BAB III
METODOLOGI PENELITIAN

Bab ini membahas mengenai metode dalam penyelasaian proses meliputi pengumpulan data dan benda kerja yang mendukung kebenaran materi uraian pembahasan. Selain itu penyelesaian masalah yang ada pada sebuah perancangan sistem, maka dibutuhkan beberapa tahapan metode yang harus dilakukan. Pada bab ini dijelaskan mengenai bahan, alat dan juga metodologi penelitian yang digunakan dalam pengembangan sistem.

3.1. Alur Penelitian

Alur penelitian ini disajikan dengan berbagai tahapan. dimana tahapan tersebut terdapat beberapa langkah kerja yang dilakukan dalam melakukan penelitian dam penyelesaian suatu penelitian yang sesuai dengan topik penelitian yang dilakukan, diantaranya:

1. Studi literatur, tahapan ini dilakukan untuk mencari refrensi dalam menyelesaikan masalah.
2. Perancangan program, tahapan ini dilakukan untuk membuat listing program pada aplikasi dan juga untuk model.
3. Pengambilan *dataset*, tahapan ini dilakukan untuk mengumpulkan data yang kemudian dilakukan tahapan *training dataset* agar mendapatkan model yang sesuai.
4. Pengujian kehandalan sistem, tahapan ini dilakukan menguji program aplikasi dan juga model yang sesuai.
5. Membuat laporan, tahapan ini yaitu penulisan laporan skripsi sebagai arsip.

3.2. Metode Pengumpulan Data

Pengumpulan data bertujuan untuk mencari dan mengumpulkan data yang terkait dengan penelitian meliputi dasar teori yang terdapat pada tinjauan pustaka, metodologi penelitian, proses, dan acuan dari penelitian yang sejenis. Dalam penelitian ini, metode pengumpulan data yang dilakukan secara primer yakni mengambil data secara langsung berupa gambar biji kopi *defect* yang diperoleh

pengerajin kopi di Gunung Karang. Pengumpulan data dalam bentuk gambar yang diperoleh mencapai 1782 gambar, kemudian dari 1782 gambar tersebut kemudian dibagi menjadi 4 kategori *defect* antara lain *full black defect*, *four sour defect*, *immature defect*, dan biji kopi dalam keadaan baik. Berikut merupakan tingkatan kategori biji kopi cacat yang disajikan dalam bentuk Gambar 3.1 di bawah ini.

Gambar 3.1 Kategori cacat pada Biji Kopi, (a) *Coffea remium*, (b) *Full black defect*, (c) *Sour defect*, (d) *Immature deffect*

Gambar 3.1 merupakan kategori nilai cacat pada biji kopi. Di bawah ini merupakan *dataset* yang digunakan berdasarkan Tabel 3.1 untuk mengklasifikasi nilai cacat pada biji kopi:

Tabel 3.1 *Dataset* yang digunakan

Jenis sampel	Jumlah Sampel
Premium	467
Full Black defect	485
Sour deffect	380
Immature Beans	448

Berdasarkan Tabel 3.1 di atas merupakan Tabel dari empat sampel jenis cacat pada biji kopi, dan juga jumlah sampel yang didapat pada tiap jenisnya. Sampel ini didapatkan dari hasil photo biji kopi yang diperoleh langsung dari pengerajin kopi di Pandegelang, sehingga jumlah sampel yang diperoleh mencapai 1.782 sampel.

3.3. Perancangan Arsitektur

Penelitian pada skripsi ini menggunakan pendekatan *transfer learning* dengan metode MobileNet. Penggunaan MobileNet bertujuan dapat meningkatkan performa dari model *learning*, tetapi dengan waktu komputasi yang cenderung lebih cepat. Adapun arsitektur dari MobileNet ditunjukan Berdasarkan Tabel 3.2.

