BAB II
LANDASAN TEORI

2.1 Anggrek Bulan

Bunga anggrek bulan (Phalaeonopsis amabilis) berwarna putih bersih dengan sedikit variasi kuning dan bintik kemerahan di bibir bunga. Bunga anggrek bulan tersusun majemuk dan muncul dari ketiak daun. Bunga simetri bilateral, helaian kelompok umumnya berwarna mirip dengan mahkota bunga. Satu helai mahkota bunga termodifikasi membentuk semacam lidah yang melindungi suatu struktur aksesoris yang membawa benang sari dan putik. Benang sari mempunyai tangkai sangat pendek dengan kepala sari berbentuk cakram kecil dan terlindungi oleh struktur kecil yang harus dibuka oleh serangga penyerbuk dan membawa serbuk sari ke putik. Anggrek bulan memiliki 2 macam akar yaitu akar lekat dan akar udara, akar lekat berfungsi untuk melekat, menahan keseluruhan tanaman agar tetap berada di tempatnya dan dapat menyerap air serta nutrisi, akar udara berfungsi untuk menyerap nutrisi dalam bentuk uap air dan gas[1].

Batang dari anggrek bulan sangat pendek dan terbungkus oleh seludang daun dengan bagian ujung batang tumbuh lurus tidak terbatas (monopodial). Daun anggrek mempunyai tulang daun yang sejajar dengan helaian daun, berwarna hijau, daun tebal, dan berdaging. Daun anggrek berbentuk lonjong dengan bagian ujung agak melebar dan tumpul. Buah anggrek merupakan buah capsular yang berbelah 6 dengan jumlah biji yang banyak. Biji-biji anggrek ini tidak memiliki endosperm yaitu jaringan penyimpan cadangan makanan seperti biji tanaman pada umumnya[2].

2.2 *Unified Modelling Language*

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi obyek (Munawar, 2018). Hal ini dikarenakan UML menyediakan Bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru (*blue* print) atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan

rancangan mereka dengan yang lain[3]. UML menjelaskan jenis diagram untuk analisis, desain, dan pemrograman berorientasi obyek, sehingga menjamin transisi tanpa batas dari persyaratan yang ditempatkan pada sistem hingga implementasi akhir. Struktur dan perilaku sistem juga ditunjukkan, sehingga menawarkan referensi yang jelas untuk optimasi solusi.

Manfaat penggunaan UML sebagai "bahasa yang umum" mengarah pada peningkatan kerjasama antara kompetensi teknis dan non-teknis seperti pemimpin proyek, analis bisnis, arsitek perangkat lunak / perangkat keras, perancang, dan pengembang. Ini membantu dalam pemahaman sistem yang lebih baik, dalam mengungkapkan pilihan penyederhanaan dan / atau pemulihan, dan dalam pengenalan lebih mudah atas risiko yang mungkin terjadi. Dengan deteksi dini kesalahan dalam tahap analisis dan perancangan proyek, biaya dapat dikurangi selama tahap implementasi, serta untuk menghemat banyak waktu. Terdapat 4 (empat) macam hubungan antar obyek di dalam penggunaan UML, yaitu;

1. *Dependency*, adalah hubungan semantik antara dua benda/things yang mana sebuah benda berubah mengakibatkan benda satunya akan berubah pula. Umumnya sebuah dependency digambarkan sebuah panah dengan garis terputus-putus.
2. *Association*, hubungan antar benda struktural yang terhubung diantara obyek. Kesatuan obyek yang terhubung merupakan hubungan khusus, yang menggambarkan sebuah hubungan struktural diantara seluruh atau sebagian. Umumnya association digambarkan dengan sebuah garis yang dilengkapi dengan sebuah label, nama, dan status hubungannya.
3. *Generalizations*, adalah menggambarkan hubungan khusus dalam obyek anak/child yang menggantikan obyek parent/induk. Dalam hal ini, obyek anak memberikan pengaruhnya dalam hal struktur dan tingkah lakunya kepada obyek induk. Digambarkan dengan garis panah.
4. *Realizations*, merupakan hubungan semantik antara pengelompokkan yang menjamin adanya ikatan diantaranya. Hubungan ini dapat diwujudkan diantara interface dan kelas atau elements, serta antara *use case*s dan collaborations. Model dari sebuah hubungan realization.

