IMPLEMENTASI TENSORFLOW *LITE* UNTUK MENGETAHUI JENIS CACAT PADA BIJI KOPI ROBUSTA BERBASIS *ANDROID*

SKRIPSI

Disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik (S.T)

Disusun oleh:

Indra Wijaya

NPM.3332180011

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SULTAN AGENG TIRTAYASA

CILEGON

2023

LEMBAR PERNYATAAN KEASLIAN SKRIPSI

Dengan ini saya sebagai penulis Skripsi berikut:

Judul	: Implementasi Tensorflow Lite untuk mengetahui jenis Cacat pada Biji Kopi Robusta Berbasis Android.
Nama Mahasiswa	: Indra Wijaya
NPM	: 3332180011
Fakultas/Jurusan	: Teknik Elektro

Menyatakan dengan sesungguhnya bahwa Skripsi tersebut di atas adalah benar-benar hasil karya asli saya dan tidak memuat hasil karya orang lain, kecuali dinyatakan melalui rujukan yang benar dan dapat dipertanggungjawabkan. Apabila dikemudian hari ditemukan hal-hal yang menunjukkan bahwa sebagian atau seluruh karya ini bukan karya saya, maka saya akan bersedia dituntut melalui hukum yang berlaku. Saya juga bersedia menanggung segala akibat hukum yang timbul dari pernyataan yang secara sadar dan sengaja nyatakan melalui lembar ini.

Cilegon, 2 Desember 2022

Indra Wijaya

3332180011

HALAMAN PENGESAHAN

Dengan ini ditetapkan bahwa Skripsi berikut:

Judul	: Implementasi TensorFlow Lite untuk mengetahui jenis Cacat pada Biji Kopi Robusta Berbasis Android.
Nama Mahasiswa	: Indra Wijaya
NPM	: 3332180011
Fakultas/Jurusan	: Teknik Elektro

Telah diuji dan dipertahankan pada tanggal 13 Januari 2023 melalui Sidang Skripsi di Fakultas Teknik Universitas Sultan Ageng Tirtayasa Cilegon dan dinyatakan **LULUS**

	Dewan Penguji	Tanda Tangan
Pembimbing I	: Alief Maulana, S.T., M.T.	.....
Pembimbing II	: Rian Fahrizal, S.T., M.Eng.	.....
Penguji I	: Dr. Ing. M. Iman Santoso, M.Sc.	.....
Penguji II	: Prof. Dr. Ir. Supriyanto, M.Sc., IPM.	.....

Mengetahui

Ketua Jurusan Teknik Elektro

Dr. Romi Wiryadinata, S.T., M.Eng.

NIP.197605082003121002

PRAKATA

Puji syukur saya panjatkan kepada Allah SWT karena atas berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul Implementasi Tensorflow *Lite* untuk mengetahui jenis Cacat pada Biji Kopi Robusta Berbasis Android. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Jurusan Teknik Elektro pada Fakultas Teknik Universitas Sultan Ageng Tirtayasa. Penulis menyadari bahwa dalam proses pembuatan skripsi ini banyak menerima bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan skripsi ini. Oleh karena itu, saya mengucapkan terima kasih sebesar-besarnya kepada:

1. Kedua Orang tua yang telah mendidik dan memberikan kasih sayang dan semangatnya sehingga dapat membantu penulis dalam mengerjakan skrpsi;
2. Bapak Dr. Romi Wiryadinata, S.T., M.Eng., selaku Kepala Jurusan Teknik Elektro.
3. Bapak Alief Maulana S.T., M.T., sebagai pembimbing I yang telah memberikan arahan dan bimbingan kepada penulis selama penyusunan skripsi.
4. Bapak Rian Fahrizal S.T., M.Eng., sebagai pembimbing II yang telah memberikan arahan dan bimbingan kepada penulis selama penyusunan skripsi.
5. Ibu Dr. Irma Saraswati, S.Si., M.T., selaku dosen wali yang telah memberikan bimbingan dan juga motivasinya selama perkuliahan.

Akhir kata, penulis mohon maaf yang sebesar-besarnya apabila ada kesalahan dalam penulisan laporan ini. Semoga skripsi ini bermanfaat bagi pengembangan ilmu pengetahuan.

Cilegon, 24 September 2022

Penulis

ABSTRAK

Indra Wijaya
Teknik Elektro

Implementasi Tensorflow Lite Untuk Mengetahui Jenis Cacat
Pada Biji Kopi Robusta Berbasis Android

Kopi merupakan sejenis minuman yang berasal dari proses pengolahan biji tanaman kopi. Kopi digolongkan ke dalam famili *Rubiaceae* dengan *genus Coffea* Secara umum kopi hanya memiliki dua spesies yaitu *Arabica coffea* dan *Robusta coffea*. Tujuan dari penelitian ini ialah membuat sistem klasifikasi untuk mengetahui jenis cacat pada biji kopi robusta menggunakan metode CNN (*Convolution Neural Network*) yang sesuai dari standar. Selain itu pengklasifikasian dapat dilakukan dengan perangkat Android (*Mobile Apps*), karena memiliki kemudahan antara lain dapat digunakan dimana saja, penulis mencoba mengimplementasikan klasifikasi Objek berbasis CNN dengan menggunakan *freamwork* Tensorflow Lite, dengan Tensorflow lite dapat menjadi solusi untuk klasifikasi objek melalui perangkat Android. Penulis mencoba mengimplementasikan klasifikasi objek berbasis CNN kedalam perangkat *mobile*. *Dataset* yang digunakan bersumber dari hasil pengambilan manual menggunakan kamera *smartphone* berupa biji kopi dari daerah gunung karang banten sebanyak 1325 biji kopi. Dari hasil pebelitian aplikasi dapat berjalan pada perangkat *low and* dengan tingkat akurasi 100% dan nilai keyakinan rata-rata 97%.

Kata Kunci: Tensorflow, Tensorflow *Lite*, CNN, Biji kopi, Robusta.

ABSTRACT

Indra Wijaya
Electrical Engineering

Tensorflow Lite Implementation To Find Out Types Of Defect
On Android Based Robusta Coffee Beans

Coffee is a type of beverage that comes from the processing of coffee beans. Coffee is classified into the Rubiaceae family with the genus Coffea. In general, coffee only has two species, namely Arabica coffea and Robusta coffea. The purpose of this study was to create a classification system to determine the types of defects in robusta coffee beans using the CNN (Convolution Neural Network) method. that conforms to the standard. In addition, classification can be done with Android devices (Mobile Apps), because it has convenience, among others, can be used anywhere, the author tries to implement CNN-based object classification using the Tensorflow Lite *framework*. With Tensorflow lite can be a solution for object classification through Android devices. The author tries to implement CNN-based object classification into mobile devices. The *dataset* used is sourced from the results of manual retrieval using a smartphone camera in the form of coffee beans from the Gunung Karang area of Banten as many as 1325 coffee beans. From the research results, the application can run on low and low devices with an *accuracy* rate of 100% and an average confidence value of 97%.

Keyword: Tensorflow, Tensorflow *Lite, CNN, Coffee beans, Robusta*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1. Latar Belakang

Aplikasi saat ini merupakan teknologi yang banyak dipakai oleh kalangan masyarakat, begitu pula perkembangannya begitu pesat sehingga dapat mempermudah segala aktivitas sehari-hari. Manfaat dari aplikasi ialah bersifat mudah dan dapat digunakan dimana saja sehingga kegunaannya sangat diperlukan.

Kopi merupakan sejenis minuman yang berasal dari proses pengolahan biji tanaman kopi. Kopi digolongkan ke dalam *familiy Rubiaceae* dengan *genus Coffea* Secara umum kopi hanya memiliki dua spesies yaitu *Arabica coffea* dan *Robusta coffea*[1]. Kopi Robusta adalah salah satu jenis tanaman kopi dengan nama ilmiah *Canephora Coffea*. Nama robusta diambil dari kata *robust*, istilah dalam bahasa Inggris yang artinya kuat. Biji Kopi Robusta banyak digunakan sebagai bahan baku kopi siap saji, pencampur kopi racikan, dan juga digunakan untuk membuat minuman kopi berbasis susu seperti *cappucino, cafe latte* dan *macchiato* [2]. Struktur buah Kopi Robusta terdiri dari kulit buah yang berwarna hijau waktu masih muda dan berubah menjadi kuning lalu menjadi merah, daging buah yang berwarna putih, kulit tanduk yang merupakan lapisan biji kopi yang keras, kulit ari yang membungkus biji kopi dan biji yang mengandung unsur, zat rasa, dan aroma kopi [3]. Biji Kopi Robusta memiliki rasa yang cenderung pahit, tidak memiliki banyak karakter rasa dan lebih kekacang-kacangan. Bentuk biji bulat utuh dan ukurannya lebih kecil dari Kopi Arabika [4]. Kopi Robusta memiliki tekstur lebih kasar dari Kopi Arabika. Jenis lainnya dari Kopi Robusta seperti *Qillou, Uganda* dan *Chanepora* [5]. Pertumbuhan kopi robusta hampir sama dengan kopi arabika yakni tergantung pada kondisi tanah [6][7]. Penentuan kualitas biji kopi merupakan salah satu faktor yang mempengaruhi harga serta keunggulan kopi. Hasil survei dari beberapa daerah penghasil kopi khususnya wilayah banten mengatakan pentingnya sebuah klasifikasi biji kopi guna meningkatkan kualitas produksi.

Proses klasifikasi terhadap biji kopi membutuhkan waktu yang tidak sedikit dalam menentukannya [8]. Semakin baik kualitas kopi yang di hasilkan maka

semakin tinggi harga jual yang diberikan dan semakin tinggi pula jumlah permintaan pasar [9]. Biji kopi dapat menjadi potensi perekonomian bagi Indonesia dengan didorong oleh kemajuan teknologi dalam meningkatkan hasil dan kualitas produksi[10].

Penelitian yang berkaiatan tentang kualitas biji kopi mengenai identifikasi kualitas biji kopi dengan pendekatan kecerdasan buatan sudah dilakukan dalam penelitian lain. Metode *Convolution Neural Network* diterapkan untuk secara otomatis mengidentifikasi informasi kecacatan biji kopi arabika [11]. *Input* yang digunakan dalam penelitian ini adalah gambar biji kopi arabika dengan proses penguraian yang telah dikeringkan. Skenario yang terlibat dalam penelitian ini adalah pengumpulan data, *preprocessing*, klasifikasi dan pengujian. Pengujian dilakukan untuk dua jenis model, model 2 kelas, dan model 4 kelas. Hasil percobaan menunjukkan bahwa akurasi terbaik yang diperoleh untuk model 2 kelas adalah 82,46 % dengan menggunakan tingkat pembelajaran 0,0001, konvolusi lapisan tunggal dengan lima belas *filter* dan 100 *neuron* pada lapisan tersembunyi. Ukuran *filter* adalah $3 \times 3 \times 3$. Penambahan kelas klasifikasi menurunkan akurasi dengan perolehan akurasi terbaik 70,73% [12].

