+(0*51)+(-1*41)) = -142 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*107)+(0*59)+(-1*45)+ \\ &\quad (1*76)+(0*42)+(-1*45)+ \\ &\quad (1*51)+(0*41)+(-1*46)) = 98 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*59)+(0*45)+(-1*58)+ \\ &\quad (1*42)+(0*45)+(-1*65)+ \\ &\quad (1*41)+(0*46)+(-1*65)) = -46 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*45)+(0*58)+(-1*0)+ \\ &\quad (1*45)+(0*65)+(-1*0)+ \\ &\quad (1*46)+(0*65)+(-1*0)) = 136 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*76)+(-1*42)+ \\ &\quad (1*0)+(0*51)+(-1*41)+ \\ &\quad (1*0)+(0*42)+(-1*41)) = -124 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*76)+(0*42)+(-1*45)+ \\ &\quad (1*51)+(0*41)+(-1*45)+ \\ &\quad (1*42)+(0*41)+(-1*48)) = 30 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*42)+(0*45)+(-1*65)+ \\ &\quad (1*41)+(0*46)+(-1*65)+ \\ &\quad (1*41)+(0*48)+(-1*64)) = -70 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*45)+(0*65)+(-1*0)+ \\ &\quad (1*46)+(0*65)+(-1*0)+ \\ &\quad (1*48)+(0*64)+(-1*0)) = 139 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*51)+(-1*41)+ \\ &\quad (1*0)+(0*42)+(-1*41)+ \\ &\quad (1*0)+(0*0)+(-1*0)) = -82 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*51)+(0*41)+(-1*46)+ \\ &\quad (1*42)+(0*41)+(-1*48)+ \\ &\quad (1*0)+(0*0)+(-1*0)) = 9-1 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*41)+(0*46)+(-1*65)+ \\ &\quad (1*41)+(0*48)+(-1*64)+ \\ &\quad (1*0)+(0*0)+(-1*0)) = -47 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*46)+(0*65)+(-1*0)+ \\ &\quad (1*48)+(0*64)+(-1*0)+ \\ &\quad (1*45)+(0*0)+(-1*0)) = 94 \end{aligned}$$

Setelah perhitungan di *channel green* maka didapatkanlah hasil pada tabel dibawah ini:

Tabel 4.3 Hasil tahapan perhitungan konvolusi *channel green*

-101	93	-22	90
-142	98	-46	136
-124	30	-70	139
-82	-1	-47	94

c. *Channel blue*

0	0	0	0	0	0
0	112	69	51	59	0
0	79	48	47	63	0
0	56	43	46	60	0
0	47	42	46	59	0
0	0	0	0	0	0

Gambar4.14 Pixel *blue*

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*0)+(0*112)+(-1*69)+ \\ &\quad (1*0)+(0*79)+(-1*48)) = -117 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*112)+(0*69)+(-1*51)+ \\ &\quad (1*79)+(0*48)+(-1*47)) = 93 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*69)+(0*51)+(-1*59)+ \\ &\quad (1*48)+(0*47)+(-1*63)) = -5 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*0)+(-1*0)+ \\ &\quad (1*51)+(0*59)+(-1*0)+ \\ &\quad (1*47)+(0*63)+(-1*0)) = 98 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*0)+(0*112)+(-1*69)+ \\ &\quad (1*0)+(0*79)+(-1*48)+ \\ &\quad (1*0)+(0*56)+(-1*43)) = -160 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*112)+(0*69)+(-1*51)+ \\ &\quad (1*79)+(0*48)+(-1*47)+ \\ &\quad (1*56)+(0*43)+(-1*46)) = 103 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*69)+(0*51)+(-1*59)+ \\ &\quad (1*48)+(0*47)+(-1*63)+ \\ &\quad (1*43)+(0*46)+(-1*60)) = -22 \end{aligned}$$

$$\begin{aligned} &\triangleright ((1*51)+(0*59)+(-1*0)+ \\ &\quad (1*47)+(0*63)+(-1*0)+ \\ &\quad (1*46)+(0*60)+(-1*0)) = 144 \end{aligned}$$

➤ $((1*0)+(0*79)+(-1*48)+$ $(1*0)+(0*56)+(-1*43)+$ $(1*0)+(0*47)+(-1*42)) = -133$ ➤ $((1*79)+(0*48)+(-1*47)+$ $(1*56)+(0*43)+(-1*46)+$ $(1*47)+(0*42)+(-1*46)) = 43$ ➤ $((1*48)+(0*47)+(-1*63)+$ $(1*43)+(0*46)+(-1*60)+$ $(1*42)+(0*46)+(-1*59)) = -49$ ➤ $((1*47)+(0*63)+(-1*0)+$ $(1*46)+(0*60)+(-1*0)+$ $(1*46)+(0*59)+(-1*0)) = 139$	➤ $((1*0)+(0*56)+(-1*43)+$ $(1*0)+(0*47)+(-1*42)+$ $(1*0)+(0*0)+(-1*0)) = -85$ ➤ $((1*56)+(0*43)+(-1*46)+$ $(1*47)+(0*42)+(-1*46)+$ $(1*0)+(0*0)+(-1*0)) = 11$ ➤ $((1*43)+(0*46)+(-1*60)+$ $(1*42)+(0*46)+(-1*59)+$ $(1*0)+(0*0)+(-1*0)) = -34$ ➤ $((1*46)+(0*60)+(-1*0)+$ $(1*46)+(0*59)+(-1*0)+$ $(1*0)+(0*0)+(-1*0)) = 92$
---	---

Setelah perhitungan di *channel blue* maka didapatkanlah hasil pada tabel dibawah ini:

Tabel 4. 4 Hasil tahapan perhitungan konvolusi *channel blue*

-117	93	-5	98
-160	103	-22	144
-133	43	-49	139
-85	11	-34	92

Setelah melakukan tahapan konvolusi disetiap layer maka didapatkanlah nilai *red*, *green* dan *blue* pada tabel dibawah ini.

Red

-60	57	-30	58
-88	56	-51	90
-83	9	-64	97
-58	-6	-41	67

Green

-101	93	-22	90
-142	98	-46	136
-124	30	-70	139
-82	-1	-47	94

Red

-117	93	-5	98
-160	103	-22	144
-133	43	-49	139
-85	11	-34	92

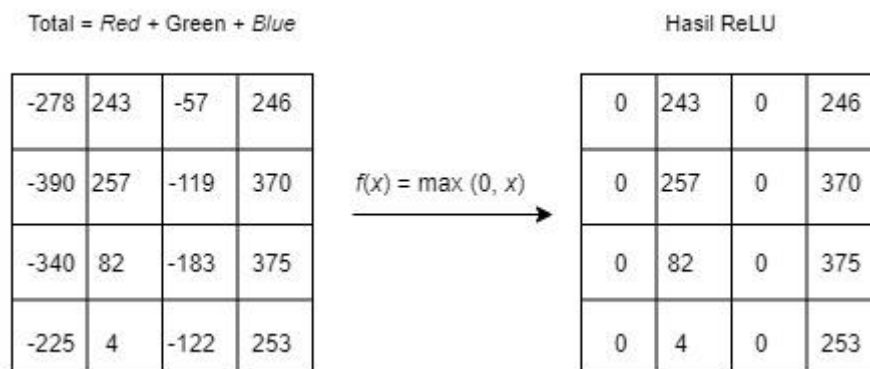
Gambar 4. 11 Hasil perhitungan layer konvolusi *channel red*, *green*, *blue*

Setiap nilai dari *channel* yang sudah melalui tahapan perhitungan pada gambar 4. 11 akan dijumlahkan sehingga menghasilkan output total seperti pada tabel 4. 4.