Di dalam UML terdapat beberapa jenis diagram, antara lain yaitu:

(a) *Class Diagram*, diagram ini terdiri dari class, interface, association, dan collaboration. Diagram ini menggambarkan objek - objek yang ada di sistem.

(b) *Object Diagram*, diagram ini menggambarkan hasil instansi dari class diagram. Diagram ini digunakan untuk membuat prototype

(c) *Component Diagram*, diagram ini menggambarkan kumpulan komponen dan hubungan antar komponen. Komponen terdiri dari class, interface, atau collaboration

(d) *Deployment Diagram*, diagram ini menggambarkan kumpulan node dan hubungan antar node. Node adalah entitas fisik dimana komponen di-deploy. Entitas fisik ini dapat berupa server atau perangkat keras lainnya.

(e) *Use Case Diagram*, diagram ini menggambarkan kumpulan use case, aktor, dan hubungan mereka. Use case adalah hubungan antara fungsionalitas sistem dengan aktor internal/eksternal dari sistem.

(f) *Sequence Diagram*, diagram ini menggambarkan interaksi yang menjelaskan bagaimana pesan mengalir dari objek ke objek lainnya.

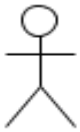
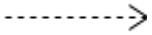
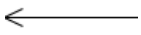
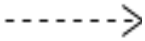
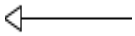
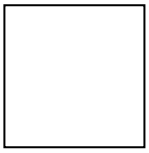
(g) *Collaboration Diagram*, diagram ini merupakan bentuk lain dari sequence diagram. Diagram ini menggambarkan struktur organisasi dari sistem dengan pesan yang diterima dan dikirim.

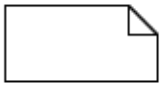
(h) *Statechart Diagram*, diagram ini menggambarkan bagaimana sistem dapat bereaksi terhadap suatu kejadian dari dalam atau luar. Kejadian (event) ini bertanggung jawab terhadap perubahan keadaan sistem.

(i) *Activity Diagram*, menggambarkan aliran kontrol sistem. Diagram ini digunakan untuk melihat bagaimana sistem bekerja ketika dieksekusi.

Berikut beberapa contoh simbol dalam UML:

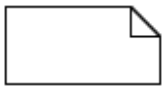
Tabel 1 Simbol Simbol Use Case Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

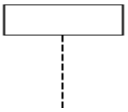
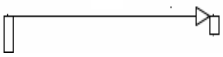
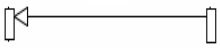
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Tabel 2 Daftar Simbol Activity Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		State	Nilai atribut dan nilai link pada suatu waktu tertentu, yang dimiliki oleh suatu objek.
2		Initial Pseudo State	Bagaimana objek dibentuk atau diawali
3		Final State	Bagaimana objek dibentuk dan dihancurkan
4		Transition	Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya
5		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.

6		Node	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

Tabel 3 Daftar Simbol Sequence Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		Actor	Menggambarkan orang yang sedang berinteraksi dengan sistem
2		LifeLine	Objek entity, antarmuka yang saling berinteraksi.
3		Message	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi
4		Message	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi

2.3 Penelitian Terkait

Berikut adalah daftar acuan penelitian dalam penelitian kali ini:

Tabel 4 Tabel Daftar Acuan

No	Nama	Judul
1	M. Rohman	Anggrek Bulan (Phalaenopsis amabilis) di Pt Anugrah Anggrek Nusantara[4]