Pada penelitian lainnya yang berkaitan tentang proses sistem pengklasifikasian tingkat mutu biji kopi jenis robusta berdasarkan fitur ekstraksi warna dan tekstur dengan metode *Backpropagation Neural Network*. Pengolahan citra melalui proses ekstraksi fitur warna dan tekstur. Penelitian ini menggunakan 110 biji kopi pada tahap pelatihan, 10 biji kopi pada tahap pengujian klasifikasi jenis cacat dan 700 biji pada tahap pengujian klasifikasi tingkat mutu. Pengujian sistem dilakukan untuk mengukur akurasi klasifikasi jenis cacat yang berpengaruh pada tingkatan mutu biji kopi. Hasil pengujian yang menunjukkan akurasi terbaik berada pada nilai *learning rate* 0,001, dan jumlah *neuron* pada *hidden layer* 20 *neuron*. Sistem dapat mengklasifikasi tingkat mutu biji kopi sebesar 71,42% [12].

Permasalahan dari penelitian dan kebutuhan saat ini memiliki relevansi dengan perkembangan teknologi dan *hybrid base system.* Proses identifikasi kualitas biji kopi berdasarkan permasalahan yang ada memiliki nilai akurasi yang cukup rendah, sehingga perlu adanya peningkatan performa melalui pelatihan model *machine learning* berbasis konvolusi. Kebutuhan masyarakat akan *mobile*

system menunjukan implementasi identifikasi cepat berbasis *mobile* diperlukan sebagai dukungan upaya kebutuhan masyarakat akan teknologi.

Penelitian ini mengimplementasikan Tensorflow Lite pada Android untuk menentukan kualitas biji kopi. Tensorflow Lite dapat menjadi solusi untuk mengklasifikasi objek melalui Android [13]. Tensorflow Lite bekerja dengan mengubah model Tensorflow yang sudah jadi menjadi model TFLite, membuat proses klasifikasi menjadi lebih ringan dan dengan ukuran *file* yang lebih kecil[14]. Komputasi pada TFLite telah direduksi agar menjadi lebih cepat dan kompatibel dengan *micro device* [15]. Penelitian ini diharapkan dapat meningkatkan akurasi dan bemanfaat untuk mengenali objek yang terdapat pada gambar khususnya objek biji kopi.

1.2. Rumusan Masalah

Berdasarkan pada uraian latar belakang di atas dapat dirumuskan masalah bahwa perlunya sebuah sistem klasifikasi untuk menentukan nilai cacat pada biji kopi robusta Gunung Karang Banten menggunakan Tensorflow *lite* yang dapat di implementasikan di sistem operasi Android.

1.3. Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengidentifikasi nilai cacat pada biji kopi menggunakan sistem operasi Android sehingga:

1. Dapat mengimplentasi sistem klasifikasi yang dapat dijalankan menggunakan sistem operasi Android.
2. Merancang sebuah sistem klasifikasi yang dapat digunakan untuk menentukan nilai cacat biji kopi robusta dengan mencapai akurasi terbaik.
3. Menyederhanakan proses pembuatan model dan mengimplementasikan ke dalam perangkat *smartphone*.

1.4. Manfaat Penelitian

Manfaat yang diharapkan dari penelitian skripsi ini adalah dapat membuat rancangan sistem klasifikasi yang mudah digunakan sehingga:

1. Mempermudah proses klasifikasi sehingga dapat dijalankan di perangkat Android.
2. Mempermudah proses implementasi machine learning dan mengimplementasi modelnya.

1.5. Batasan Masalah

Dalam penelitian ini dibutuhkan adanya pembatasan masalah. Batasan masalah yang digunakan adalah:

1. Penggunaan aplikasi yang digunakan ialah Android Studio.
2. *Freamwork* yang digunakan adalah Tensorflow *Lite.*
3. Jenis biji kopi yang akan di adalah biji kopi robusta yang terletak di gunung karang.
4. Model yang digunakan menggunakan TFLite.
5. Jenis perangkat yang digunakan untuk uji coba yakni *xiaomi redmi note* 8 *Pro*.
6. *Dataset* yang digunakan ialah gambar biji kopi dengan jenis kopi robusta.
7. Masukan yang digunakan pada aplikasi ini ialah biji kopi robusta asal Gunung Karang Banten.

1.6. Sistematika Penulisan

Dalam penelitian, sistematika penulisan dapat digunakan untuk memberikan gambaran yang jelas mengenai susunan materi yang akan dibahas. Pada Bab 1 (Pendahuluan) berisi tentang latar belakang, tujuan penelitian, rumusan masalah, batasan masalah, manfaat penelitian, dan sistematika penulisan. Bab ini juga membahas secara ringkas mengenai permasalahan yang menjadi dasar dari penelitian yang akan dilakukan. Kemudian pada Bab II (Tinjauan Pustaka) memuat landasan teori yang isinya membahas dan mendukung *machine learning*, kopi robusta dan Android serta kebutuhan studi literatur dalam penelitian, selain itu terdapat beberapa review perbandingan dari metode implementasi mechine learning terkait klasifikasi kualitas biji kopi robusta.

Pada Bab III Metodologi Penelitian memuat tentang metodologi penelitian yang digunakan dengan penjabaran penelitian serta langkah penyelesaian penelitian untuk memperoleh data penelitian. Bab ini juga menjelaskan cara untuk merancang

aplikasi klasifikasi untuk mendeteksi kualitas biji kopi menggunakan Tensorflow lite dengan metode CNN yang akan digunakan. Kemudian bab ini juga membahas instrumen penelitian perangkat-perangkat pendukung yang digunakan dalam penelitian ini baik perangkat lunak maupun perangkat keras, dan yang terakhir mengenai tempat dan jadwal penelitian.

Pada Bab IV Hasil dan Pembahasan yang memuat tentang hasil penelitian dan pembahasan yang disampaikan berupa penjelasan dari hasil pengujian yang telah didapat pada penelitian tersebut.

Bab V Penutup yang berisi penjelasan dari hasil penelitian yang dijabarkan secara singkat serta menyimpulkan hasil yang telah diteliti secara ringkas dengan tujuan penelitian, dengan saran yang dituliskan untuk memberikan perbaikan yang telah dilakukan dan kemudian akan dikembangkan pada penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1. Biji Kopi Robusta

Biji kopi robusta merupakan salah satu jenis tanaman kopi dengan nama ilmiah *coffea canephora*. Nama kopi tersebut diambil dari kata robust, istilah dalam bahasa Inggris yang diartikan kuat. Minuman yang diekstrak dari biji kopi robusta biasanya memiliki citra rasa yang kuat dan cenderung lebih pahit dibandingkan arabika, berikut di bawah ini merupakan contoh dari buah kopi yang ditunjukkan pada Gambar 2.1.

Gambar 2.1 Kopi Robusta menjelang matang [2]

Gambar 2.1 merupakan buah kopi yang sudah matang dan akan siap diolah menjadi bubuk kopi. Biji Kopi robusta banyak sekali digunakan sebagai bahan baku kopi siap saji atau biasanya disebut *instant* dan juga sebagai pencampur kopi racikan untuk menambah kekuatan citra rasa kopi [2]. Selain itu kopi robusta juga bisa digunakan untuk membuat minuman kopi dengan campuran susu seperti *capucino, café latte,* dan *macchiato*. Biji kopi robusta dihargai lebih rendah dibandingkan dengan biji kopi arabika, dan Indonesia merupakan salah satu negara penghasil Kopi Robusta terbesar di dunia.

2.2. Klasifikasi Citra

Klasifikasi citra merupakan sebuah pekerjaan untuk memasukkan sebuah citra dan menetapkannya ke sebuah kategori. Salah satu permasalahan dalam

computer vision yang dapat disederhanakan dan memiliki berbagai macam aplikasinya. Salah satu aplikasi dalam klasifikasi citra adalah pengklasifikasian nama tempat pada suatu citra, pengklasifikasian jenis biji kopi, dan lain sebagainya berikut ini merupakan contoh klasifikasi citra biji kopi pada Gambar 2.2 di bawah ini.

Gambar 2.2 Klasifikasi Citra Biji Kopi [9]

Gambar 2.2 merupakan hasil klasifikasi yang terawasi, setiap masukkan citra pada *training set* diberi label. Saat klasifikasi, label tersebut akan menjadi perbandingan dengan hasil hipotesis yang diberikan oleh model pembelajaran dan akan menghasilkan nilai *error* [9]. Klasifikasi yang terawasi bisa sangat efektif dan akurat dalam mengklasifikasikan citra tempat maupun objek lainnya. Banyak metode dan algoritma yang dapat mendukung proses klasifikasi yang terawasi terutama dengan teknik *deep learning.*

2.3. *Deep Neural Network*

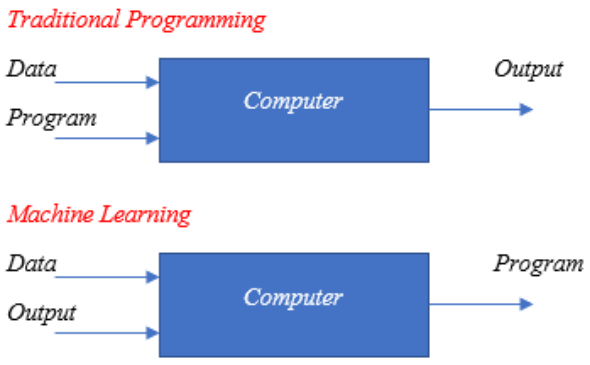
Artificial Neural Network (ANN) merupakan metode yang biasanya digunakan dalam peramalan maupun pengenalan pola. pada peramalan jaringan syaraf tiruan biasa digunakan sebagai peramalan nilai tukar mata uang asing, peramalan harga saham, peramalan cuaca, dan lain sebagainya, sedangkan untuk pengenalan pola biasanya jaringan syaraf tiruan digunakan untuk pengenalan pola huruf, pola tanda tangan hingga pola suara, serta wajah [16].