Tabel 4. 5 Total = Red + green + Blue

-278	243	-57	246
-390	257	-119	370
-340	82	-183	375
-225	4	-122	253

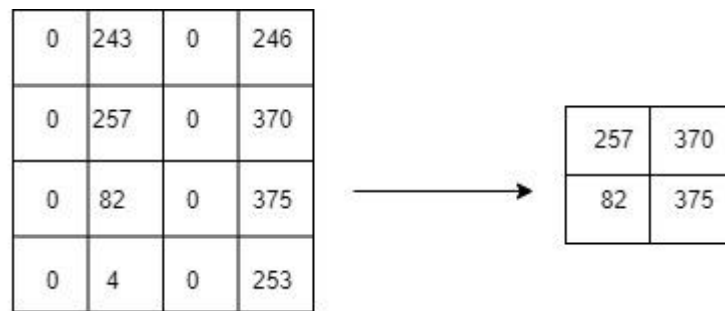
Dengan *activation* ReLu maka setiap nilai yang negative akan diubah menjadi 0 sehingga didapatkanlah hasil *convolutional* seperti pada gambar 4.12 berikut:



Gambar 4. 12 Hasil *convolutional*

d. *Pooling layer*

Dari hasil pada gambar 4. 12 *output* tahapan konvolusi akan dijadikan *input* pada *pooling layer*, pada tahapan ini menggunakan *maxpooling* dengan kernel size 2x2 dengan *stride* 2 dengan output dari tahapan konvolusi yaitu 4x4x32. Ilustrasi proses *maxpooling* ditunjukkan seperti gambar dibawah ini:



Gambar 4. 13 *Maxpooling*

Gambar 4.13 merupakan *maxpooling* dimana tahapan ini dilakukan dengan kernel size 2x2, dari nilai kernel ini akan di ambil nilai tertinggi sehingga didapatkanlah nilai pertama yaitu 257. Langkah selanjutnya dilakukan pergeseran kernel sebanyak 2 karena menggunakan *strides* 2. Sehingga didapatkanlah nilai kernel 2x2 pada matrix selanjutnya dengan nilai 0, 246, 0 dan 370 dimana setelah dilakukan *maxpooling*. Sehingga didapatkan *output* pada *layer maxpooling* yaitu 2x2x32 dengan nilai 257, 370, 81 dan 357 yang dimana *output* ini akan dijadikan inputan pada *dropout layer*.

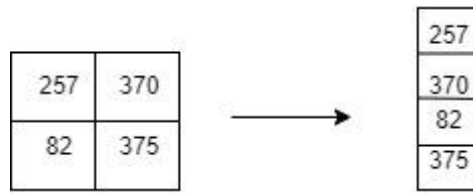
e. *Dropout Layer*

Dropout berfungsi untuk mengurangi kompleksitas model yang telah dibangun dengan membuang *neural network* yang tidak terpakai.

Ukuran model akan tetap 2x2x32 tidak ada perubahan.

f. *Flatten Layer*

Flatten merupakan tahapan mengubah *matrix/vector* 1 dimensi. Dari output dropout layer sebesar 2x2x32 akan didapatkan bentuk vector baru.



Gambar4. 19 Ilustrasi proses *flatten*

Gambar diatas merupakan proses *flatten* dimana matrix 2x2 diubah menjadi matrix 4x1 atau vector. *Output* pada tahapan ini adalah vector 1 dimensi yang akan digunakan sebagai *input* di *fully connected layer* (*dense*).

g. *Fully Connected layer*

Perhitungan pada setiap hidden layer sebagai berikut:

$$J_1 = (257*0,1) + (370*0,1) + (82*0,1) + (375*0,1) = 108,4$$

$$J_2 = (257*0,4) + (370*0,4) + (82*0,4) + (375*0,4) = 433,6$$

$$J_3 = (257*0,5) + (370*0,5) + (82*0,5) + (375*0,5) = 542$$

$$J_4 = (257*0,3) + (370*0,3) + (82*0,3) + (375*0,3) = 325,2$$

Hidden layer pertama di ilustrasikan dengan J_1, J_2, J_3, J_4 Dimana setiap neuron dikalikan dengan nilai *weight* yang berbeda-beda. Setiap neuron dari J_1, J_2, J_3, J_4 ini akan dikalikan lg dengan *weight* yang berbeda-beda untuk menghasilkan nilai H_1, H_2, H_3, H_4 sebagai *hidden layer* selanjutnya.

$$H_1 = (108,4*0,3) + (433,6*0,3) + (542*0,3) + (325,2*0,3) = 422,76$$

$$H_2 = (108,4*0,5) + (433,6*0,5) + (542*0,5) + (325,2*0,5) = 704,06$$

$$H_3 = (108,4*0,2) + (433,6*0,2) + (542*0,2) + (325,2*0,2) = 281,84$$

$$H_4 = (108,4*0,4) + (433,6*0,4) + (542*0,4) + (325,2*0,4) = 563,68$$

Yang setiap *neuron* dari H_1, H_2, H_3, H_4 ini akan dikalikan lgi dengan *weight* yang berbeda-beda untuk menghasilkan o_1 dan o_2 sebagai *hidden layer* selanjutnya.

$$O_1 = (422,76 * 0,1) + (704,06 * 0,1) + (281,84 * 0,1) + (563,68 * 0,1) = 1973,28$$

$$O_2 = (422,76 * 0,3) + (704,06 * 0,3) + (281,84 * 0,3) + (563,68 * 0,3) = 591,864$$

Langkah selanjutnya adalah perhitungan *softmax* dengan rumus eksponensial o_2 . Tahapan ini dilakukan 2 kali baik pada o_1 dan o_2 dengan rumus dan ilustrasi sebagai berikut:

$$s(O_i) = \frac{e^{O_i}}{\sum_j^n e^j}$$

$$s(O_1) = \frac{e^{1973,28}}{e^{1973,28} + e^{591,864}} = 0,8$$

$$s(O_2) = \frac{e^{591,864}}{e^{1973,28} + e^{591,864}} = 0,2$$

$$s(O_1) + s(O_2) = 1$$

$$0,8 + 0,2 = 1$$

Hasil dari perhitungan *softmax* dengan rumus eksponensial yaitu 0,8 dan 0,2.

C. Rancangan *Training Model*

Proses *training* diawali dengan membuat model dasar dengan metode *image processing* atau *image classifier*. Model yang digunakan adalah MobileNetV2. Arsitektur MobileNetV2 dipilih karena model ini mampu memiliki nilai akurasi yang lebih tinggi namun dengan jumlah komputasi yang lebih

rendah, sehingga sangat tepat diaplikasikan pada perangkat *mobile* atau yang memiliki sumber daya terbatas. Dalam perancangan model ada beberapa tahapan yang dilakukan yaitu sebagai berikut:

a. Pengumpulan Dataset

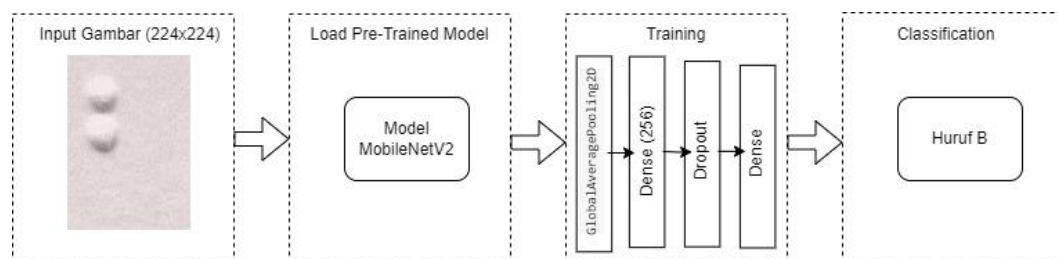
Dataset yang digunakan adalah sekumpulan data gambar yang diambil secara langsung menggunakan kamera *smartphone*. Dataset *image* yang digunakan berjumlah 8664 image dengan jumlah *class* 27.

b. *Preprocessing*

Preprocessing merupakan tahap mengolah data gambar yang telah dikumpulkan untuk dilakukan penyesuaian. Pada *preprocessing* terdapat beberapa tahap yaitu: pelabelan, *resizing* atau mengubah ukuran gambar, dan *split* dataset.

c. *Training* model

Rancangan arsitektur yang digunakan dalam pelatihan model ini terdapat pada gambar 4.2.