Tabel 3.2 Arsitektur MobileNet

Tipe	***Filter***	***Input***
Conv2D + *Stride* 2	3 x 3 x 3 x 32	224 x 224 x 3
Conv DW + *Stride* 1	3 x 3 x 32 DW	112 x 112 x 32
Conv2D + *Stride* 1	1 x 1 x 32 x 64	112 x 112 x 32
Conv DW + *Stride* 2	3 x 3 x 64 DW	112 x 112 x 64
Conv2D + *Stride* 1	1 x 1 x 64 x 128	56 x 56 x 64
Conv DW + *Stride* 1	3 x 3 x 128 DW	56 x 56 x 128
Conv2D + *Stride* 1	1 x 1 x 128 x 128	56 x 56 x 128
Conv DW + *Stride* 2	3 x 3 x 128 DW	56 x 56 x 128
Conv2D + *Stride* 1	1 x 1 x 128 x 256	28 x 28 x 128
Conv DW + *Stride* 1	3 x 3 x 256 DW	28 x 28 x 256
Conv2D + *Stride* 1	1 x 1 x 256 x 256	28 x 28 x 256
Conv DW + *Stride* 2	3 x 3 x 256 DW	28 x 28 x 256
Conv2D + *Stride* 1	1 x 1 x 256 x 512	14 x 14 x 256
5((Conv DW+ *Stride* 1) +(Conv2D + *Stride* 1))	3 x 3 x 512 DW 1 x 1 x 512 x 512	14 x 14 x 512 14 x 14 x 512
Conv DW + *Stride* 2	3 x 3 x 512 DW	14 x 14 x 512
Conv2D + *Stride* 1	1 x 1 x 512 x 1024	7 x 7 x 512
Conv DW + *Stride* 2	3 x 3 x 1024 DW	7 x 7 x 1024
Conv2D + *Stride* 1	1 x 1 x 1024 x 1024	7 x 7 x 1024
Average Pool + *Stride* 1	Pool 7 x 7	7 x 7 x 1024
Fully Connected Layer + *Stride* 1	1024 x 1000	1 x 1 x 1024
Softmax + *Stride* 1	*Classfication Out*	N *Input*

Berdasarkan Tabel 3.2 merupakan desain urutan Arsitektur MobileNet sebagai metode *transfer learning* yang digunakan. *Transfer Learning* pada dasarnya mengadopsi pendekatan *Convolution Neural Network* (CNN), tetapi dalam Arsitektur MobileNet terdapat 2 konvolusi yang berbeda. Pada konvolusi standar 2 dimensi berlaku proses yang ditunjukan pada Persamaan (3.1).

$$D_k * D_k * M * N * D_F * D_F \quad (3.1)$$

Pada Persamaan (3.1) merupakan persamaan standar metode konvolusi dengan indeks K dan D_k merupakan nilai dimensi spasial dari kernel konvolusi berbentuk persegi. Variabel M merupakan total *input* kanal warna yang digunakan dan nilai N merupakan keluaran dengan nilai yang telah didefinisikan dalam

algoritma. Hasil dari proses konvolusi akan didapatkan *feature map* konvolusi 2 dimensi diuraikan bentuknya pada Persamaan (3.2).

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} * F_{k+i-a,l+j-a,m} \quad (3.2)$$

Pada Persamaan (3.2) merupakan keluaran dari proses konvolusi atau dapat diartikan sebagai *feature map* yang biasa dikenal dengan ekstraksi ciri. Proses ekstraksi ciri ini sangat bergantung pada nilai parameter *stride* yang digunakan. Parameter *stride* pada Persamaan (3.2) dinotasikan dengan simbol a. Berdasarkan Tabel 3.2 uraian dari susunan Arsitektur MobileNet terdapat penggunaan *Stride* 1 dan 2. Notasi a pada Persamaan (3.2) dapat berubah menjadi 1 atau 2 bergantung pada keadaan *stride*. Proses konvolusi *Depth Wise* (DW) ditunjukan pada Persamaan (3.3).

$$D_k * D_k * M * D_F * D_F \quad (3.3)$$

Pada Persamaan (3.3) merupakan proses konvolusi DW. Pada persamaan tersebut tidak memiliki unsur variabel keluaran atau *feature map* yang umumnya dinotasikan dengan N. Proses konvolusi DW memerlukan proses tambahan yaitu *depth wise separable* untuk menyatakan nilai keluarannya. Persamaan *depth wise separable* dinyatakan pada Persamaan (3.4).