Deep learning merupakan bagian dari *machine learning* yang terdiri dari banyak lapisan yang dikenal dengan *hidden layer* dan membentuk jaringan saling terkoneksi. Lapisan tersebut adalah algoritma yang melakukan klasifikasi perintah yang di masukkan hingga menghasilkan keluaran.

Metode *deep learning* atau *deep neural network* yang sedang berkembang salah satunya adalah *convolutional neural network*. Jaringan ini menggunakan masukan berupa gambar, kemudian akan melalui lapisan konvolusi dan diolah berdasarkan *filter* yang ditentukan, setiap lapisan ini menghasilkan pola dari beberapa bagian citra yang memudahkan proses klasifikasi [17][18].

2.4. *Machine learning*

Machine learning adalah serangkaian teknik yang dapat membantu dalam menangani dan memprediksi data yang sangat besar dengan cara merepresentasikan data-data tersebut dengan algoritma pembelajaran. *Machine learning* dapat membuat komputer memprogram diri mereka sendiri [19]. Jika pemrograman adalah pekerjaan untuk membuat otomasi, maka *machine learning* mengotomatisasi proses otomasi. pada dasarnya *machine learning* membiarkan data melakukan pekerjaan [20]. Berikut Gambar 2.3 merupakan perbandingan *machine learning* dengan pemrograman secara tradisional.



Gambar 2.3 *Traditional Programming and Machine learning*

Gambar 2.3 di atas dapat dilihat bahwa pemrograman secara tradisional data dan program dijalankan di komputer untuk menghasilkan keluaran. Sedangkan

pada *machine learning* data dan keluraran yang dijalankan di komputer untuk membuat sebuah program.

2.5. *Convoluition Neural Network*

Convolutional neural network merupakan salah satu pengembangan dari jaringan syaraf tiruan yang terinspirasi dari jaringan syaraf manusia dan biasa digunakan pada data gambar untuk mendeteksi dan mengenali suatu objek pada sebuah gambar. Teknik ini terinspirasi dari cara Mamalia atau Manusia menghasilkan persepsi visual seperti yang dilakukan Mamalia dan Manusia. Secara garis besar *Convolutional Neural Network* (CNN) tidak jauh beda dengan neural network biasanya. CNN terdiri dari *neuron* yang memiliki *weight*, bias dan *activation function*. *Convolutional layer* juga terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi *(pixels)* [21].

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang di desain untuk mengolah data dua dimensi. Pada CNN, setiap *neuron* direpresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap *neuron* hanya berukuran satu dimensi. CNN termasuk dalam *deep neural network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra [22]. CNN hampir sama dengan *neural network* pada umumnya yang memiliki neuron yang memiliki bobot dan bias. CNN memiliki 1 tahap *training* (*Supervised Backpropagation*). CNN terdiri dari *neuron* yang memiliki bobot, bias, dan fungsi aktivasi.

2.6. *Feature Extraction Layer*

Proses yang terjadi pada bagian ini adalah melakukan *encoding* dari sebuah *image* menjadi *features* yang berupa angka-angka yang merepresentasikan *image* tersebut. *Feature extraction layer* terdiri dari dua bagian yaitu c*onvolutional layer* dan *pooling layer* [23].

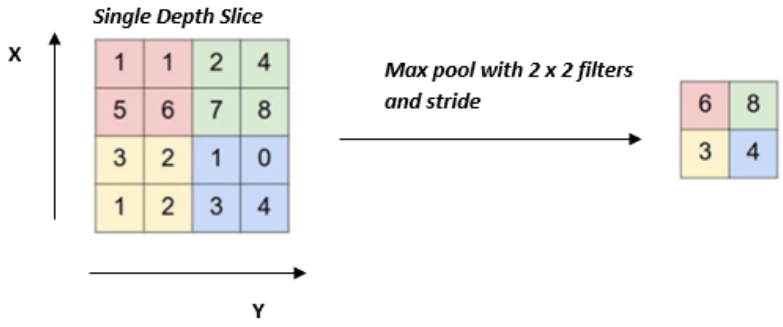
Convolutional layer terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah *filter* dengan panjang dan tinggi (*pixel*). Pada Gambar 2.6 terlihat layer pertama pada *feature extraction layer* adalah *convolutional layer* dengan ukuran $5 \times 5 \times 3$. Panjang 5 *pixel*, tinggi 5 *pixel*, dan tebal/jumlah 3 buah

sesuai dengan *channel* dari gambar tersebut. Ketiga *filter* ini akan digeser keseluruhan bagian dari gambar. Setiap pergeseran akan dilakukan operasi *dot* antara masukkan dan nilai dari *filter* tersebut sehinga menghasilkan sebuah *output* atau biasa disebut sebagai *actvation map* atau *feature map* [22].

Stride adalah parameter yang menentukan berapa jumlah pergeseran *filter*. Jika nilai *stride* adalah 1, maka conv. *filter* akan bergeser sebanyak 1 *pixel* secara *horizontal* lalu *vertical*. Semakin kecil *stride* maka akan semakin detail informasi yang kita dapatkan dari sebuah *input*, namun membutuhkan komputasi yang lebih jika dibandingkan dengan *stride* yang besar. Namun perlu diperhatikan bahwa dengan menggunakan *stride* yang kecil kita tidak selalu akan mendapatkan performa yang bagus [24].

Fungsi aktivasi berada pada tahap sebelum melakukan *pooling layer* dan setelah melakukan proses konvolusi. Pada tahap ini, nilai hasil konvolusi dikenakan fungsi aktivasi atau *activation function*. Terdapat beberapa fungsi aktivasi yang sering digunakan pada *convolutional network*, di antaranya reLU. Aktivasi reLU menjadi pilihan bagi beberapa peneliti karena sifatnya yang lebih berfungsi dengan baik.

Fungsi yang digunakan untuk aktivasi pada reLU, fungsi reLU adalah nilai *output* dari neuron bisa dinyatakan sebagai 0 jika *input*nya adalah negatif. Jika nilai *input* dari fungsi aktivasi adalah positif, maka *output* dari *neuron* adalah nilai *input* aktivasi itu sendiri. Berikut merupakan contoh dari *max pooling layer* pada Gambar 2.4 di bawah ini.



Gambar 2.4 Contoh *Max Pooling* [10]

Gambar 2.4 di atas, lapisan *pooling* menggunakan salah satu operasi maksimal yang merupakan operasi yang paling umum. Gambar 2.4 menunjukkan operasi dengan langkah 2 dan ukuran *filter* 2×2. dari ukuran *input* 4×4, pada masing-masing 4 angka pada *input* operasi mengambil nilai maksimalnya dan membuat ukuran *output* baru menjadi 2×2 [25].

2.7. Keras

Keras merupakan *neural network library* Bahasa Python yang mudah digunakan. Keras mampu berjalan di atas Tensorflow, Microsoft Cognitive Toolkit, R, Theano, atau PlaidML [26]. Keras dirancang untuk menjalankan eksperimen dengan *deep neural network* dengan cepat.

Keras berfokus untuk mudah digunakan, modular, dan dapat diperluas. Keras di kembangkan sebagai salah satu dari hasil research *pada project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System).* Penulis utama dan pengembang adalah *Francois Chollet*, seorang *Google engineer. Chollet* juga adalah seorang pembuat deep dari *XCeption neural network* model [27][28] .

2.8. Bahasa Pemrograman

Python merupakan salah satu contoh bahasa tingkat tinggi. Contoh lain bahasa tingkat tinggi adalah Pascal, C++, Pert, Java, dan sebagainya. Sedangkan bahasa tingkat rendah merupakan bahasa mesin atau bahasa *assembly*. Secara sederhana, sebuah komputer hanya dapat mengeksekusi program yang ditulis dalam bentuk bahasa mesin. Oleh karena itu, jika suatu program ditulis dalam bentuk bahasa tingkat tinggi, maka program tersebut harus diproses dulu sebelum bisa dijalankan dalam komputer. Hal ini merupakan salah satu kekurangan bahasa tingkat tinggi yang memerlukan waktu untuk memproses suatu program sebelum program tersebut dijalankan.

Bahasa tingkat tinggi mempunyai banyak sekali keuntungan. Bahasa tingkat tinggi mudah dipelajari, mudah ditulis, mudah dibaca, dan tentu saja mudah dicari kesalahannya. Bahasa tingkat tinggi juga mudah diubah porTabel untuk disesuaikan dengan mesin yang menjalankannya. Hal ini 27 berbeda dengan bahasa mesin yang hanya dapat digunakan untuk mesin tersebut, dengan berbagai kelebihan ini, maka

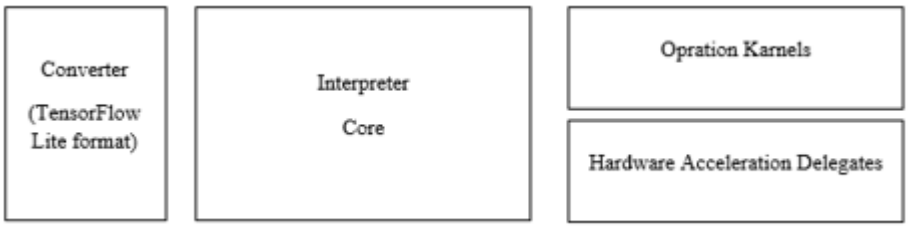
banyak aplikasi ditulis menggunakan bahasa tingkat tinggi. Proses mengubah dari bentuk bahasa tingkat tinggi ke tingkat rendah datam bahasa pemrograman ada dua tipe, yakni *interpreter* dan *compiler* [29].

Java adalah nama sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer yang berdiri sendiri (*standalone*) ataupun pada lingkungan jaringan [33]. Java merupakan bahasa pemrograman yang membangun struktur Android Studio.

Android Studio merupakan sebuah *Integrated Development Environment* (IDE) khusus untuk membangun aplikasi yang berjalan pada platform Android. Android Studio ini berbasis pada IntelliJ IDEA, sebuah IDE untuk Bahasa pemrograman Java. Bahasa pemrograman utama yang digunakan adalah Java, sedangkan untuk membuat tampilan atau layout, digunakan bahasa XML. Android studio juga terintegrasi dengan Android *Software Development Kit* (SDK) untuk mendeploy ke perangkat Android [34].

2.9. Tensorflow Lite

Tensorflow Lite adalah seperangkat alat yang dikembangkan oleh Tim *Google Brain* untuk membantu *developers* menjalankan model Tensorflow pada perangkat seluler, sistem tertanam, dan IoT [30]. Tensorflow *Lite* sendiri mempunyai keunggulan yaitu mempunyai sistem optimisasi yang membuat proses klasifikasi menjadi lebih ringan pada perangkat *mobile*. Berikut ini adalah beberapa optimisasi yang berada pada komponen Tensorflow *Lite* pada gambar 2.5 di bawah ini.