Gambar 4.20 Rancangan *Training* Model

Rancangan *training* model pada gambar 4.2 menggunakan arsitektur model MobileNetV2. Input gambar yang digunakan pada dataset berukuran sesuai dengan input MobileNetV2 yaitu 224x224 pixel. Setiap data gambar akan

memasuki tahap *training* untuk proses *feature extractor* yang berfungsi untuk mengambil fitur-fitur yang sudah ada pada citra. Ekstraksi fitur ini menggunakan arsitektur model MobileNetV2 yang sudah terlatih sebelumnya. Sehingga pada tahap ini, setiap citra yang akan diproses akan mengikuti arsitektur MobileNetV2 dalam melakukan operasi konvolusi. Setiap *convolution* akan memiliki konfigurasi yang berbeda-beda, seperti *padding*, *filter*, *channel*, dan *stride*. Proses ini merupakan tahap terpenting dalam pengolahan citra, karena pada tahap ini sebuah model akan dilatih untuk mempelajari pola-pola yang terdapat pada data citra, sehingga diharapkan dapat menghasilkan pengenalan gambar yang akurat dengan tingkat akurasi yang tinggi. Untuk *training* model ini dengan beberapa parameter untuk mendapat hasil *training* yang digunakan adalah sebagai berikut :

Tabel 4.2 Parameter *Training* Model

Parameter	Nilai
<i>Data epoch</i>	50
<i>Batch Size</i>	32

Training dilakukan beberapa percobaan dengan parameter berbeda yaitu data epoch untuk melakukan 50 putaran proses *training* pada *dataset* dan *batch size* 32 untuk jumlah sampel yang akan ditraining. Hingga kemudian mendapatkan hasil kemudian mendapatkan hasil hingga *accuracy* mencapai 0.9183 dengan nilai loss 0.2284.

```

271/271 [=====] - 462s 2s/step - loss: 0.2721 - accuracy: 0.9102 - val_loss: 0.3981 - val_accuracy: 0.8922
Epoch 35/50
271/271 [=====] - 483s 2s/step - loss: 0.2720 - accuracy: 0.9054 - val_loss: 0.4076 - val_accuracy: 0.8851
Epoch 36/50
271/271 [=====] - 449s 2s/step - loss: 0.2894 - accuracy: 0.8994 - val_loss: 0.3797 - val_accuracy: 0.8970
Epoch 37/50
271/271 [=====] - 477s 2s/step - loss: 0.2758 - accuracy: 0.9058 - val_loss: 0.3804 - val_accuracy: 0.8891
Epoch 38/50
271/271 [=====] - 469s 2s/step - loss: 0.2569 - accuracy: 0.9111 - val_loss: 0.3645 - val_accuracy: 0.9002
Epoch 39/50
271/271 [=====] - 451s 2s/step - loss: 0.2517 - accuracy: 0.9112 - val_loss: 0.3555 - val_accuracy: 0.8899
Epoch 40/50
271/271 [=====] - 430s 2s/step - loss: 0.2454 - accuracy: 0.9152 - val_loss: 0.3695 - val_accuracy: 0.8922
Epoch 41/50
271/271 [=====] - 426s 2s/step - loss: 0.2478 - accuracy: 0.9147 - val_loss: 0.3717 - val_accuracy: 0.8994
Epoch 42/50
271/271 [=====] - 457s 2s/step - loss: 0.2499 - accuracy: 0.9149 - val_loss: 0.3783 - val_accuracy: 0.8962
Epoch 43/50
271/271 [=====] - 462s 2s/step - loss: 0.2510 - accuracy: 0.9108 - val_loss: 0.3839 - val_accuracy: 0.8954
Epoch 44/50
271/271 [=====] - 421s 2s/step - loss: 0.2424 - accuracy: 0.9176 - val_loss: 0.3652 - val_accuracy: 0.8994
Epoch 45/50
271/271 [=====] - 458s 2s/step - loss: 0.2322 - accuracy: 0.9199 - val_loss: 0.3751 - val_accuracy: 0.9010
Epoch 46/50
271/271 [=====] - 433s 2s/step - loss: 0.2204 - accuracy: 0.9232 - val_loss: 0.3810 - val_accuracy: 0.9017
Epoch 47/50
271/271 [=====] - 425s 2s/step - loss: 0.2204 - accuracy: 0.9224 - val_loss: 0.3912 - val_accuracy: 0.8946
Epoch 48/50
271/271 [=====] - 428s 2s/step - loss: 0.2367 - accuracy: 0.9177 - val_loss: 0.4008 - val_accuracy: 0.8954
Epoch 49/50
271/271 [=====] - 440s 2s/step - loss: 0.2263 - accuracy: 0.9215 - val_loss: 0.4022 - val_accuracy: 0.8954
Epoch 50/50
271/271 [=====] - 414s 2s/step - loss: 0.2284 - accuracy: 0.9183 - val_loss: 0.3733 - val_accuracy: 0.8946

```

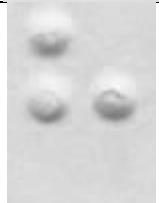
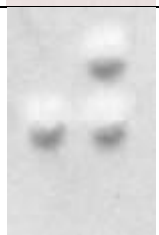
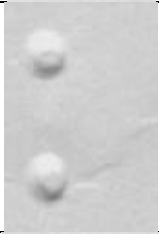
Gambar 4.3 Hasil training dataset

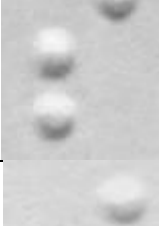
Hasil training model inilah yang akan disimpan dalam model *tensorflow*. Kemudian ubah model tersebut menjadi sebuah model *tensorflow lite* (.tflite). *TensorFlow lite* ini digunakan karena dapat mengurangi ukuran file dan meningkatkan kecepatan eksekusi tanpa mengurangi akurasi, sehingga dapat diterapkan ke ponsel android.

D. Huruf *Braille* Indonesia

Tabel 4.2 Huruf *braille*

No	Huruf	
1	A	
2	B	
3	C	
4	D	
5	E	
6	F	

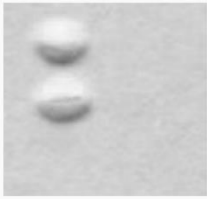
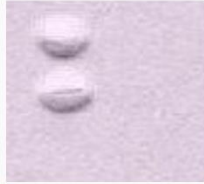
7	G		
8	H		
9	I		
10	J		
11	K		
12	L		
13	M		

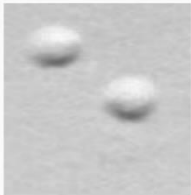
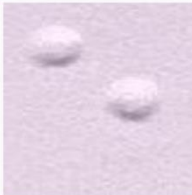
14	N		
15	O		
16	P		
17	Q		
18	R		
19	S		
20	T		
21	U		

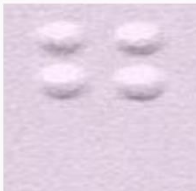
22	V		
23	W		
24	X		
25	Y		
26	Z		

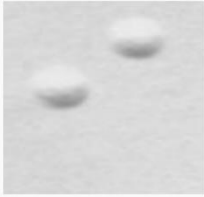
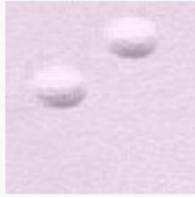
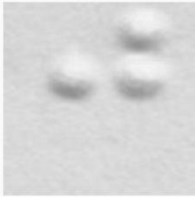
E. Pengujian huruf *braille*