$$D_k * D_k * M * D_F * D_F + M * N * D_F * D_F \quad (3.4)$$

Pada Persamaan (3.4) merupakan *depth wise separable operation* yang bertujuan untuk mereduksi waktu komputasi. Persamaan (3.3) dan Persamaan (3.4) dinotasikan sebagai pereduksian nilai pada Persamaan (3.5).

$$\frac{D_k * D_k * M * D_F * D_F + M * N * D_F * D_F}{D_k * D_k * M * N * D_F * D_F} \quad (3.5)$$

Pada Persamaan (3.5) dapat disederhanakan menjadi Persamaan (3.6).

$$\frac{1}{N} + \frac{1}{D_K^2} \quad (3.6)$$

Berdasarkan Persamaan (3.5) dan Persamaan (3.6) dapat diketahui bahwa proses konvolusi *depth wise* mampu mereduksi waktu komputasi lebih ringkas. Kecepatan proses ini akan memberikan keuntungan pada pengembangan model

machine learning yang diimplementasikan pada *device* kapasitas kecil. Penerapan MobileNet dengan penerapan *layer depth wise* memungkinkan ekstraksi ciri dilakukan dengan lebih cepat dan dapat diterapkan pada media seperti *smartphone*.

3.4. Metode Pengembangan Sistem

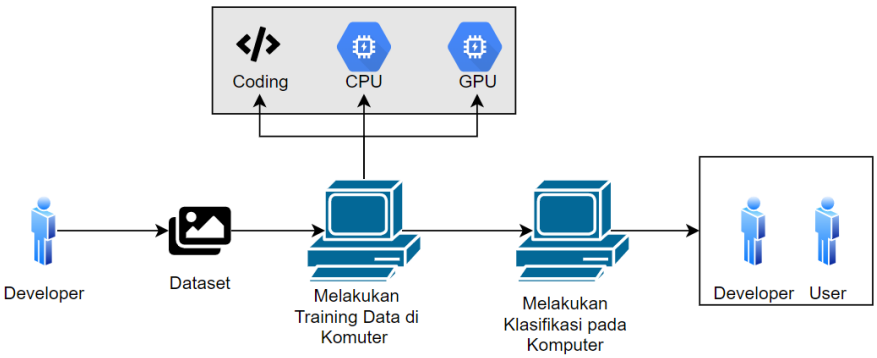
Pada metode ini berujuan untuk memetakan apa saja yang dilakukan dalam pengembangan sistem seperti tahap *requirement planning*, tahap *buildiing application*, dan tahap implementasi yang kemudian dijabarkan sebagai berikut:

3.4.1. Tahap *Requirement Planning*

Dalam tahapan ini teradapat langkah-langkah yang dilakukan untuk melakukkan pengembangan sebuah sistem, dimana sistem yang dikembangkan ialah klasifikasi berbasis Android. Tahapan yang dilakukann antara lain analisis *running system*, mengidentifikasi masalah pada sistem, menganalisis Sistem Usulan, proses training menggunakan *web-tools teachable machine*, kemudian penggunaan Arsitektur MobileNet, penggunaan *source code* Tensorflow Lite *Example*, dan komponen perangkat yang digunakan. Semuanya akan dijabarkan dan dijalaskan sebagai berikut:

1. Menganalisis running system.

   Pada hal ini merupakan alur klasifikasi gambar menggunakan *client side* yang digunakan antara lain dapat ditunjukkan pada Gambar 3.3 sebagai berikut.



Gambar 3.3 *Running System*

Berdasarkan Gambar 3.3 di atas merupakan alur klasifikasi gambar, yang dimulai dari *developer* membuat model *neural netwok*. Dalam proses klasifikasi dibutuhkan *dataset* pada penelitian ini *dataset* yang digunakan berupa gambar citra biji kopi, dengan jumlah yang sudah ditentukan. *Dataset* yang sudah didapatkan kemudian membuat model yang dilakukan dengan proses *training* dengan arsitektur *Convolutional Neural Network*, setelah model diperoleh selanjutnya melakukan proses klasifikasi menggunakan model yang sudah dibuat. Dibutuhkan komputer untuk mengklasifikasi gambar menggunakan model yang dibuat.