Gambar 2.5 Arsitektur Tensorflow *Lite*

Gambar 2.5 diatas merupakan tampilan umum pada arsitektur Tensorflow *Lite* yang terdiri dari *converter*, *interpreter*, *opration karnels*, dan *hardware*

acceleration. Arsitektur ini dapat diimplementasikan kedalam perangkat Android, sehingga pengguna dapat dengan mudah menggunakannya.

Tensorflow Lite *Converter* menggunakan model Tensorflow yang sudah dilatih untuk membuat TFLite Model. TFLite model merupakan Tensorflow model yang ukurannya sudah dikecilkan dan mampu menampilkan isi dari model tersebut. Flatbuffer merupakan format yang digunakan Tensorflow dan berperan penting untuk mengefisienkan model, membuat akses kedalam data lebih cepat selagi membuat ukurannya lebih kecil. Ada beberapa cara untuk membuat Tensorflow *Lite* model salah satunya adalah dengan menggunakan Python API [31]. Python API berfungsi untuk mempermudah mengkonversi model untuk digunakan sebagai bagian dari pengembangan model *pipeline* dan membantu mengurangi masalah lebih awal.

Interpreter Core atau TFLite *interpreter* berfungsi sebagai pengeksekusi agar TFLite model bisa berjalan di perangkat *mobile* menggunakan operator Tensorflow yang sudah dibatasi, dengan membatasi *default operators, library*, dan *tools* yang digunakan untuk menjalankan TFLite model, ukuran *Interpreter Core* sudah dikecilkan menjadi sekitar ~100KB[14].

Optimisasi pada Tensorflow *Lite* sudah sampai ketahap *hardware*. Untuk bekerja pada perangkat dengan banyak batasan berarti prosesor pada perangkat harus digunakan dengan sangat efisien. Tensorflow Lite NNAPI sudah ada pada semua Android dengan bersi 8.1 (API *Level* 27) ke atas, NNAPI berfungsi memberikan akselerasi terhadap Tensorflow Lite model pada Android dengan menggunakan akselerasi *hardware* seperti:

a. *Graphics Processing Unit* (GPU).
b. *Digital Signal Processor* (DSP).
c. *Neural Processing Unit* (NPU).
d. GPU Deligate

Tensorflow Lite juga mendukung akselerasi menggunakan GPU secara langsung. GPU dapat memproses data dengan ukuran 16 bit hingga 32 bit floating *point*, sehingga tidak dibutuhkan *quantization* untuk mendapatkan performa yang optimal [32].

2.10. Teachable Machine

Teachable Machine adalah suatu aplikasi berbasis *website* atau *web tool* yang berfungsi untuk mempermudah proses pembuatan *machine learning* sehingga dapat dilakukan semua orang. Teachable machine dapat dilakukan hanya dengan 3 tahap yaitu mengumpulkan data, melatih data, dan mengeskpor atau menyimpan model. Pada Gambar 2.6 merupakan proses dalam penggunaan Teachable Machine dalam melakukan *export* dan *training* model.

Gambar 2.6 Proses Menggunakan Teachable Machine, (a) *Gather* mengumpulkan data, (b) *Train* memproses data, (c) Export mengekstrak data

Gambar 2.6 di atas merupakan proses dari Teachable Machine yang bekerja dengan menggunakan Ternsorflow.js. Tensorflow.js merupakan sebuah *library* untuk *machine learning* yang berjalan di javascript yang berfungsi untuk melatih dan menjalankan model yang dibuat melalui *web browser* [35].

2.11. Kajian Pustaka

Pada penelitian skripsi ini mengacu pada jurnal-jurnal terkait dalam pembahasan yang sama. Kajian akan meliputi kekurangan serta kelebihan yang kemudian menjadi bahan pertimbangan dalam menambahkan metode yang akan digunakan, serta analisis yang harus ditambahkan pada penelitian ini. Penelitian yang berkaiatan tentang kualitas biji kopi sudah pernah dilakukan yakni, dalam proses pengemasan saat ini, penyortiran ini dilakukan secara manual. *Convolution Neural Network* diterapkan untuk secara otomatis untuk mengetahui informasi kecacatan Biji Kopi Arabika [11]. *Input* yang digunakan dalam penelitian ini adalah gambar Biji Kopi Arabika dengan proses penguraian yang telah dikeringkan. Sekenario yang terlibat dalam penelitian ini adalah pengumpulan data, preprocessing, klasifikasi dan pengujian. *Preprocessing* dilakukan dengan

memotong beberapa cakupan objek biji kopi yang hanya berisi gambar biji kopi. Klasifikasi dilakukan oleh CNN, untuk mendapatkan akurasi model yang terbaik, parameter yang ada harus diuji dan dievaluasi. Pengujian dilakukan untuk dua jenis model, model 2 kelas dan model 4 kelas. Hasil percobaan menunjukkan bahwa akurasi terbaik yang diperoleh untuk model 2-kelas adalah 82,46% dengan menggunakan tingkat pembelajaran 0,0001, konvolusi lapisan tunggal dengan lima belas *filter* dan 100 *neuron* pada lapisan tersembunyi. Ukuran *filter* adalah 3x3x3. Sedangkan model 4-kelas memperoleh akurasi terbaik 70,73% dengan dua lapisan konvolusional. Jumlah *filter* di setiap lapisan adalah 6 *filter* dengan ukuran 3x5x5 di lapisan pertama dan 18 *filter* dengan ukuran 6x3x3 di lapisan kedua.

Penelitian selanjutnya yang berkaitan dengan kopi yang sudah dilakukan, dalam penelitian ini kualitas biji kopi secara konvensional ditentukan oleh visual inspeksi, yang subjektif, membutuhkan upaya yang cukup besar dan waktu sehingga rawan dari kesalahan. Karakteristik biji kopi dari berbagai kota di *Cavite* [5]. Teknik pencitraan yang secara otomatis mengklasifikasi sampel biji kopi menurut jenis kopi. Yang menjadi titik uji pada penelitian ini berkaitan dengan spesifikasi biji kopi itu sendiri seperti luas, keliling serta diameter dari biji kopi tersebut dan presnetasi kebulatan yang diekstarsi dari 195 gambar dan 60 pengujian gambar. Pada penelitian ini teknik pengolahan citra yang digunaka yakni *Artificial neural network* (ANN) and *K nearest neighbor* (KNN), dengan skor klasifikasi 84,12%(k=1), 84,10%(k=2), 81,53%(k=3), 82,56%(k=4), 75,38%(k=5),80,35% (k=6), 38,79%(k=7), 77,44%(k=8), 72,82%(k=9) dan 78,45% (k=10). dari hasil penelitian tersebut dapat disimpulkan bahwa teknik tersebut dapat digunakan sebagai metode efektif untuk klasifikasi jenis biji kopi.

Penelitian selanjutnya yang berkaitan dengan kopi yang sudah dilakukan, dalam penelitiannya yakni mengklasifikasi tiga jenis kopi arabika menggunakan *computer vision* dengan metode klasifikasi menggunakan alexnet yang merupakan arsitektur CNN dengan anaisis beberapa variasi seperti SGDm, dan RMSProp dan *learning rate* 0,00005 dan 0,0001 [37]. Setiap jenis kopi menggunakan 500 data untuk pelatihan dan validasi dengan distribusi 70% pelatihan dan 30% validasi. Hasilnya menunjukkan bahwa semua Model AlexNet mencapai akurasi validasi yang sempurna nilai 100% dalam 1.040 iterasi. Penelitian ini juga menggunakan

100 data *testing-set* pada setiap jenis kopi kacang. Pada pengujian confusion matrix, akurasi mencapai 99,6%. Penelitian selanjutnya yang berkaitan dengan kopi yang sudah dilakukan, menjelaskan proses klasifikasi menggunakan metode CNN dengan akaurasi 93,33% dalam klasifikasi. Pengembangan aplikasi Android yang digunakan untuk klasifikasi menggunakan Tensorflow dan MobileNetV2 dengan datasheet 500 citra yang didistribusikan dalam lima kelas dengan 80% gambar yang digunakan untuk *training* dan 20% untuk validasi nya, dan hasilnya 80% akurasi pengenalan empat kelas, kinerja dari aplikasinya dengan metriks yang lebih tinggi dari 75% untuk tiga kelas yang di-identifikasi.

Penelitian selanjutnya yang berkaitan dengan *Image Clasification* yang sudah dilakukan, dalam penelitian ini klasifikasi menggunakan metode deep learning yang masukkan (input data) berupa citra yang menghasilkan sebuah pola dari beberapa bagian citra yang nantinya akan mudah diklasifikasikan. Pengklasifikasian ini diimplementasikan pada kebun dan sawah, dengan tujuan untuk membedakan karakteristiknya. Berdasarkan hasil klasifikiasi diperoleh hasil akurasi testing 75%. Dapat disimpulkan bahwa metode CNN dapat mengklasifikasikan citra kebun dan sawah dengan baik [23].

Penelitian selanjutnya yang berkaitan dengan *Image Clasification* yang sudah dilakukan, dalam penelitian ini klasifikasi menggunakan metode Deep learning (DL) yang dapat digunakan untuk mendeteksi dan mengenali suatu objek pada citra digital. Hasil yang didapatkan pada penelitian ini bahwa aplikasi android dapat berjalan dengan baik dengan akurasi sebesar 92. 33% dapat dilihat dari hasil pengujian dengan menggunakan metode 10-fold cross validation, semua menu yang tersedia dapat dijalankan dan penyebutan label objek sudah sesuai untuk pengenalan dan klasifikasi citra. Perhitungan presisi dan recall memiliki nilai yang baik, masing-masing sebesar 97,51% dan 94,33%. Pada proses klasifikasi, objek yang tidak ada pada dataset yang telah dimodelkan oleh sistem akan menjadi null atau tidak dikenali, terutama pada citra objek yang ditangkap oleh kamera Android yang memiliki banyak objek dan berdekatan [16].