Tabel 4.3 Pengujian huruf *braille*

No	Huruf	Gambar I Dengan jarak 1cm	Gambar II Berwarna dengan jarak 1cm	Gambar III Dengan jarak 5 cm
1	A	 LOAD IMAGE Output: Huruf A Berhasil, karena outputnya sudah benar yaitu huruf A	 LOAD IMAGE Output: Huruf A Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Bukan Huruf Braille Tidak berhasil
2	B	 LOAD IMAGE Output: Huruf B Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf B Berhasil dengan menggunakan background berwarna	 LOAD IMAGE Output: Bukan Huruf Braille Tidak berhasil

3	C	 LOAD IMAGE Output: Huruf C Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf C Berhasil dengan menggunakan background berwarna	 LOAD IMAGE Output: Bukan Huruf Braille Tidak berhasil
4	D	 LOAD IMAGE Output: Huruf D Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf D Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf E Tidak berhasil
5	E	 LOAD IMAGE Output: Huruf I Tidak berhasil	 LOAD IMAGE Output: Huruf E Berhasil dengan menggunakan background berwarna	 LOAD IMAGE Output: Huruf D Tidak berhasil

6	F	 LOAD IMAGE Output: Huruf F Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf I Tidak berhasil	 LOAD IMAGE Output: Bukan Huruf Braille Tidak berhasil
7	G	 LOAD IMAGE Output: Huruf G Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf I Tidak berhasil	 LOAD IMAGE Output: Bukan Huruf Braille Tidak berhasil
8	H	 LOAD IMAGE Output: Huruf I Berhasil, karena outputnya sudah benar	 LOAD IMAGE Output: Huruf J Tidak berhasil	 LOAD IMAGE Output: Bukan Huruf Braille Tidak berhasil

9	I	  Output: Huruf I Berhasil, karena outputnya sudah benar	  Output: Huruf I Berhasil dengan menggunakan background berwarna	  Output: Bukan Huruf Braille Tidak berhasil
10	J	  Output: Huruf J Berhasil, karena outputnya sudah benar	  Output: Huruf I Tidak berhasil	  Output: Bukan Huruf Braille Tidak berhasil

Tabel diatas merupakan pengujian *image braille* ke abjad terdapat 30 percobaan yang dimana pengujian gambarnya ada 3 macam yaitu gambar *braille* dengan jarak 1cm, gambar *braille background* berwarna dengan jarak 1cm dan pengujian gambar *braille* dengan jarak 5 cm. Untuk mengetahui persentase kesalahan dengan rumus sebagai berikut:

$$\text{Persentase kesalahan} = \frac{\text{Jumlah gambar yang benar}}{\text{Jumlah gambar yang dicoba}} \times 100\%$$

Berdasarkan dari data pengujian aplikasi dapat diperoleh persentase kesalahan aplikasi sebagai berikut:

$$\text{Persentase kesalahan} = \frac{15}{30} \times 100\% = 50\%$$

Maka tingkat keakurasian *image braille* ke abjad yaitu 50%

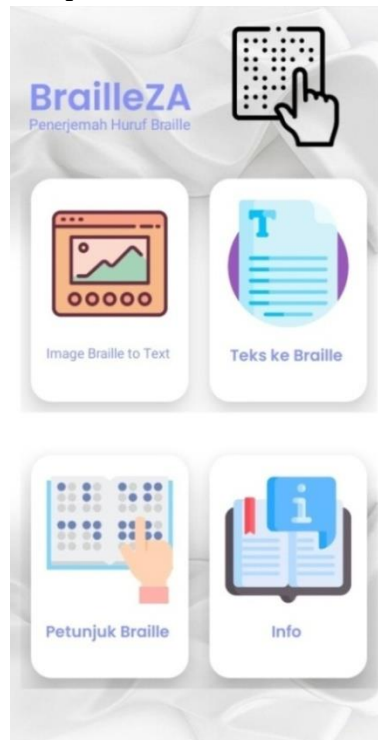
F. Detail aplikasi

1. Halaman *splashscreen* aplikasi



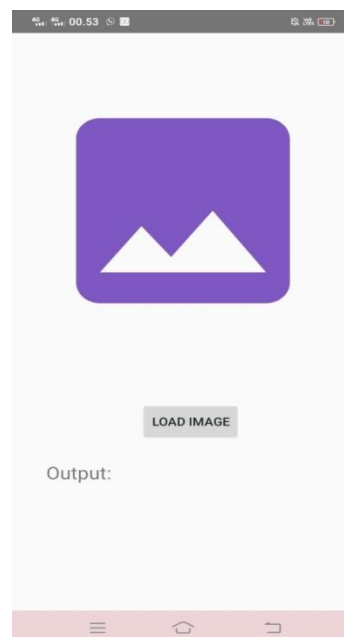
Gambar 4. 5 Halaman *splashscreen*

2. Halaman awal aplikasi



Gambar 4. 6 Halaman menu awal aplikasi

3. Halaman *image braille* ke teks

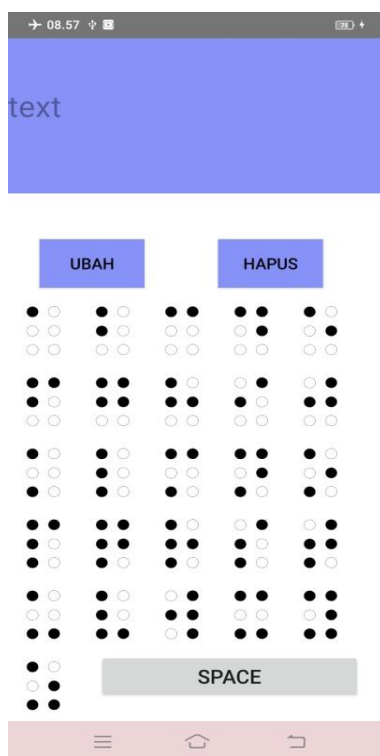


Gambar 4.7 Halaman *image braille* ke teks

4. Halaman penerjemah huruf *braille*



Gambar 4.8 Halaman penerjemah huruf *braille*

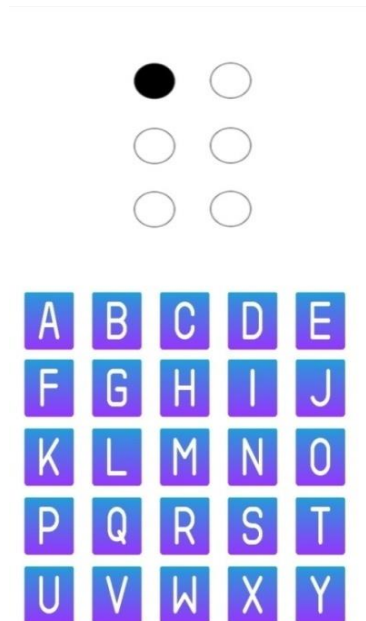


Gambar 4.9 Halaman penerjemah huruf *braille* ke teks



Gambar 4.10 Halaman penerjemah teks ke braille

5. Halaman petunjuk huruf *braille*



Gambar 4. 11 Halaman petunjuk huruf *braille*

6. Halaman informasi aplikasi



Gambar 4.11 Halaman informasi aplikasi

G. Implementasi

Implementasi sistem adalah suatu fase inovasi sebuah teknologi yang dirancang siap beroperasi. Bahasa pemrograman yang digunakan untuk membuat aplikasi ini adalah bahasa pemrograman java dan kotlin.