2. Web-tools Teachable machine, tujuan peneliti mengguanakan web-tools teachable mechine agar proses pembuatan model mudah. Pada proses pembuatan model Tensorflow lite memiliki 3 tahapan saja. Teachable machine bekerja menggunakan *framework* Tensorflow.js dengan menggunakan kemampuan *web browser* yang dapat mengakses GPU untuk melakukan komputasi yang cepat. Untuk melakukan komputasi yang cepat browser yang saat ini dilengkapi dengan WebGL yakni sebuah API yang memungkinkan *web browser* mengakses GPU untuk kebutuhan komputasi.
3. Source Code Tensorflow *Lite example*, pada peneliti mengguanakan Source Code Tensorflow *Lite example* yang bekerja secara *real-time* dengan menggunakan input yang didapat dari frame kamera dan kemudian diproses menggunakan Tensorflow *lite interpreter*. Pada *source code* tersebut memanfaatkan optimasi yang tada pada Tensorflow *lite interpreter*, yang dimana optimasi tersebut memanfaatkan kemampuan CPU, GPU, dan NNAPI yang ada pada perangkat Android untuk mempermudah proses pada Android sehingga memperoleh performa yang maksimal.
4. Komponen perangkat yang digunakan, dalam hal ini komponen pengembangan dan pengujian aplikasi klasifikasi gambar, peneliti menggunakan *software* dan *hardware*. Berikut adalah *software* dan *hardware* yang digunakan untuk membuat dan menguji aplikasi klasifikasi gambar:

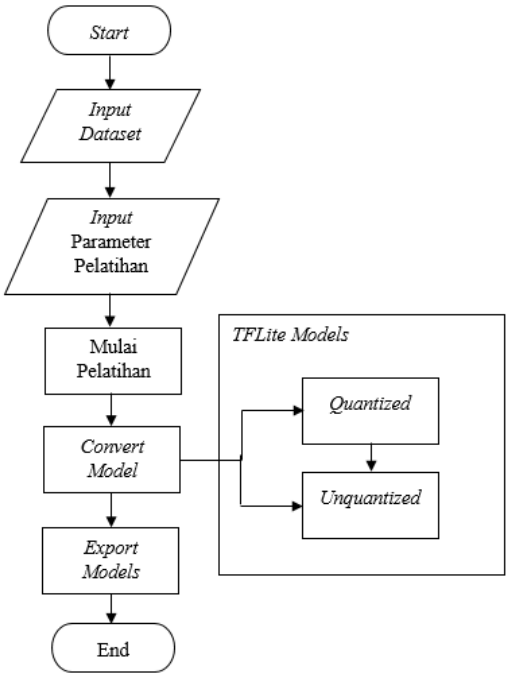
a. Perangkat pengembang, berikut adalah kebutuhan perangkat lunak maupun perangkat keras yang digunakan dalam pengembangan aplikasi klasifikasi gambar:
   1. Laptop HP am128TX
   2. OS Windows 10
   3. RAM 8GB
   4. Chrome Version 85.0.3183.121
   5. Android Studio 4.0
   6. *Web-tools Teachable machine*

b. Perangkat Pengujian, berikut ini adalah perangkat Android yang digunakan dalam pengujian aplikasi klasifikasi gambar:
   1. Xiaomi Redmi Note 8 Pro
   2. Camera 64MP
   3. Video 1080p@120fps
   4. CPU Octa-core (2x2.05 GHz Cotrex-A76 & 6x2.0 GHz Cortex-A55)
   5. GPU Mali-G76 MC4
   6. OS Android 11
   7. *Chipset* Meditak Helio G90T (12nm)
   8. RAM 6GB

3.4.2. Tahap *Build Application*

Tahapan ini terdapat langkah-langkah yang dilakukan dalam melakukan *Build Application.* Berikut tahapan yang dilakukan dalam fase ini sebagai berikut.