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

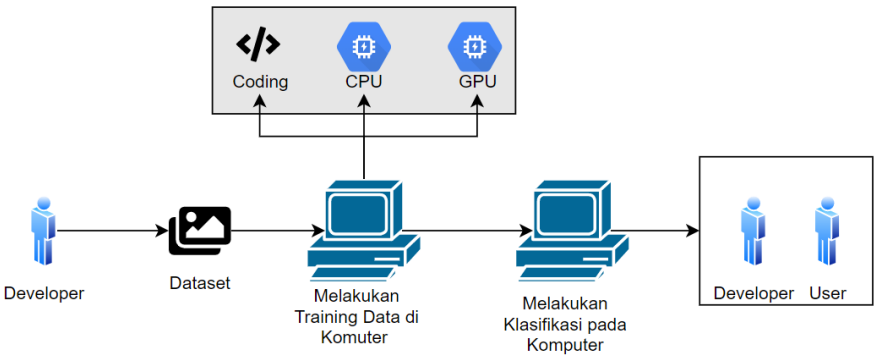
Pada metode ini berujuan untuk memetakan apa saja yang dilakukan dalam pengembangan sistem seperti tahap *requirement planning*, tahap *buildiing application*, dan tahap implementasi yang kemudian dijabarkan sebagai berikut:

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:

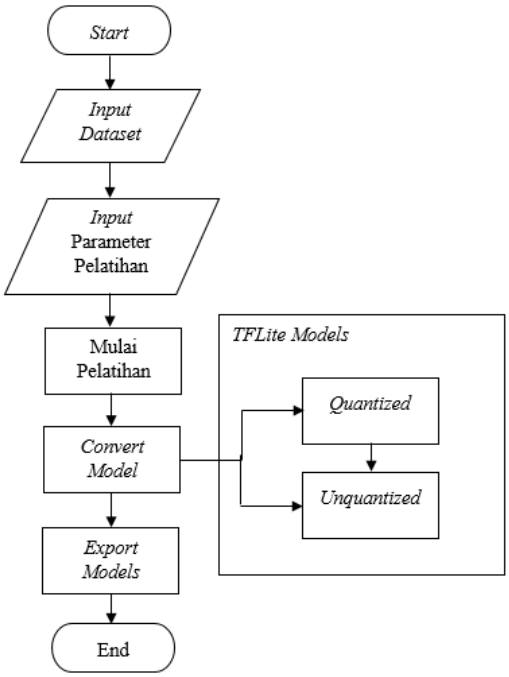
a. Perangkat pengembang, berikut adalah kebutuhan perangkat lunak maupun perangkat keras yang digunakan dalam pengembangan aplikasi klasifikasi gambar:
   1. Laptop HP am128TX
   2. OS Windows 10
   3. RAM 8GB
   4. Chrome Version 85.0.3183.121
   5. Android Studio 4.0
   6. *Web-tools Teachable machine*

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.

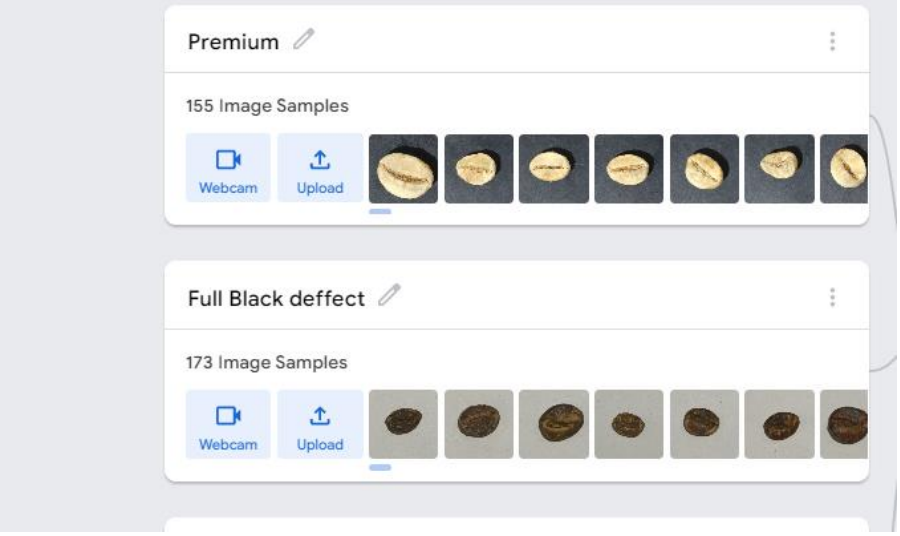
1. Perancangan model klasifikasi dengan *Pre-Trained* Mobile Net menggunakan Web-Tools Teachable Mechine, pada tahap ini ada beberapa alur yang akan dilakukan untuk memperoleh model yang digunakan, yang dimulai dari masukkan *dataset* kemudian membuat *class* atau parameter dilanjutkan *training* data, 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.*

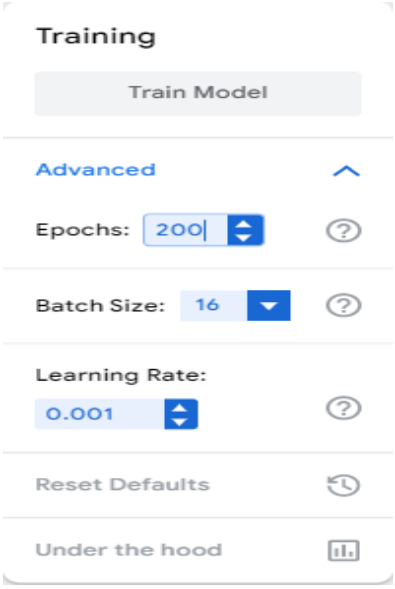
2. Pada proses pembuatan model Tensorflow *lite* untuk mendeteksi nilai cacat pada biji kopi melalui *Teachable machine* dibutuhkan *dataset* berupa gambar biji kopi yang dipisah berdasarkan jenis *defect* pada biji kopi, gambar kemudian akan dimasukkan kedalam *web-tool* untuk dilakukan proses *neural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

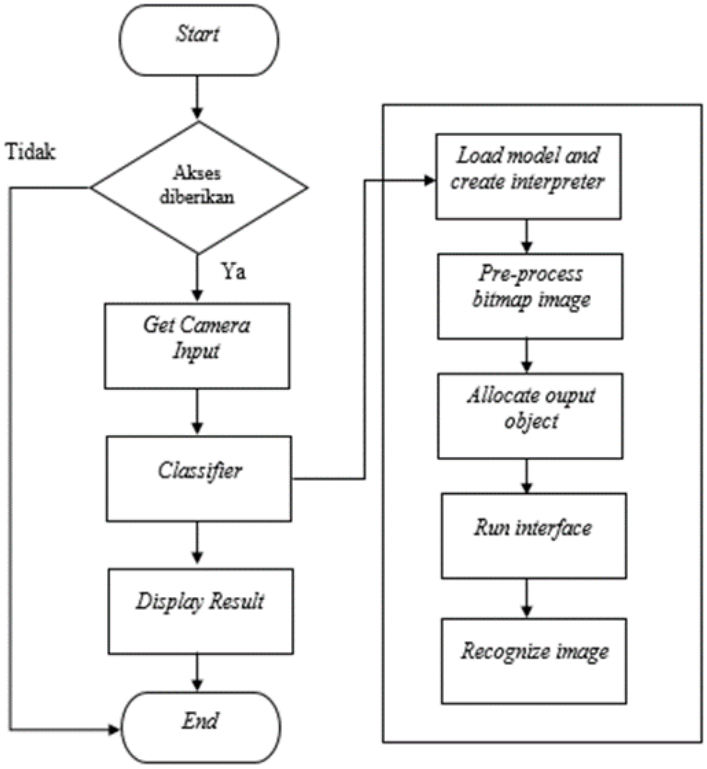
3. Setelah memasukan gambar kemudian memasukan parameter pelatihan, parameter yang dimasukan adalah *dataset, batch size,* dan *learning rate.* Berikut ini contoh parameter *training* yang terdapat di *web-tool teachable machine* pada Gambar 3.6.



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:

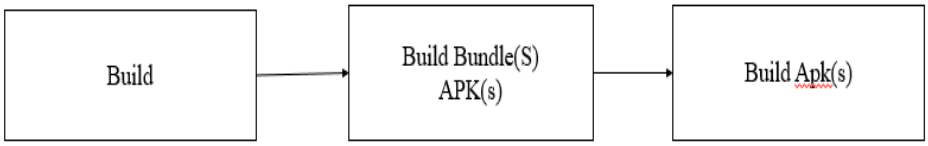
a. ClassifierFloatMobileNet.java

```
“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.

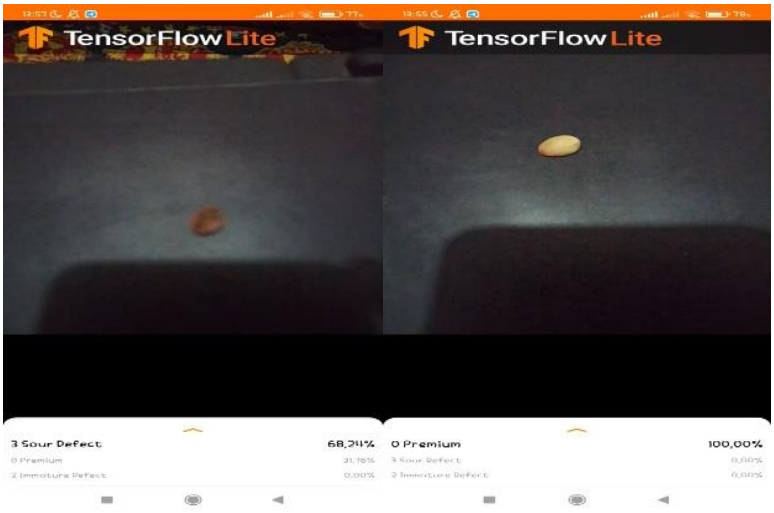


Gambar 3.8 Diagram Blok *Build* Aplikasi

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

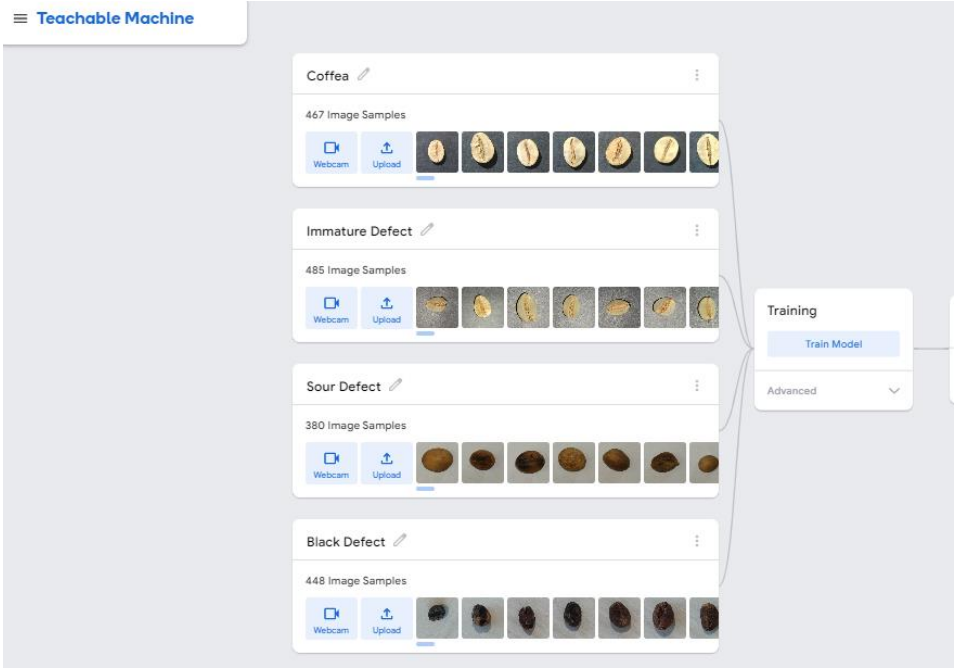
Gambar 3.9 di atas merupakan contoh hasil dari percobaan aplikasi, dalam hal ini dapat dilihat bahwa aplikasi dapat bekerja dengan jelas ketika ada objek biji kopi, aplikasi ini dapat membaca jenis biji kopi yang dideteksi beserta akurasi yang dihasilkan.

BAB IV
HASIL DAN PEMBAHASAN

4.1. Tahap Persiapan *Dataset*

Pada tahap ini data yang berupa gambar akan dilakukan proses *training* agar dapat menghasilkan model yang sesuai. Proses ini menggunakan sebuah *web-tools teachable machine* yakni sebuah aplikasi berbasis web yang digunakan untuk membuat model klasifikasi.

Gambar yang akan diklasifikasi berupa gambar biji kopi yang terbagi menjadi empat *class* yakni *coffea*, *Immature Defect*, *Sour Defect*, dan *Black Defect.* Pengelompokkan biji kopi ini menggunakan jenis kopi robusta yang dikategorikan pada bentuk *defect* atau cacat. Sehingga dapat membedakan objek gambar yang akan dihasilkan. Berikut ini merupakan tampilan *web tools teachable machine* pada Gambar 4.1.



Gambar 4.1 Tampilan *Web Teachable machine*

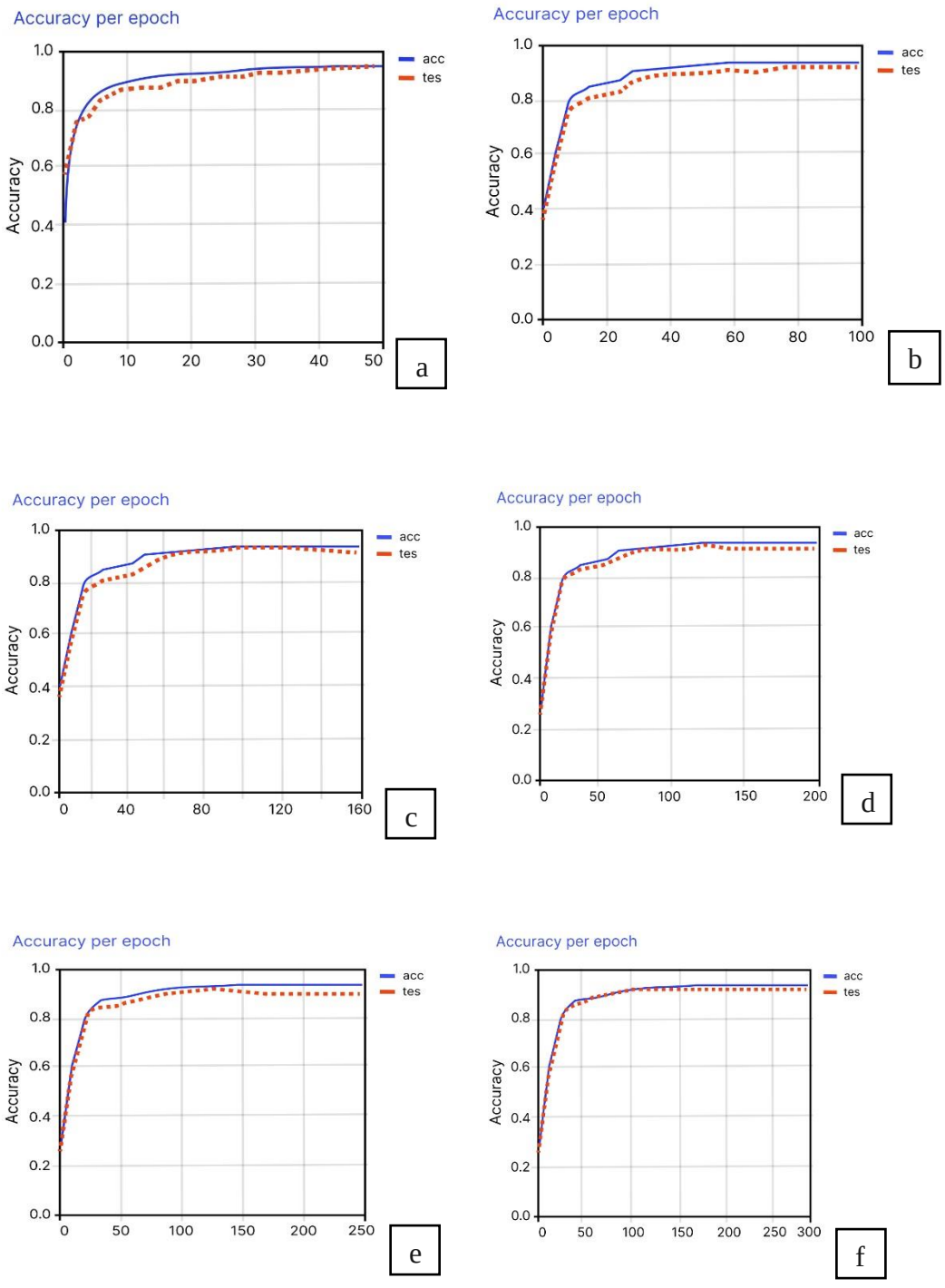
Berdasarkan Gambar 4.1 merupakan tampilan dari hasil *input dataset* biji kopi yang akan di-*training* menggunakan *web tools teachable machine* pada proses ini *dataset di-training* dengan mempertimbangkan jenis *dataset*, *batch size* dan juga *learning rate.*

Dataset merupakan *hyperparameter* yang menentukan berapa kali algoritma *deep leraning* bekerja melewati seluruh *dataset*, yang artinya satu *dataset* tercapai ketika semua batch telah dilewatkan melalui *neural network* satu kali. *Batch size* merupakan jumlah sampel *dataset* yang akan dilalui dalan satu waktu.

Pada *batch size* ini dapat menentukan jumlah sampel yang akan dikerjakan. *Learnig rate* merupakan salah satu parameter *training* untuk menghitung nilai koreksi bobot pada waktu proses *training*. Pada hal ini *learning rate* menjadi parameter seberapa cepat proses *training* yang akan dilakukan.

4.2. Pengaruh *Dataset* Terhadap Model

Pada tahap pengujian yang dilakukan dengan mengetahui baiknya sebuah model *neural network* yang dibuat dengan membandingkan *dataset* terhadap model sehingga dapat menjadi bahan acuan dalam menentukan tingkat akurasi model yang diinginkan. Pada penelitian ini digunakan enam kategori *epoch* yang berbeda yaitu 50, 100, 150, 200, 250, 300 dengan batch size 64 dengan *dataset* yang sama.Variasi *epoch* dilakukan untuk menemukan model *learning* terbaik dan ideal pada kasus penelitian kualitas biji kopi ini. *Epoch* yang rendah berpotensi untuk mengalami kegagalan dalam pembelajaran model, sementara pada kasus *epoch* yang lebih tinggi berpotensi terjadinya *overfitting*. *Epoch* standar dalam *machine learning* adalah 100 iterasi, tetapi pada pembelajaran yang lebih berat penambahan hingga 500 *epoch* dapat dipertimbangkan. Hasil dari variasi *epoch* akan ditunjukan dalam bentuk hasil *training* berupa grafik *train accuracy* dan *train testing* yang biasa disebut dengan *validation*. Model yang baik merupakan model yang mencapai tingkat konvergensi yang ditandai dengan tidak adanya perubahan nilai pada parameter akurasi mendekati 1. Hasil dari visualisasi proses *training* dengan variasi *epoch* ditunjukan pada Gambar 4.2.



Gambar 4.2 Hasil Training *Dataset* per *Epoch*, (a) epoch 50, (b) epoch 100, (c) epoch 150, (d) epoch 200, (e) epoch 250, (f) epoch 300.

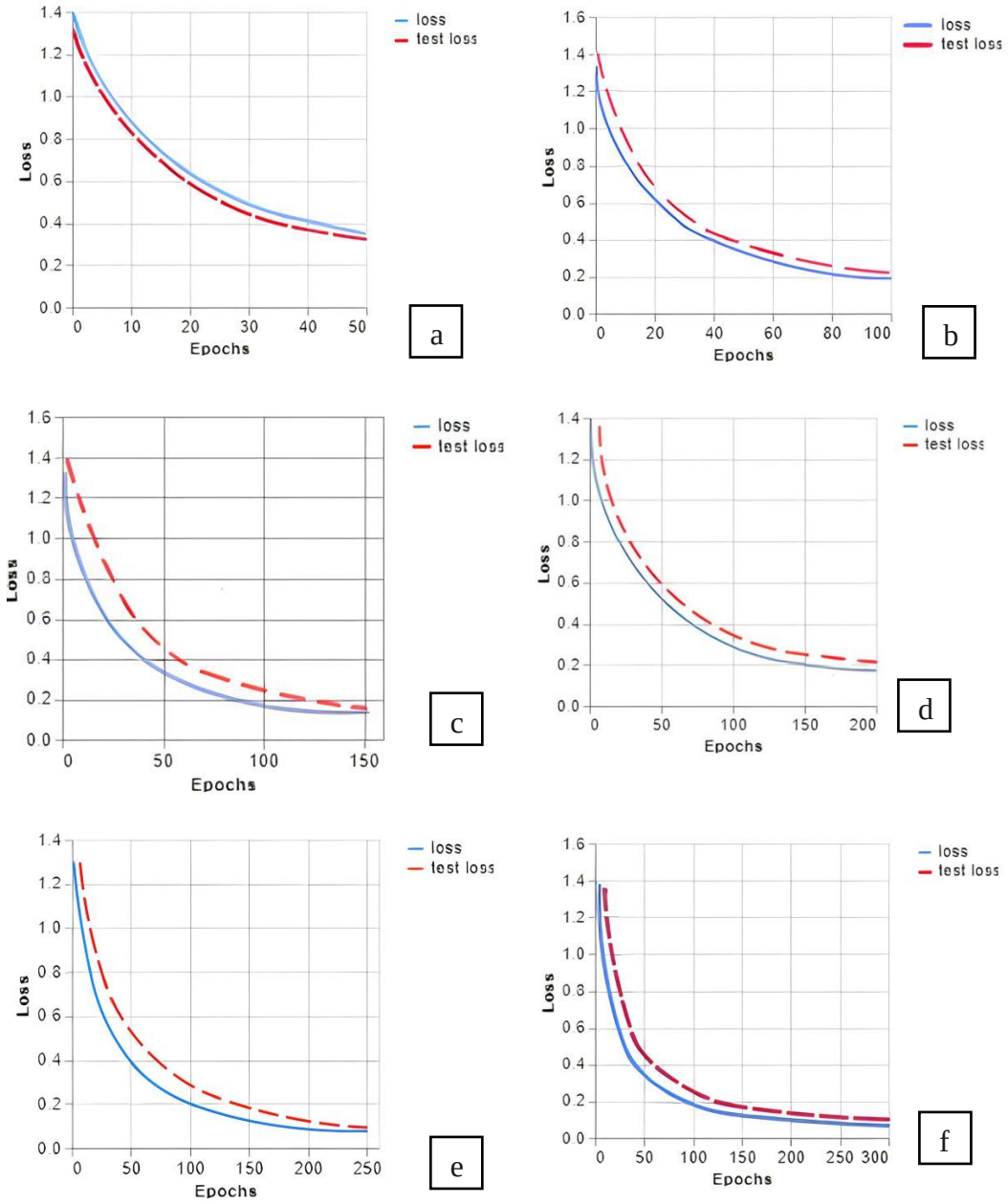
Gambar 4.2 merupakan hasil dari *training dataset* yang dilakukan. Dapat dilihat peningkatan akurasi biji kopi per-*dataset*, pada peniltian ini terdapat 6 parameter *dataset* untuk menentukan jenis model apa yang akan digunakan pada pengujian aplikasi. Gambar 4.2 (a) *Epoch* 50 memiliki akurasi sebesar 93% dengan test akurasi sebesar 89% dan *test sample* akurasi sebesar 89%, dalam hal ini *test* akurasi bertujuan untuk mengidentifikasi model yang dilatih terhadap gambar *independent* yang tidak digunakan dalam *training*. Akurasi bertujuan untuk mengidentifikasi gambar yang digunakan baik untuk *training* dan juga testing. Sehingga selisih dari akurasi dan test akurasi sebesar 4%. Gambar 4.2 (b) *Epoch* 100 memiliki akurasi sebesar 95% dengan test akurasi sebesar 91%. Sehingga selisih dari akurasi dan test akurasi Berdasarkan Tabel *dataset* 100 sebesar 4%. Gambar 4.2 (c) *Epoch* 150 memiliki akurasi sebesar 95% dengan *test* akurasi sebesar 95%. Sehingga selisih dari akurasi dan test akurasi pada Gambar *Epoch* 150 sebesar 0%. Gambar 4.2 (d) *Epoch* 200 memiliki akurasi sebesar 96% dengan test akurasi sebesar 93%. Selisih dari akurasi dan test akurasi pada Gambar *Epoch* 200 sebesar 3%. Gambar 4.2 (e) *Epoch* 250 memiliki akurasi 96% dengan test akurasi sebesar 95%. Sehingga selisih dari akurasi dan test akurasi pada Gambar *Epoch* 250 sebesar 1%. Gambar 4.2 (f) *Epoch* 300 memiliki akurasi 97% dengan test akurasi sebesar 95%. Sehingga selisih dari akurasi dan test akurasi Berdasarkan Tabel 300 sebesar 2%.