1. Kebutuhan perangkat keras (*hardware*)

Spesifikasi perangkat keras dapat dilihat dari tabel berikut :

Tabel 4. 4 Spesifikasi *hardware*

Spesifikasi <i>hardware</i>	
Laptop/PC	Toshiba Satelite l40-A
Processor laptop	Intel(R) CPU 1037U @1.80GHz (2 CPUs), ~1.8 GHz
RAM laptop	9 GB

2. Kebutuhan perangkat lunak(*Software*)

Spesifikasi dapat dilihat dari tabel sebagai berikut :

Tabel 4. 5 Spesifikasi *software*

Spesifikasi <i>software</i>	
Sistem operasi laptop	Windows 10
Tool <i>IDE</i>	Android Studio
Bahasa Pemrograman	Java & kotlin

H. Pengujian sistem

Pengujian aplikasi dapat dilakukan dengan cara 2 pengujian yaitu *Black Box* dan *White Box*. Pengujian sistem ini berfokus untuk mengetahui proses dan cara kerja aplikasi.

1. Black box

Berikut ini merupakan pengujian *black box* dari aplikasi penerjemah huruf *braille* Indonesia ke dalam huruf abjad.

a) Pengujian tampilan menu utama pada aplikasi

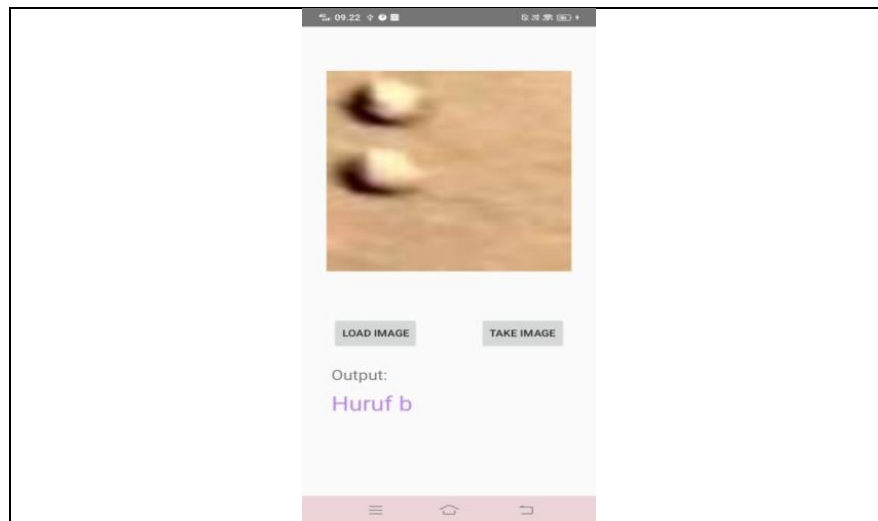
Tabel 4.6 Pengujian tampilan menu utama

Test faktor	Hasil	Kesimpulan
Menjalankan aplikasi pertama kali.	✓	Berhasil, karena dapat menampilkan halaman menu utama.
Screenshoot		
		

b) Pengujian deteksi huruf *braille* pada aplikasi

Tabel 4.7 Pengujian deteksi huruf *braille*

Test faktor	Hasil	Kesimpulan
Pengujian deteksi huruf <i>braille</i>	✓	Berhasil, karena dapat menampilkan <i>output</i> huruf abjad
Screenshoot		

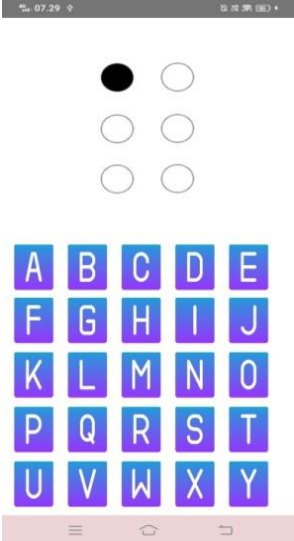


c) Pengujian huruf *braille* tidak terdeteksi pada aplikasi

Tabel 4.8 Pengujian huruf *braille* tidak terdeteksi

Test faktor	Hasil	Kesimpulan
Pengujian aplikasi ketika deteksi object bukan huruf <i>braille</i>	✓	Berhasil, karena dapat menampilkan <i>output</i> bukan huruf <i>braille</i> / huruf <i>braille</i> tidak terdeteksi
Screenshoot		

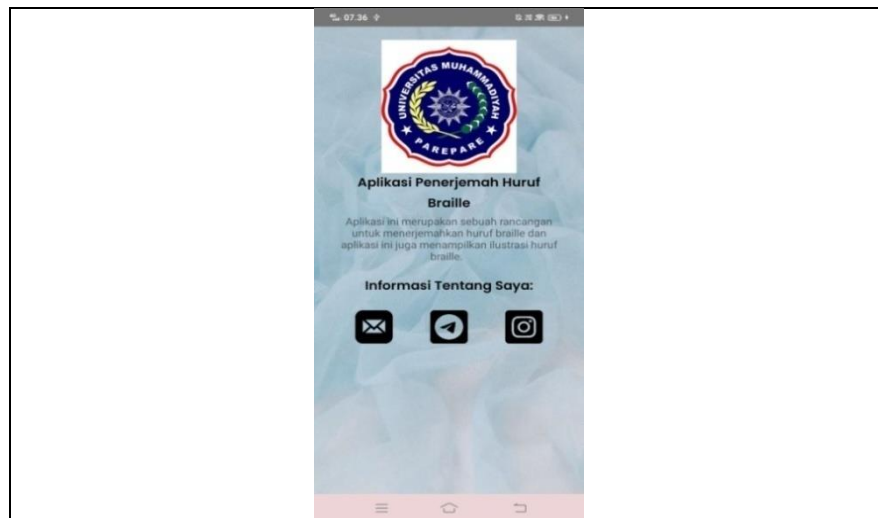
d) Pengujian animasi huruf *braille* pada aplikasi**Tabel 4.9** Pengujian animasi huruf *braille*

Test faktor	Hasil	Kesimpulan
Memilih menu petunjuk <i>braille</i>		Berhasil, karena dapat menampilkan huruf <i>braille</i> sesuai dengan huruf yang di tekan
Screenshoot		
		

e) Pengujian halaman informasi aplikasi

Tabel 4.10 Pengujian informasi aplikasi

Test faktor	Hasil	Kesimpulan
Memilih menu informasi pada aplikasi		Berhasil, karena dapat menampilkan tentang halaman informasi aplikasi
Screenshoot		

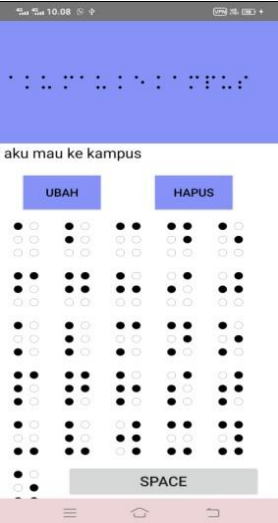


f) Pengujian huruf ke *braille*

Tabel 4.11 Pengujian huruf ke *braille*

Test faktor	Hasil	Kesimpulan
Memilih menu teks ke <i>braille</i>	✓	Berhasil, karena dapat menampilkan huruf <i>braille</i>
Screenshoot		