2	D. P. Pamungkas	Ekstraksi Citra menggunakan Metode GLCM dan KNN untuk Identifikasi Jenis Anggrek (Orchidaceae)[5]
3	L. H. Mukminin, M. P. Al Asna, and F. . Setiowati	Pengaruh Pemberian Giberelin Dan Air Kelapa Terhadap Perkecambahan Biji Anggrek Bulan (Phalaenopsis sp.)[6]
4	B. Mulyana	Identifikasi Jenis Bunga Anggrek Menggunakan Pengolahan Citra Digital Dengan Metode KNN[7]
5	M. A. . Putra, I. M. A. S. Wijaya, and Y. Setiyo	Pengembangan Algoritma Image Processing Untuk Menduga Hasil Panen Padi[8]

2.4 Usecase Diagram

Use case Diagram merupakan suatu diagram yang mendeskripsikan hubungan antara aktor dengan sistem. Use case diagram dapat mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Use case diagram juga bisa digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan bisa mempresentasikan komunikasi antara aktor dengan sistem yang ada. Dengan demikian use case dapat dipresentasikan dengan urutan yang sederhana dan akan mudah dipahami oleh para konsumen. Manfaat use case sendiri adalah untuk memudahkan komunikasi dengan menggunakan domain expert dan juga end user, memberikan kepastian pemahaman yang pas tentang requirement atau juga kebutuhan sebuah sistem.

Use case diagram memiliki 3 komponen :

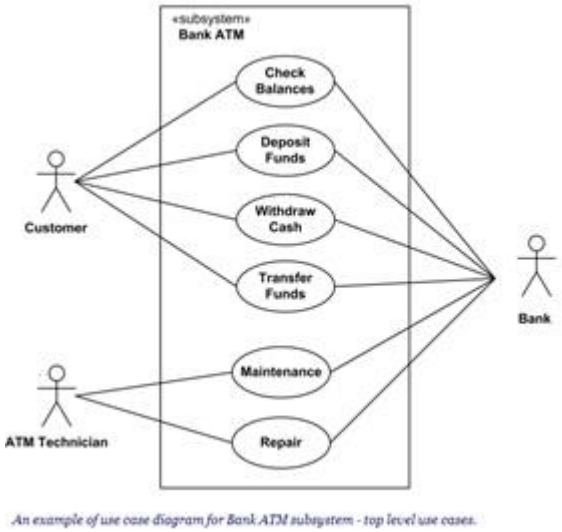
1. **Sistem**

   Sistem menyatakan batasan sistem dalam relasi dengan aktor-aktor yang menggunakannya (diluar sistem) dan fitur-fitur yang harus disediakan(dalam sistem).
2. **Aktor**

Aktor adalah segala hal diluar sistem yang akan menggunakan sistem tersebut untuk melakukan sesuatu. Bisa merupakan manusia, sistem, atau device yang memiliki peranan dalam keberhasilan operasi dari sistem.

3. **Use Case**

   Gambaran fungsional dari sebuah sistem, dengan demikian antara konsumen dan pengguna akan paham mengenai fungsi suatu sistem yang sedang di bangun.



Gambar 3 Use Case Diagram

2.5 Activity Diagram

Activity diagram merupakan aliran aktivitas atau aliran kerja dalam sebuah sistem yang akan dijalankan. Activity Diagram juga digunakan untuk mendefinisikan atau mengelompokan alur tampilan dari sistem tersebut. Activity Diagaram memiliki komponen dengan bentuk tertentu yang dihubungkan dengan tanda panah. Panah tersebut mengarah ke-urutan aktivitas yang terjadi dari awal hingga akhir

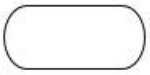
- Initial State

Initial State adalah awal dimulainya suatu aliran kerja pada activity diagram dan pada sebuah activity diagram hanya terdapat satu initial state.

- Final State

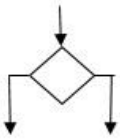
Final State adalah bagian akhir dari suatu aliran kerja pada sebuah activity diagram dan pada sebuah activity diagram bisa terdapat lebih dari satu final state.

- Activity



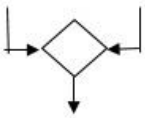
Activitas adalah aktivitas atau pekerjaan yang dilakukan dalam aliran kerja.