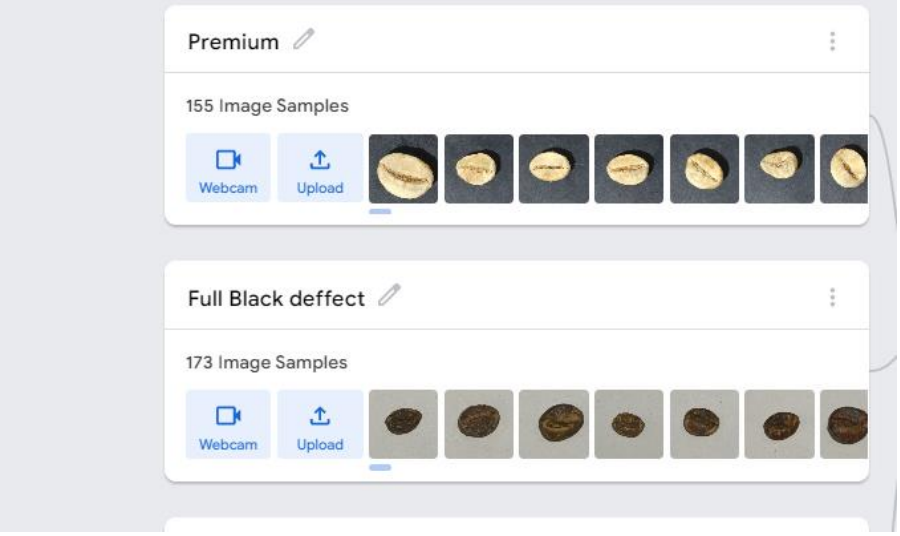
1. Perancangan model klasifikasi dengan *Pre-Trained* Mobile Net menggunakan Web-Tools Teachable Mechine, pada tahap ini ada beberapa alur yang akan dilakukan untuk memperoleh model yang digunakan, yang dimulai dari masukkan *dataset* kemudian membuat *class* atau parameter dilanjutkan *training* data, setalah dilakukan *training* data kemudian lanjut ke tahap *convert model*. *Convert model* yang dapat digunakan yakni *mobile quantized.net*. dan yang terakhir tahap *export model* agar bisa di*build* ke Android studio. Hal ini dapat dilihat pada Gambar 3.4.



Gambar 3.4 Alur Pembuatan Model Menggunakan *Web-Tools Teachable Machine*

Gambar 3.4 merupakan alur pembuatan model menggunakan *web-tools teachable machine*, menunjukkan pembuatan model dimulai pada saat *input dataset* kemudian proses selanjutnya memberikan *input* parameter pelatihan agar ditentukkan berapa *epoch* dan *batch size* nya. Kemudian dilakukan *training* setelah mendapatkan hasil *file* di-*convert* untuk mendapatkan *file* berformat tflite, setelah di-*convert file* tersebut di*export* agar bisa di-*compile.*

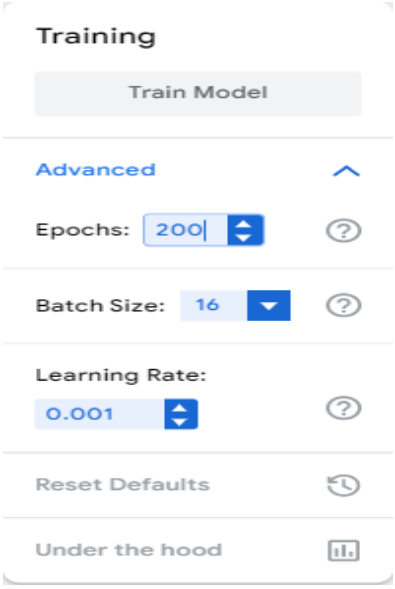
2. Pada proses pembuatan model Tensorflow *lite* untuk mendeteksi nilai cacat pada biji kopi melalui *Teachable machine* dibutuhkan *dataset* berupa gambar biji kopi yang dipisah berdasarkan jenis *defect* pada biji kopi, gambar kemudian akan dimasukkan kedalam *web-tool* untuk dilakukan proses *neural network* yang dapat di lihat pada Gambar 3.5.