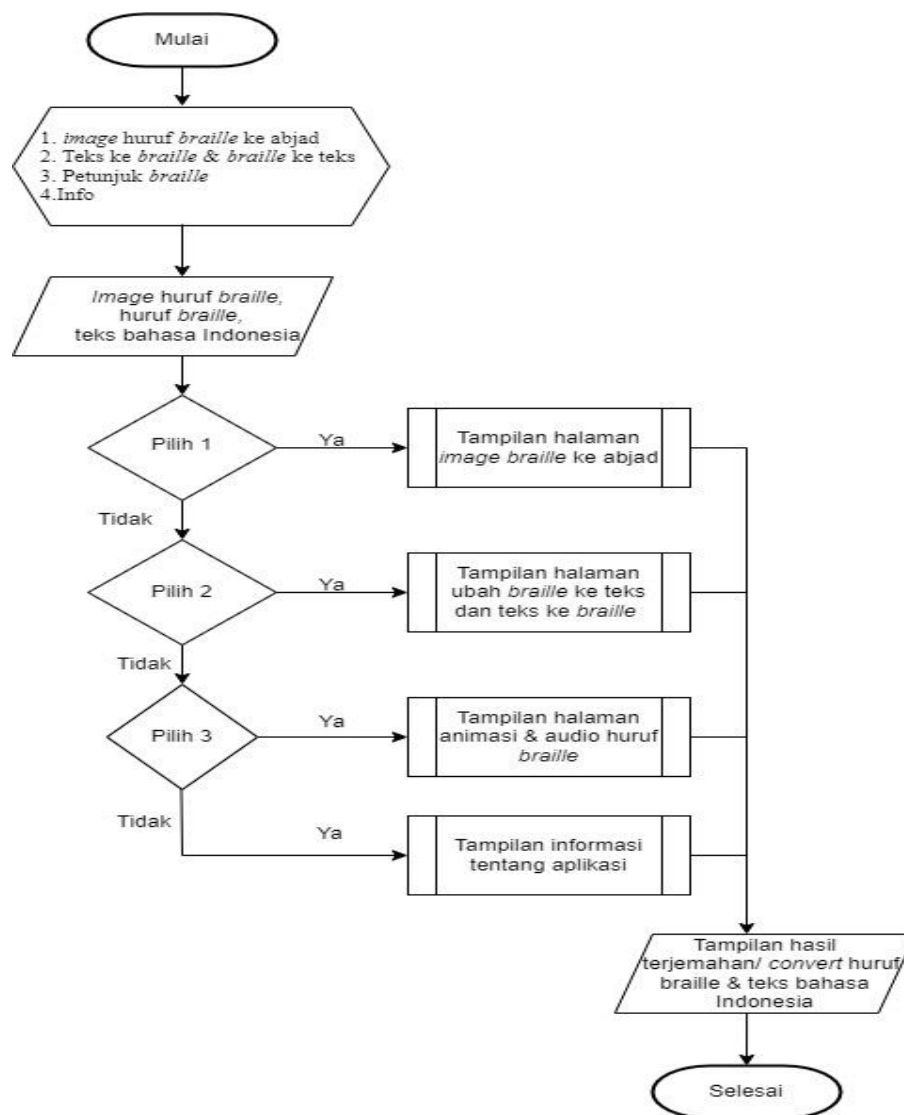
g) Pengujian *braille* ke huruf**Tabel 4.12** Pengujian *braille* ke huruf

Test faktor	Hasil	Kesimpulan
Memilih menu <i>braille</i> ke huruf		Berhasil, karena dapat mengubah huruf <i>braille</i> ke huruf abjad
Screenshoot		
		

2. White box

a. White box aplikasi form utama

1) Flowchart sistem

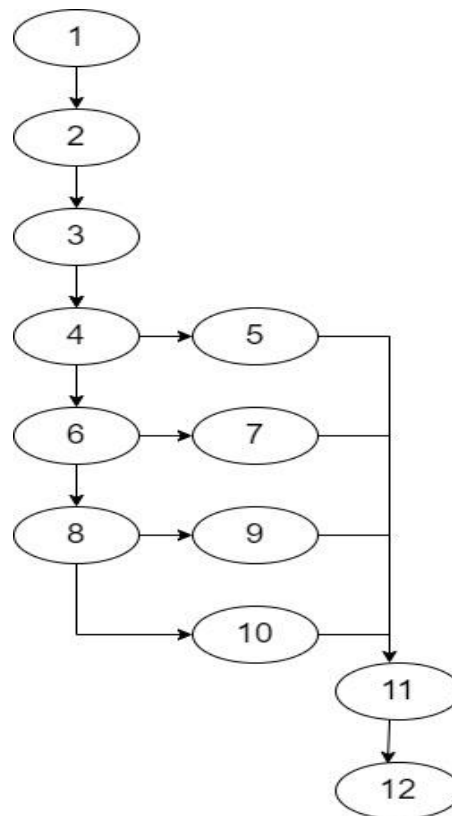


Gambar 4. 12 Flowchart sistem

Gambar flowchart di atas menjelaskan variabelnya yaitu *image braille* ke abjad, teks ke *braille* & *braille* ke teks, petunjuk *braille*, dan info setelah itu, inputannya berupa *image* huruf *braille*, huruf *braille*, dan teks bahasa Indonesia jika pilih menu 1 akan menampilkan tampilan halaman *image braille* ke abjad,

jika pilih menu 2 maka akan menampilkan halaman ubah huruf *braille* ke teks dan teks ke *braille*, jika pilih 3 maka akan menampilkan halaman animasi & audio huruf *braille*, jika tidak pilih 3 maka akan menampilkan halaman informasi tentang aplikasi.

2) Flowgraph sistem



Gambar 4. 13 Flowgraph sistem

Dari *flowgraph* sistem di atas maka dapat dilakukan proses perhitungan sebagai berikut:

1. Menghitung *cyclomatic complexity* $V(G)$ asal *Edge* dan *Node*:

Menggunakan rumus berikut : $V(G) = E - N + 2$

E (*edge*) = 14

N (*Node*) = 12

$$P(\text{Predikat } Node) = 3$$

Penyelesaian : $V(G) = E - N + 2$
 $= 14 - 12 + 2 = 4$

$$\begin{aligned} \text{Predikat (P)} &= P + 1 \\ &= 3 + 1 = 4 \end{aligned}$$

2. Memiliki *region* = 4 menurut perhitungan kompleksitas siklomatik dari grafik aliran di atas.
3. Pada diagram alur di atas, jalur independennya adalah:

Path 1 = 1 – 2 – 3 – 4 – 10 – 11 – 12

$$Path\ 2 = 1 - 2 - 3 - 5 - 6 - 10 - 11 - 12$$

$$Path\ 3 = 1 - 2 - 3 - 5 - 7 - 8 - 10 - 11 - 12$$

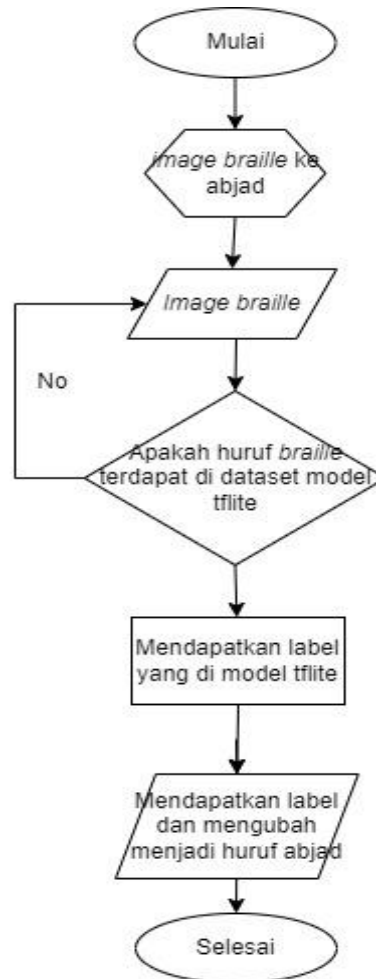
$$Path\ 4 = 1 - 2 - 3 - 5 - 7 - 9 - 10 - 11 - 12$$

Tabel 4. 13 Grafik matriks sistem

[illegible]

b. *White box image* huruf braille ke abjad

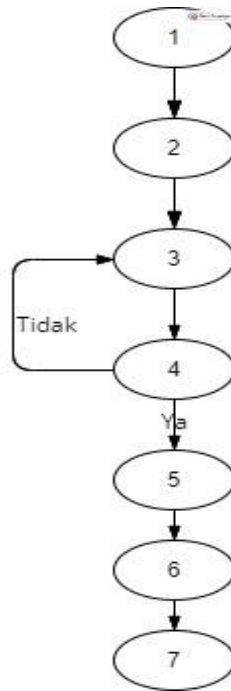
1) *Flowchart image* huruf braille ke abjad



Gambar 4. 14 *Flowchart image* huruf braille ke abjad

Gambar *flowchart* di atas menjelaskan tentang *image braille* ke abjad dengan menginput *image braille* dan apakah huruf *braille* terdapat di dataset model tflite jika iya maka akan mendapatkan label yang di model tflite dan outputnya berubah huruf abjad.