- Decision



Decision berfungsi untuk menggambarkan pilihan kondisi dimana ada kemungkinan perbedaan transisi. untuk memastikan bahwa aliaran kerja dapat mengalir ke lebih dari satu jalur.

- Merge

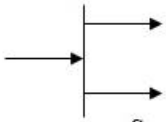


Merge berfungsi untuk menggabungkann kembali aliran kerja yang sebelumnya telah dipecah oleh Decission.

- Transition / Association

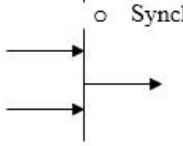
Transition untuk menghubungkan aktivitas selanjutnya setelah aktivitas sebelumnya

- Synchronization
  - Synchronization Fork



Synchronization Fork digunakan untuk memecah behavior menjadi aktivitas yang pararel (Contoh : User dapat memilih menu yang dapat dilakukan secara pararel).

  - Synchronization Join



Synchronization Join digunakan untuk menggabungkan kembali aktivitas yang pararel.

2.6 Class Diagram

Class diagram atau diagram kelas adalah salah satu jenis diagram struktur pada UML yang menggambarkan dengan jelas struktur serta deskripsi class, atribut, metode dan hubungan dari setiap objek. Class diagram bersifat statis, dalam artian diagram kelas bukan menjelaskan apa yang terjadi jika kelas-kelasnya berhubungan, melainkan menjelaskan hubungan apa yang terjadi.

2.7 Sequence Diagram

Sequence Diagram adalah diagram yang digunakan untuk menjelaskan dan menampilkan interaksi antar objek dalam suatu sistem secara detail. Selain itu, sequence diagram akan menampilkan pesan atau perintah yang telah dikirimkan, serta waktu eksekusi. Objek yang terkait dengan proses yang sedang berjalan biasanya diurutkan dari kiri ke kanan.

Berikut beberapa komponen utama yang sering digunakan:

1. Aktor.

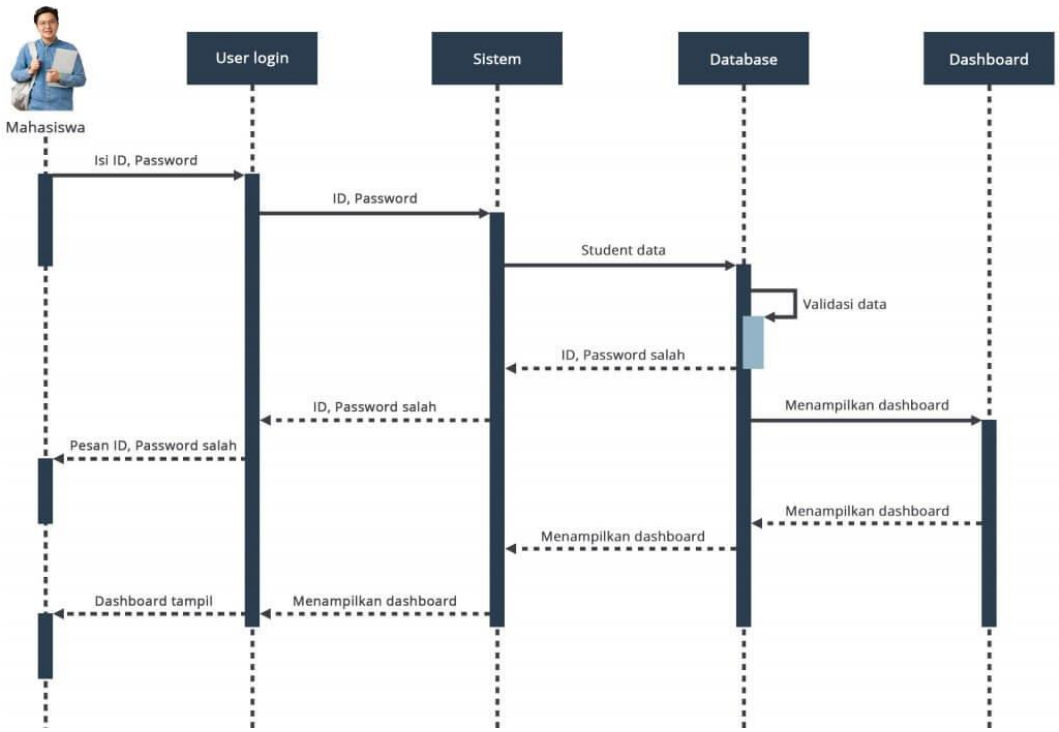
   Komponen pertama adalah aktor. Komponen ini menggambarkan pengguna di luar dan berinteraksi dengan sistem. Dalam diagram urutan, aktor biasanya diwakili oleh figur tongkat.