Gambar 3.5 Tampilan Proses *Input Dataset* pada *Teachable machine*

Gambar 3.5 merupakan proses *input dataset* menggunakan web-tools *teachable machine*, dengan menggunakan ini, dapat memudahkan proses training.

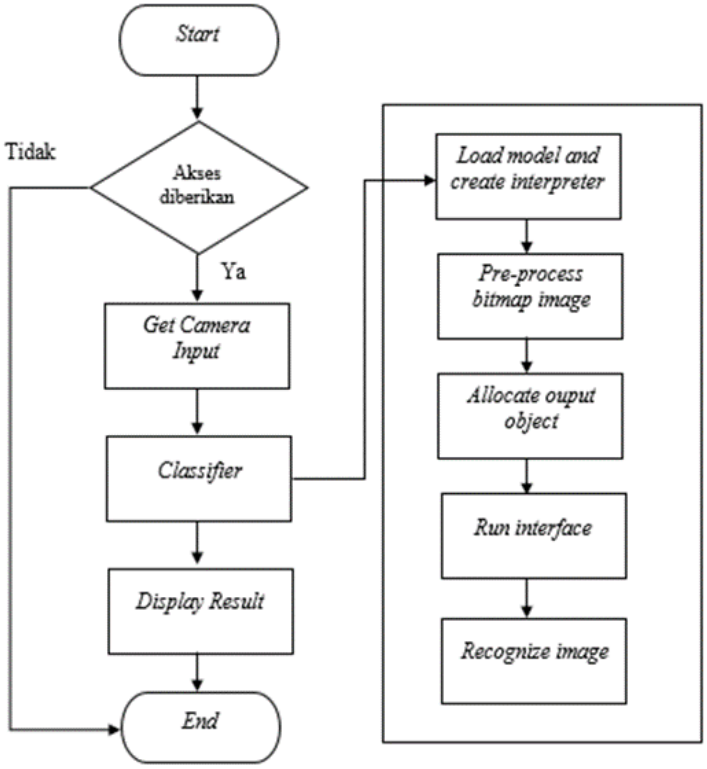
3. Setelah memasukan gambar kemudian memasukan parameter pelatihan, parameter yang dimasukan adalah *dataset, batch size,* dan *learning rate.* Berikut ini contoh parameter *training* yang terdapat di *web-tool teachable machine* pada Gambar 3.6.



Gambar 3.6 Parameter *Training*

Gambar 3.6 merupakan parameter untuk menentukkan proses training, pada proses ini dapat mengatur *Epoch*, *Batch size*, dan *Learning rate*. *Dataset* adalah parameter pelatihan yang menentukan banyaknya pelatihan yang akan dilakukan, *dataset* 200 berarti seluruh data akan melewati proses *forward* and *backward* sebanyak 200 kali. *Batch Size* adalah parameter pelatihan yang menentukan banyaknya gambar yang akan dilatih untuk tiap 1 iterasi, *batch size* 16 berarti 255 gambar akan dibagi menjadi 16 bagian dan setiap 16 gambar melewati *forward and backward* akan menghasilkan 1 iterasi, dan ketika seluruh gambar melewati *forward and backward* maka akan menghasilkan 1 *dataset*. Parameter *learning rate* berfungsi untuk menentukan berapa banyaknya perubahan pada *weight* untuk meningkatkan akurasi. Semakin kecil nilai learning rate maka model akan lebih akurat namu proses pelatihan akan berjalan semakin lambat, di penelitian ini *learning rate* yang digunakan adalah 0.001.