2) *Flowgraph image* huruf *braille* ke abjad



Gambar 4. 15 *Flowgraph image* huruf *braille* ke abjad

Dari *flowgraph image* huruf *braille* ke abjad di atas maka dapat dilakukan proses perhitungan sebagai berikut:

1. Menghitung *cyclomatic complexcity* $V(G)$ asal *Egde* dan *Node*:

Menggunakan rumus berikut : $V(G) = E - N + 2$

E (*edge*) = 7

N (*Node*) = 7

P (*Predikat Node*) = 1

Penyelesaian : $V(G) = E - N + 2$

$$= 7 - 7 + 2 = 2$$

Predikat (P) = $P + 1$

$$= 1 + 1 = 2$$

2. Memiliki *region* = 2 menurut perhitungan *cyclomatic complexity* dari grafik aliran di atas.

3. Pada diagram alur di atas, jalur independennya adalah:

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7

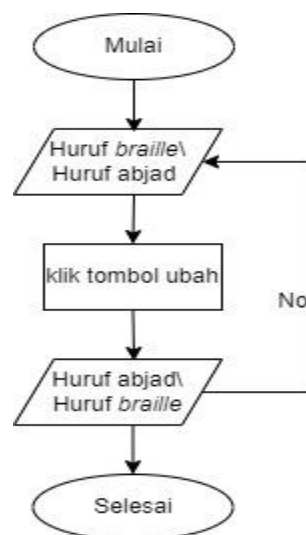
Path 2 = 1 – 2 – 3 – 4 – 3 – 4 – 5 – 6 – 7

Tabel 4. 14 Grafik matriks *image braille* ke abjad

	1	2	3	4	5	6	7	E-1
1		1						1-1=0
2			1					1-1=0
3				1				1-1=0
4			1		1			2-1=1
5						1		1-1=0
6							1	1-1=0
7								0
	SUM (E+1)							1+1=2

c. *White box* teks ke *braille* & *braille* ke abjad

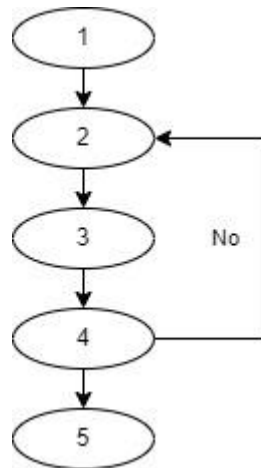
1) *Flowchart* teks ke *braille* & *braille* ke abjad



Gambar 4. 16 *Flowchart* teks ke *braille* & *braille* ke teks

Flowchart di atas menjelaskan tentang huruf *braille* ke teks dan sebaliknya kemudian inputannya berupa huruf *braille* & huruf abjad setelah klik tombol ubah dan outputnya berupa huruf abjad/ huruf *braille*.

2) *Flowgraph* teks ke *braille* & *braille* ke abjad



Gambar 4. 17 *Flowgraph* teks ke *braille* & *braille* ke teks

Dari *flowgraph* teks ke *braille* & *braille* ke teks di atas maka dapat dilakukan proses perhitungan sebagai berikut:

1. Menghitung *cyclomatic complexcity* $V(G)$ asal *Egde* dan *Node*:

Menggunakan rumus berikut : $V(G) = E - N + 2$

E (*edge*) = 5

N (*Node*) = 5

P (*Predikat Node*) = 1

Penyelesaian : $V(G) = E - N + 2$

$$= 5 - 5 + 2 = 2$$

Predikat (P) = $P + 1$

$$= 1 + 1 = 2$$

2. Memiliki *region* = 2 menurut perhitungan *cyclomatic complexity* dari grafik aliran di atas.

3. Pada diagram alur di atas, jalur independennya adalah:

Path 1 = 1 – 2 – 3 – 4 – 5

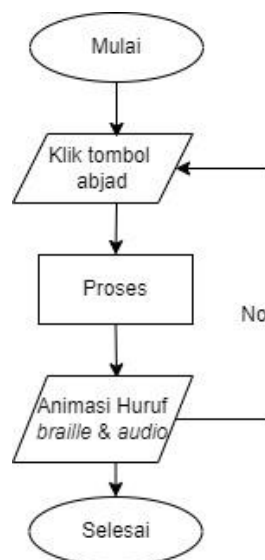
Path 2 = 1 – 2 – 4 – 5

Tabel 4. 16 Grafik matriks *image braille* ke abjad

	1	2	3	4	5	E-1
1		1				1-1=0
2			1			1-1=0
3				1		1-1=0
4		1			1	2-1=1
5						0
	SUM (E + 1)					1+1 = 2

d. *White box* petunjuk *braille*

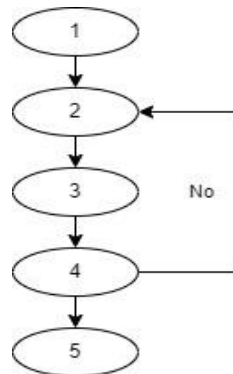
1) *Flowchart* petunjuk *braille*



Gambar 4. 18 *Flowchart* petunjuk *braille*

Flowchart diatas menjelaskan tentang petunjuk *braille* klik tombol abjad maka akan menampilkan output berupa animasi huruf *braille*.

2) *Flowgraph* petunjuk *braille*



Gambar 4. 19 *Flowgraph* petunjuk *braille*

Dari *flowgraph* petunjuk *braille* di atas maka dapat dilakukan proses perhitungan sebagai berikut:

1. Menghitung *cyclomatic complexcity* $V(G)$ asal *Egde* dan *Node*:

Menggunakan rumus berikut : $V(G) = E - N + 2$

E (*edge*) = 5

N (*Node*) = 5

P (*Predikat Node*) = 1

Penyelesaian : $V(G) = E - N + 2$

$$= 5 - 5 + 2 = 2$$

P (*Predikat* (P)) = $P + 1$

$$= 1 + 1 = 2$$

2. Memiliki *region* = 2 menurut perhitungan *cyclomatic complexcity* dari grafik aliran di atas.
3. Pada diagram alur di atas, jalur independennya adalah:

Path 1 = 1 – 2 – 3 – 4 – 5

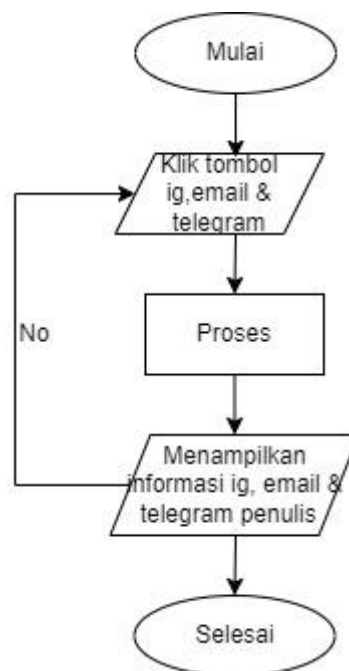
Path 2 = 1 – 2 – 4 – 5

Tabel 4. 17 Grafik petunjuk *braille*

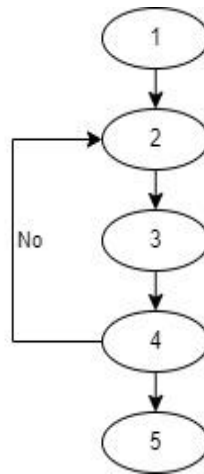
	1	2	3	4	5	E-1
1		1				$1-1=0$
2			1			$1-1=0$
3				1		$1-1=0$
4		1			1	$2-1=1$
5						0
SUM (E + 1)						$1+1 = 2$

e. *White box* info

1) *Flowchart* info

**Gambar 4. 20** *Flowchart* info

Flowchart diatas menjelaskan tentang informasi tentang aplikasi, klik tombol ig, email & telegram kemudian akan menampilkan informasi ig, email & telegram penulis.