2. Activation box

   Activation box ini mewakili waktu yang dibutuhkan objek untuk menyelesaikan tugasnya. Semakin lama waktu, semakin lama kotak aktivasi secara otomatis. Komponen ini diwakili oleh persegi panjang.

3. Lifeline

   Komponen ini digambarkan dengan bentuk garis putus-putus. Lifeline ini biasanya memiliki kotak yang berisi objek yang memiliki fungsi untuk menggambarkan aktifitas dari objek.



Gambar 4 Sequence Diagram

2.8 Image Processing

Image processing adalah suatu bentuk pengolahan atau pemrosesan sinyal dengan input berupa gambar (image) dan ditransformasikan menjadi gambar lain sebagai keluarannya dengan teknik tertentu. Image processing dilakukan untuk memperbaiki kesalahan data sinyal gambar yang terjadi akibat transmisi dan selama akuisisi sinyal, serta untuk meningkatkan kualitas penampakan gambar agar lebih mudah diinterpretasi oleh sistem penglihatan manusia baik dengan melakukan manipulasi dan juga penganalisisan terhadap gambar.[9]

Operasi image processing dapat dikelompokkan berdasarkan dari tujuan transformasinya, yaitu sebagai berikut :

1. Image Enhancement (peningkatan kualitas gambar)
2. Image Restoration (pemulihan gambar)
3. Image Compression (kompresi gambar)
4. Image Refresention & Modelling (representasi dan permodelan gambar)

2.9 Algoritma Convolutional Neural Network

Convolutional Neural Network adalah salah satu metode machine learning dari pengembangan Multi Layer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis Deep Neural Network karena dalamnya tingkat jaringan dan banyak diimplementasikan dalam data citra. CNN memiliki dua metode; yakni klasifikasi menggunakan feedforward dan tahap pembelajaran menggunakan backpropagation. Cara kerja CNN setiap neuron dipresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berukuran satu dimensi.[1]

2.10 Android

Android adalah sistem operasi (OS) yang umum digunakan pada perangkat mobile seperti HP dan tablet. Sejarah android sendiri dimulai pada tahun 2007. Saat itu, OS ini secara resmi dikembangkan oleh Open Handset Alliance, yaitu konsorsium (asosiasi) yang terdiri dari 84 perusahaan. Beberapa di antaranya adalah perusahaan multi-nasional ternama seperti Google, Intel, Sony, dan Samsung.

Sepanjang sejarahnya, Android pun sudah beberapa kali melakukan upgrade. Dan setiap upgrade memiliki penamaan versi yang berbeda-beda.

Berikut adalah daftar versi Android beserta penamaan :[10]

Android 1.0	
Android 1.1	
Android 1.5	Cupcake
Android 1.6	Donut
Android 2.0 – 2.1	Eclair
Android 2.2 – 2.2.3	Froyo
Android 2.3 – 2.3.7	Gingerbread
Android 3.0 – 3.2.6	Honeycomb
Android 4.0 – 4.0.4	Ice Cream Sandwich
Android 4.1 – 4.3.1	Jelly Bean
Android 4.4 – 4.4.4	KitKat

Android 5.0 – 5.1.1	Lollipop
Android 6.0 – 6.0.1	Marshmallow
Android 7.0 – 7.1.2	Nougat
Android 8.0 – 8.1	Oreo
Android 9	Pie
Android 10	Android 10
Android 11	Android 11