4.3. Pengaruh *Loss* Terhadap Model

Pada pengujian yang dilakukan dengan mengetahui nilai *loss*, yakni nilai dari perhitungan *loss function* dari *training dataset* dan prediksi dari model. Pengujian ini akan terfokus pada konvergensi nilai *loss* model yang telah dilakukan proses training dengan *setting epoch* sebesar 50, 100, 150, 200, 250, 300. Konvergensi *loss* dapat diindikasikan dengan tidak adanya perubahan nilai *loss* secara fluktuatif dan nilai mendekati nilai 0.

Variasi rendah pada nilai *epoch* akan memberikan waktu *training* yang lebih cepat, tetapi kemungkinan gagal *learning* yang cukup tinggi. Pada kasus variasi yang lebih tinggi dapat memberikan waktu untuk melakukan proses *training*, tetapi

dengan membutuhkan waktu yang lebih lama. Pada Gambar 4.3 merupakan visualisasi hasil *training* dengan hasil *loss training*.



Gambar 4.3 Hasil *Loss Training*, (a) *Epoch* 50, (b) *Epoch* 100, (c) *Epoch* 150, (d) *Epoch* 200, (e) *Epoch* 250, (f) *Epoch* 300

Gambar 4.3 di atas merupakan hasil *loss training* pada tiap *dataset* yang dilakukan. Pada penelitian ini mengetahui nilai *loss* dan *test loss* dengan tujuan mengetahui antara keluaran yang dibuat oleh algoritma yang sedang di training

dengan keluaran yang diharapkan. Pada *function loss* ini akan mengevaluasi seberapa baik algoritma dalam memodelkan data dan menghasilkan *output* yang tepat, jika prediksi model mengandung banyak *error*/kesalahan maka *loss function* akan menghasilkan angka yang lebih tinggi. Begitu sebaliknya jika model sudah cukup bagus, maka *loss function* akan memberikan nilai yang lebih rendah. Gambar 4.3 (a) *dataset* 50 *loss function* mencapai 35% dengan *test loss* mencapai 33%. Selisih antara *loss function* dengan *test loss* pada *dataset* 50 yakni 2%. Gambar 4.3 (b) *dataset* 100 *loss function* mencapai 18% dengan *test loss* sebesar 21%. Sedangkan selisih antara *loss function* dengan *test loss* pada *dataset* 100 yakni 3%. Pada Gambar 4.3 (c) *dataset* 150 *loss function* mencapai 13% dengan *test loss* mencapai 17%. Selisih antara *loss function* dengan *test loss* pada *dataset* 150 yakni 4%. Gambar 4.3 (d) *dataset* 200 *loss function* mencapai 9% dengan *test loss* sebesar 15%. Sedangkan selisih antara *loss function* dengan *test loss* pada *dataset* 200 yakni 6%. Gambar 4.3 (e) *dataset* 250 *loss function* mencapai 7% dengan *test loss* sebesar 9%. Selisih antara *loss function* dengan *test loss* pada *dataset* 250 sebesar 2%. Gambar 4.3 (f) *dataset* 300 *loss function* mencapai 6% dengan *test loss* mencapai 9%. Sedangkan selisish antara *loss function* dengan *test loss* pada *dataset* 300 yakni 3%.

4.4. Analisis Model *CNN Learning*

Pada pembahasan ini analisis dari masing-masing penelitian ini dilakukan menggunakan *teachable machine*. Parameter yang digunakan sebagai analisis model ini ialah *F1-Score, accuracy, precision,* dan *recall* dari tiap-tiap *dataset,* selain itu terdapat juga parameter lainnya seperti *class* yang dibagi menjadi empat yakni *coffea, Immature Defect, Sour Defect,* dan *Black Defect.*

4.4.1. Analisis *Object Classification Class* 1 *Coffea*

Pada penelitian ini menggunakan metode CNN dengan klasifikasi gambar biji kopi. Hasil penelitian akan ditunjukkan menggunakan *confussion matrix* per objek dari klasifikasi gambarnya. Penelitian ini mengacu pada parameter dari *object classification class 1 (Coffea),* oleh sebab itu parameter hasil dari *confussion matrix*

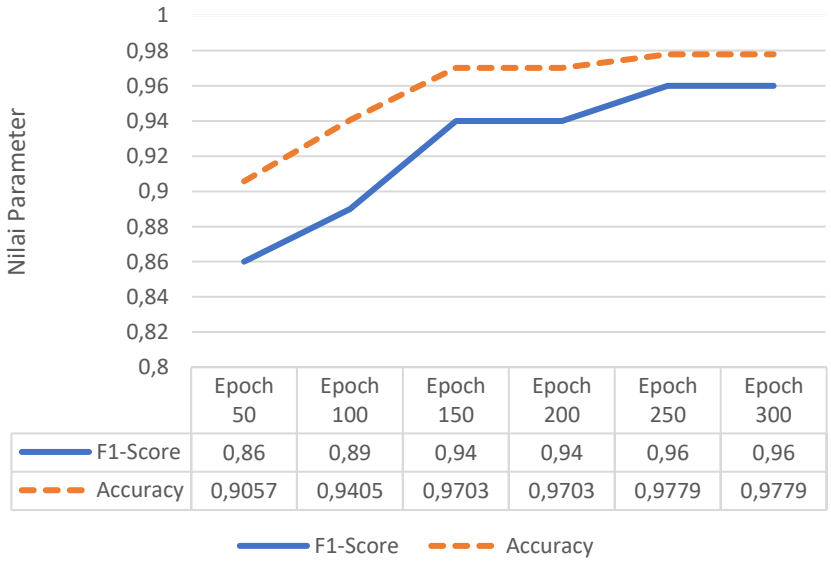
didapatkan dari hasil *training* pada *web tools teachable machine* Berdasarkan Tabel 4.1 di bawah ini.

Tabel 4.1 *Object Classification Class* 1 *Coffea*

Epoch	***Accuracy* (%)**	***Precision***	***Recall***	***F1-Score***
50	92,57	0,85	0,87	0,86
100	94,05	0,92	0,87	0,89
150	97,03	0,92	0,97	0,94
200	97,03	0,92	0,97	0,94
250	97,79	0,94	0,97	0,96
300	97,79	0,94	0,97	0,96
$\bar{X}$	**96,04**	**0,915**	**0,936**	**0,952**

Berdasarkan Tabel 4.1 merupakan hasil pengukuran dari parameter *object classification coffea* dengan menggunakan metode CNN. Berdasarkan tabel 4.1 tersebut dapat diketahui bahwa nilai rata-rata dari akurasi yang didapatkan pada model bagian *sample coffea* sebesar 96,04%

Evaluasi *F1-score* nilai rata-rata nya sebesar 0,952 menunjukan standar performa yang cukup tinggi. Nilai rata-rata *precision* dan *recall* yang merupakan variabel yang berpengaruh terhadap F*1-score* itu sendiri sebesar 0,915 dan 0,936. Berdasarkan data Berdasarkan Tabel 4.1 dapat ditentukan grafik dari perbandingan sebagai.



Gambar 4.4 Performa Model pada *Object Classification Class* 1 *Coffea*

Gambar 4.4 merupakan grafik performa dengan diperlihatkan *f1-score* dan juga *accuracy.* Berdasarkan Gambar 4.4 ialah hasil dari performa model, dapat disimpulkan bahwa besar nilai *f1-score* akan sama dengan *accuracy*. Pada *dataset* ke 250 dan ke 300 memiliki *accuracy* dan *f1-score* tertinggi.