4. Proses *training* akan berjalan setelah *train button* diaktifkan. Proses pelatihan tersebut menggunakan metode *Transfer Learning* dimana beban dari *pre-trained* model *Mobile*Net yang sudah dilatih dengan banyak *dataset* akan digunakan kembali untuk kasus yang berbeda. Pada proses ini setiap gambar akan mengikuti arsitektur *Mobile*Net dan melakukan *convolution* sesuai arsitektur pada *Mobile*Net. Proses ini terjadi dalam *hiden* proses dimana proses yang dilakukan tidak dapat dilihat, dengan melakukan studi literatur diketahui cara kerja arsitektur *Mobile*Net mirip dengan CNN pada umumnya hanya berbeda dibagian konvolusi layernya. Perancangan sistem klasifikasi pada perangkat *Android*, untuk mengklasifikasi pada perangkat *Android* peneliti menggunakan source code Tensorflow *lite example*. Proses pada Tensorflow *lite example* dapat dilihat pada Gambar 3.7.



Gambar 3.7 Alur Klasifikasi pada Android.

Untuk menghasilkan pengambilan gambar dan juga akurasi objek yang didapatkan maka dibutuhkan konfigurasi kamera yang bertujuan menghasilkan gambar yang sesuai. Berikut ini Langkah-langkah dalam konfigurasi akses kamera.

a. Meminta akses kamera

   Agar aplikasi dapat mengakses kamera aplikasi harus meminta izin dengan memasukan *code* berikut pada *file* AndroidManifest.xml.

```
<uses-permission Android:name="Android.permission.CAMERA" /
<uses-feature   Android:name="Android.hardware.camera"   />
<uses-feature
Android:name="Android.hardware.camera.autofocus" />
```

b. Mengambil *input* kamera

   Pada *source code file* CameraActivity.java terdapat fungsi yang digunakan untuk mengambil *input* dari kamera.

c. Mengklasifikasi gambar

Pada *file* Classifier.java berisi *complex logic* untuk memproses *input kamera* dan menjalankan *inference*. Terdapat 2 sub *file* yang digunakan untuk mendemonstrasikan kedua *file float* dan *quantized* yang berada pada *ClassifierQuantizedMobile.java* dan *ClassifierFloatMobile.java*

d. Menampilkan hasil klasifikasi

Setelah hasil klasifikasi didapat selanjutnya adalah menampilkan hasil dengan fungsi *process Image*() yang berada pada *file ClassifierActivity.java*.

e. Tahap pengujian pada *teachable machine*.

Pada pengujian ini peneliti memanfaatkan fitur *under the hood* yang berada pada *web-tool* untuk mengevaluasi model yang sudah dilatih. Hasil evaluasi yang dihasilkan dalah:

1. Pengaruh *dataset* terhadap akurasi.
2. Pengaruh *dataset* terhadap nilai *loss.*
3. Akurasi model terhadap *test sample.*

f. Tahap Pengujian pada Perangkat Android

Pada pengujian ini aplikasi Android yang sudah dibuat akan diuji untuk menampilkan pengaruh sebelum dan sesudah model diaplikasikan kedalam perangkat. Pengujian yang dilakukan akan berfokus pada akurasi dan performa dari aplikasi.

3.4.3. *Implementation*

Pada implementasi pada sebuah perangkat Android dapat melalui tahapan-tahapan sebelum melakukan pengujian. Berikut merupakan tahapan-tahapan yang dilakukan sebelum pengujian diantaranya:

1. Mendefiniskan *file* pada *source code*

Pendefinisian *file* yang sudah di *input* sebelumnya, *code* yang dimasukan harus sesuai dengan lokasi dari *file*. Pada penelitian ini *file* tersebut adalah *"model_unquant.tflite", "model_quant.tflite"* proses penyesuaian berbeda pada `"java\org\Tensorflow\lite\examples\klasifikasi\tflite"`. *File* yang disesuaikan ialah:

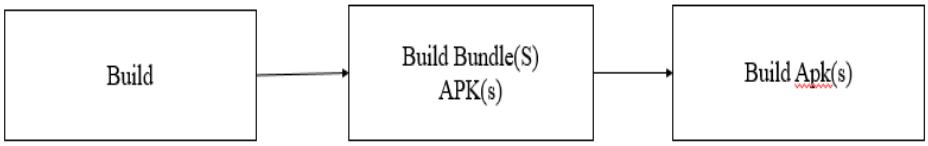
a. ClassifierFloatMobileNet.java

```
“protected String getModelPath() {
return "model_unquant.tflite"; }“
“protected String getLabelPath() {
return "labels.txt"; }”
```

b. Classifier*Quant*ized*Mobile*Net.java:

```
“protected String getModelPath() {
return " model_quant.tflite "; }”
“protected String getLabelPath(){ return "labels.txt";}
```

2. Mem-*build* aplikasi, *building* aplikasi dilakukan dengan menggunakan fitur *software* pada Android Studio dengan tahapan seperti Gambar 3.8 berikut.

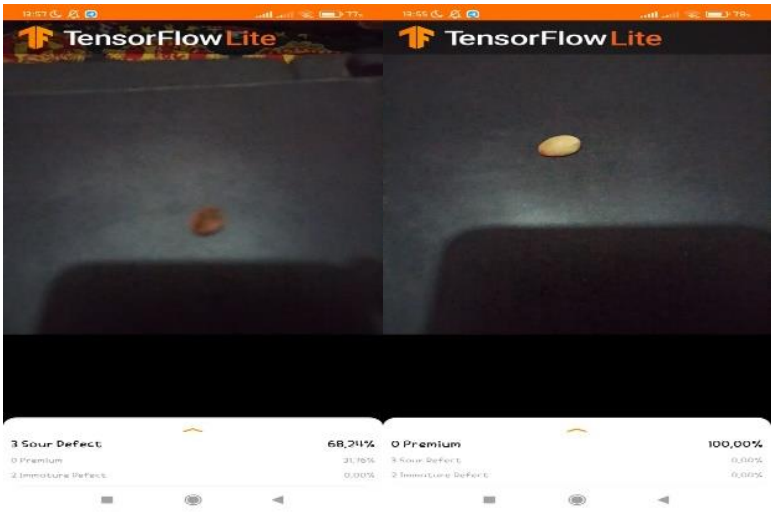


Gambar 3.8 Diagram Blok *Build* Aplikasi

Gambar 3.8 merupakan diagram blok dalam perancangan aplikasi berbasis *mobile* dengan Android Studio. Pada proses *build* aplikasi *mobile* memerlukan beberapa waktu untuk proses kompilasi, sehingga faktor penggunaan media komputer sangat diperlukan untuk meningkatkan kecepatan waktu *build* aplikasi. Proses *Build Bundle*(S) merupakan format publikasi yang menyertakan semua kode dan *resource* yang dikompilasi aplikasi yang akan dibuat. Project aplikasi tidak memerlukan banyak upaya untuk membuat *app bundle* yang mendukung aplikasi yang dioptimalkan. Saat menggunakan format *app bundle* untuk mempublikasi aplikas, secara opsional memanfaatkan *Play Feature Delivery* untuk menambahkan modul fitur ke *project* aplikasi. Modul ini berisi fitur dan resource yang hanya disertakan dengan aplikasi berdasarkan kondisi yang telah ditentukan. Pada tahapan selanjutnya membuat aplikasi yang sesuai dengan mengompilasi *resource* menggunakan *Gradle* dengan ini optimasi dan pengeloloa proses *build* custom yang fleksibel. Sehingga dapat memudahkan proses pembuatan aplikasi secara optimal dan efisien.

3.5. *Testing*

Pada tahapan ini dilakukan pengujian aplikasi. Pengujian aplikasi ini dilakukan untuk memastikan bahwa program dapat berjalan dengan baik saat digunakan.



Gambar 3.9 Testing Aplikasi

Gambar 3.9 di atas merupakan contoh hasil dari percobaan aplikasi, dalam hal ini dapat dilihat bahwa aplikasi dapat bekerja dengan jelas ketika ada objek biji kopi, aplikasi ini dapat membaca jenis biji kopi yang dideteksi beserta akurasi yang dihasilkan.