2.11 Tensorflow

Tensorflow ialah Python library open source untuk komputasi numerik yang dapat mempercepat dan memudahkan dalam menggunakan machine learning. Tensorflow diciptakan oleh tim Google Brain, di mana framework ini ialah library open source yang digunakan untuk komputasi numerik dan project machine learning berskala besar. Tensorflow menggabungkan banyak model dan algoritma machine learning termasuk deep learning (neural network). Framework ini di susun menggunakan Python front-end API untuk membuat suatu aplikasi penggunaannya, dan menggunakan C++ yang memiliki kinerja terbaik dalam hal mengeksekusi.[11]

Tensorflow dapat melatih dan menjalankan neural network untuk keperluan mengklasifikasikan tulisan tangan, pengenalan gambar/object, serta menggabungkan suatu kata. Selanjutnya adalah re-current neural network, yang merupakan model sequential, dapat digunakan untuk Natural Language Processing (NLP), PDE (Partial Differential Equation) berdasarkan simulasi. Dan yang palng utama adalah bahwa Tensorflow dapat digunakan pada skala yang besar untuk produksi dengan menggunakan model yang sama pada ketika proses training data.

2.12 Machine Learning

Machine learning (ML) adalah mesin yang dikembangkan untuk bisa belajar dengan sendirinya tanpa arahan dari penggunanya. Pembelajaran mesin dikembangkan berdasarkan disiplin ilmu lainnya seperti statistika, matematika dan

data mining sehingga mesin dapat belajar dengan menganalisa data tanpa perlu di program ulang atau diperintah. Cara kerja machine learning sebenarnya berbeda-beda sesuai dengan teknik atau metode pembelajaran seperti apa yang kamu gunakan pada ML. Namun pada dasarnya prinsip cara kerja pembelajaran mesin masih sama, meliputi pengumpulan data, eksplorasi data, pemilihan model atau teknik, memberikan pelatihan terhadap model yang dipilih dan mengevaluasi hasil dari ML. Untuk memahami cara kerja dari ML, mari kita ulas cara kerja dari beberapa penerapannya berikut ini.[12]

2.13 Keras

Keras adalah perpustakaan perangkat lunak sumber terbuka yang menyediakan antarmuka Python untuk jaringan saraf tiruan. Keras bertindak sebagai antarmuka untuk perpustakaan TensorFlow. Keras berisi banyak implementasi blok bangunan jaringan saraf yang umum digunakan seperti lapisan, tujuan, fungsi aktivasi, pengoptimal dan sejumlah alat untuk membuat bekerja dengan data gambar dan teks lebih mudah untuk menyederhanakan pengkodean yang diperlukan untuk menulis kode jaringan saraf yang dalam. Kode dihosting di GitHub, dan forum dukungan komunitas menyertakan halaman masalah GitHub, dan saluran Slack.

Selain jaringan saraf standar, Keras memiliki dukungan untuk jaringan saraf convolutional dan berulang. Ini mendukung lapisan utilitas umum lainnya seperti putus sekolah, normalisasi batch dan penyatuan[13]

2.14 Android Studio

Android Studio adalah Integrated Development Enviroment (IDE) untuk sistem operasi Android, yang dibangun di atas perangkat lunak JetBrains IntelliJ IDEA dan didesain khusus untuk pengembangan Android. IDE ini merupakan pengganti dari Eclipse Android Development Tools (ADT) yang sebelumnya merupakan IDE utama untuk pengembangan aplikasi android.[14]

2.15 Google Colabs

Google Collaboratory atau Google Colab merupakan tools yang berbasis cloud dan free untuk tujuan penelitian. Google colab dibuat dengan environment jupyter dan mendukung hampir semua library yang dibutuhkan dalam lingkungan pengembangan Artificial Intelegence (AI). Pada dasarnya google colab sama dengan Jupyter Notebook dan bisa dikatakan google colab adalah jupyter notebook yang dijalankan secara online dan gratis. Berikut adalah beberapa kelebihan dalam menggunakan google colab[15]