2) *Flowgraph* info**Gambar 4. 21** *Flowgraph* info

Dari *flowgraph* info di atas maka dapat dilakukan proses perhitungan sebagai berikut:

1. Menghitung *cyclomatic complexity* $V(G)$ asal *Edge* dan *Node*:

Menggunakan rumus berikut : $V(G) = E - N + 2$

E (*edge*) = 5

N (*Node*) = 5

P (*Predikat Node*) = 1

Penyelesaian : $V(G) = E - N + 2$

$$= 5 - 5 + 2 = 2$$

Predikat (P) = $P + 1$

$$= 1 + 1 = 2$$

2. Memiliki *region* = 2 menurut perhitungan *cyclomatic complexity* dari grafik aliran di atas.
3. Pada diagram alur di atas, jalur independennya adalah:

$Path\ 1 = 1 - 2 - 3 - 4 - 5$

$Path\ 2 = 1 - 2 - 4 - 5$

Tabel 4. 18 Grafik matriks info

	1	2	3	4	5	E-1
1		1				1-1=0
2			1			1-1=0
3				1		1-1=0
4		1			1	2-1=1
5						0
SUM (E + 1)						1+1 = 2

Tabel 4. 19 Hasil pengujian

No	Flowchart	Independent path	Region	Cyclomatic Complexity
1	Flowchart sistem	4	4	4
2	Flowchart <i>image</i> huruf braille ke abjad	2	2	2
3	Flowchart teks ke <i>braille</i> & <i>braille</i> ke teks	2	2	2
4	Petunjuk <i>braille</i>	2	2	2
5	Info	2	2	2

BAB V

PENUTUP

A. Kesimpulan

Dari hasil penelitian yang telah dilakukan oleh penulis maka dapat disimpulkan yaitu aplikasi penerjemah huruf *braille* ini merupakan aplikasi berbasis *android* yang di jalankan di *smartphone* android untuk membantu *user* mempelajari dan mengetahui huruf *braille*, program ini juga menyediakan animasi huruf *braille* yang sesuai dengan *braille* bahasa Indonesia. Pada penerjemahan *braille* ke teks bahasa Indonesia dan teks bahasa Indonesia ke huruf *braille* menggunakan *class hashmap* di java. Sedangkan pada penerjemahan *image braille* ke huruf abjad menggunakan metode klasifikasi *Convolutional Neural Network* dengan menggunakan arsitektur model MobileNetV2 yang dimana terdiri dari 27 *class* dengan jumlah 8664 *image*. Hasil klasifikasi tersebut akan di *export* menjadi model selanjutnya yang akan diolah menggunakan *tensorflow lite* android studio. Berdasarkan hasil dari pengujian model, tingkat akurasi yang dihasilkan mencapai 0.9183. Sehingga performa dari deteksi aplikasi tersebut belum optimal.

B. Saran

Setelah melakukan penelitian, maka penulis menyadari ada beberapa yang perlu di perbaiki dan pengembangan maka dari itu penulis memberikan saran yaitu penulis berharap aplikasi ini dapat dikembangkan dengan menerjemahkan *image braille* ke dalam teks ataupun audio dalam bahasa Indonesia dan dapat

digunakan secara *real time* pada penerjemah *image* huruf *braille*. Penulis berharap jumlah datasetnya diperbanyak agar nilai akurasinya mencapai 0,99.

DAFTAR PUSTAKA

- Anwar, Y. (2021). Aplikasi Penerjemah Bahasa Isyarat Indonesia ke dalam Huruf Abjad. Skripsi tidak dipublikasikan. Parepare: Universitas Muhammadiyah Parepare
- Ayu, S. (2022). Klasifikasi dan pencarian jurnal menggunakan algoritma KNN. Skripsi tidak dipublikasikan. Parepare: Universitas Muhammadiyah Parepare
- Atmaja, R. D. L. Novamizanti, J. Halomoan, and M. A. Murti. (2017). *An Image Processing Algorithm To Find Edges In Grayscale Image*. ISSN: 0973-7006
- Andono, P. N., Sutojo, T. and Muljono. (2017). *Dalam Pengolahan Citra Digital*. Diedit oleh Pramesta, A. Yogyakarta: Andi
- Azmi, F., dkk. (2019). Implementasi metode CNN pada pengenalan objek video CCTV . Jurnal Mantik Penusa. 3(1.1): 74-79
- Hansun, S., Kristanda, M. B. & Saputra, M. W., (2018). Pemrograman Android Dengan Android Studio IDE. Yogyakarta: Andi.
- Herlambang, F. M., Hermana, N. A., & Putra, R. K. (2020). Pengenalan Karakter Huruf Braille dengan metode Convolutional Neural Network. *Systematic*. 6(2): 20-26.
- Haviluddin. (2011). *Memahami Penggunaan UML (Unified Modelling Language)*. Jurnal Informatika Mulawarman, 1-15
- Muliawati., Wijanto, A. H., & Magdalena, R. (2011). *Konverter Otomatis Karakter Braille ke Teks Berbasis Image dan SVM*
- Munir. R. (2004). *Pengolahan Citra Digital dengan Pendekatan Logaritmik*. Penerbit: Informatika, Bandung
- Nurwahida. (2016). Aplikasi Pendeteksi Keaslian Dokumen Menggunakan Metode K-Nearest Neighbour. Skripsi tidak dipublikasikan. Parepare: Universitas Muhammadiyah Parepare
- Putra, D. (2010). *Pengolahan Citra Digital*. Penerbit: Andi Offset
- Rahman, A. (2014). *Simbol simbol pada Flowchart dan penjelasannya*
- Ramiati., Aulia, S., & Lifwarda. (2020). *Aplikasi Identifikasi Huruf Braille Menggunakan Computer Vision Berbasis Raspberry Pi*. ISSN: 2302-2949

- Sulviyani, I., Novamizanti, L. and Atmaja, D. R.(2017). *Rancangan Konversi Braille Ke Teks Berbasis Android*. Prosiding Seminar Nasional XII
- Romando, E., Sudaryanto, A.(2018). *Sistem Pengenalan Pola Huruf Braille Berbasis Audio Menggunakan Metode Naïve Bayes*. Jurnal Ilmu Komputer dan Desain Komunikasi Visual
- Subur, J (2015). Pengenalan Huruf Braille menggunakan Metode Blob Analysis dan Artificial Neural Network. Tesis tidak dipublikasikan. Surabaya: Institut Teknologi Sepuluh Nopember
- Sumiati, M., Abdillah, R., Cahyo, A. (2021). Pemodelan UML untuk Sistem Informasi Persewaan Alat Pesta. *Jurnal Fasilkom*. 11(2): 79-86
- Winarno, E., & Zaki, A. (2011). *Membuat Sendiri Aplikasi Android untuk Pemula*. Penerbit : PT Elex Media Komputindo
- Wibowo, A. (2017). Simplikasi *list* android dengan penggunaan *hashmap*. *Jurnal matrik*. Vol 16(2)
- Widodo, B. (2016). Machine Learning Dan Computational Intelligence
- Zufar, M., & Setiyono, B. (2016). Convolutional neural networks untuk pengenalan wajah secara real-time. *Jurnal Sains Dan Seni Its* Vol. 5 No. 2 (2016) 2337-3520 (2301-928X Print), A72-A77