Berdasarkan keseluruhan model yang didapatkan memiliki rata-rata *accuracy* sebesar 0,9404, nilai rata-rata *precision* sebesar 0,915, nilai rata-rata *recall* sebesar 0,936, dan nilai rata-rata *f1-score* sebesar 0,952, dari nilai rata-rata yang didapatkan pada penelitian ini, disimpulkan bahwa model yang diperoleh menggunakan *web-tools teachable machnine* memiliki hasil *learning* dan *prediction* yang cukup baik, serta memiliki kehandalan dalam mendeteksi objek menggunakan kamera android, namun dalam proses pengambilan *dataset*-nya terdapat *noise* cahaya yang masuk tetapi bisa digunakan dalam mengenali objek.

4.4.2. Analisis *Object Classification Class* 2 *Immature Defect*

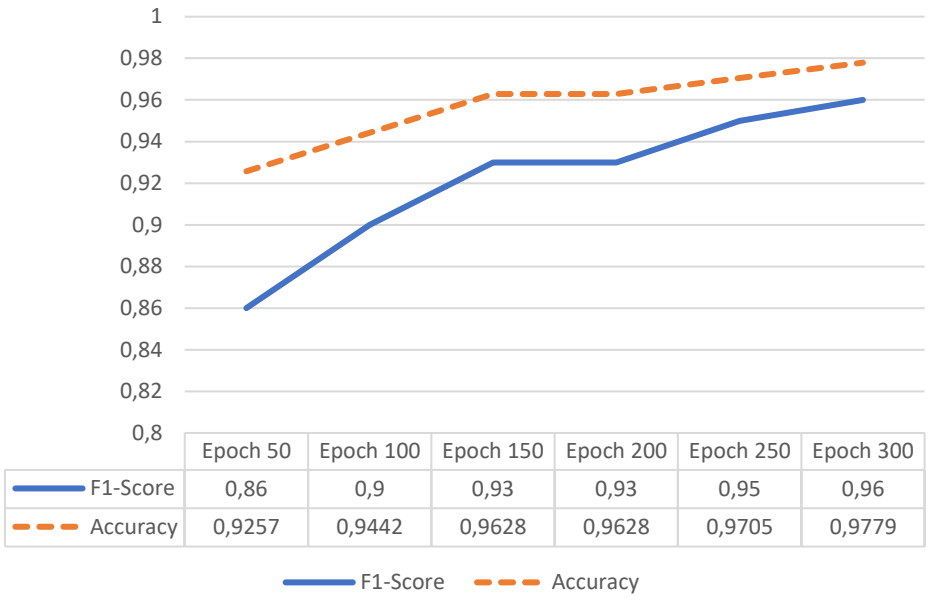
Pada analisis selanjutnya yang mengacu pada parameter dari *object classification class 2 immature defect*. Pada pembahasan ini menggunakan parameter hasil dari *confussion matrix* yang didapatkan dari hasil *training* pada *web tools teachable machine* Berdasarkan Tabel 4.2 di bawah ini.

Tabel 4.2 *Object Classification Class* 2 *Immature Defect*

Epoch	***Accuracy***	***Precision***	**Recall**	***F1-score***
50	92,57%	0,88	0,85	0,86
100	94,42%	0,88	0,91	0,90
150	96,28%	0,97	0,90	0,93
200	96,28%	0,97	0,90	0,93
250	97,05%	0,97	0,92	0,95
300	97,79%	0,97	0,97	0,96
$\bar{X}$	**95,73%**	**0,94**	**0,91**	**0,92**

Berdasarkan Tabel 4.2 merupakan hasil pengukuran dari parameter *object classification Immature Defect* dengan menggunakan metode CNN. Berdasarkan Tabel 4.2 tersebut dapat diketahui bahwa nilai rata-rata dari akurasi yang didapatkan pada sampel *Immature Defect* sebesar 95,73%, sedangkan evaluasi untuk *f1-score* nilai rata-rata nya sebesar 0,921, untuk nilai rata-rata *precision* dan *recall* yang merupakan nilai harmonik dari *f1-score* itu sendiri sebesar 0,94 dan

0,908. Berdasarkan data Berdasarkan Tabel 4.2 dapat ditentukan grafik dari perbandingan yang ditunjukan pada Gambar 4.5 berikut.



	Epoch 50	Epoch 100	Epoch 150	Epoch 200	Epoch 250	Epoch 300
F1-Score	0,86	0,9	0,93	0,93	0,95	0,96
Accuracy	0,9257	0,9442	0,9628	0,9628	0,9705	0,9779

Gambar 4.5 Performa Model pada *Object Classification Class* 2 *Immature Defect*

Pada Gambar 4.5 merupakan grafik performa dengan perbandingan *f1-score* dan juga *accuracy*. Berdasarkan Gambar 4.5 hasil dari performa model dapat disimpulkan bahwa besar nilai *f1-score* akan sama dengan *accuracy*. Titik tertinggi pada nilai *f1-score* dan *accuracy* terdapat pada *dataset* 250 dan 300. Dari keseluruhan model yang didapatkan pada objek ini memiliki rata-rata *accuracy* sebesar 0,9573, nilai rata-rata *precision* sebesar 0,94, nilai rata-rata recall sebesar 0,908, dan nilai rata-rata *f1-score* sebesar 0,921.

Berdasarkan nilai rata-rata yang didapatkan pada percobaan ini dapat disimpulkan bahwa model yang diperoleh memiliki hasil *learning* dan *prediction* yang cukup baik. Serta memiliki keandalan dalam mendeteksi objek menggunakan kamera android, namun dalam proses pengambilan *dataset* nya terdapat noise cahaya yang masuk tetapi bisa digunakan dalam mengenali objek.

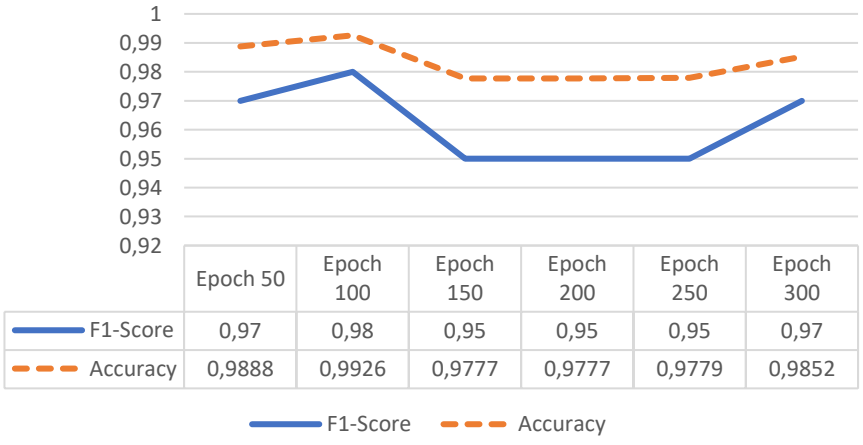
4.4.3. Analisis *Object Classification Class* 3 *Sour Defect*

Pada penelitian selanjutnya yang mengacu pada parameter dari *object classification class 3 (Sour Defect)*, pada parameter ini hampir sama dengan analisis sebelumnya yakni membahas parameter hasil dari *confussion matrix* didapatkan dari hasil *training* pada *web tools teachable machine* Berdasarkan Tabel 4.3 dibawah ini.

Epoch	*Accuracy* (%)	*Precision*	Recall	*F1-score*
50	98,88	0,98	0,97	0,97
100	99,26	0,98	0,98	0,98
150	97,77	0,93	0,96	0,95
200	97,77	0,93	0,96	0,95
250	97,79	0,93	0,96	0,95
300	98,52	0,97	0,97	0,97
$\bar{X}$	**98,33**	**0,95**	**0,96**	**0,96**

Tabel 4.3 *Object Classification Class* 3 *Sour Defect*

Berdasarkan Tabel 4.3 merupakan hasil dari pengukuran dari parameter *object classification* pada class 3 dengan metode CNN. Berdasarakan tabel tersebut dapat diketahui bahwa nilai rata-rata dari akurasi yang didapatkan pada model bagian sample *Sour Defect* sebesar 98,33%, sedangkan evaluasi untuk *f1-score* nilai rata-rata nya sebesar 0,961, untuk nilai rata-rata *precision* dan recall ialah sebesar 0,953 dan 0,966. Berdasarkan data Berdasarkan Tabel 4.3 dapat ditentukan grafik dari perbandingan pada Gambar 4.6.



Gambar 4.6 Performa Model pada *Object Classification Class* 3 *Sour Defect*

Gambar 4.6 merupakan grafik performa dengan perbandingan *f1-score* dan juga *accuracy* pada *object classification class 3*. Berdasarkan gambar di atas ilah hasil dari performa model, dapat disimpulkan bahwa besar nilai *f1-score* akan sama dengan nilai *accuracy*. Titik tertinggi pada nilai *f1-score* dan *accuracy* terdapat pada *dataset* 250 dan 300.

Berdasarkan keseluruhan model yang didapatkan pada objek ini memiliki rata-rata *accuracy* sebesar 0,9833, dan nilai rata-rata *f1-score* sebesar 0,961. Nilai yang didapatkan pada penelitian ini dapat disimpulkan bahwa model yang diperoleh memiliki hasil *learning* dan *prediction* yang cukup baik. Kehandalan dalam mendeteksi objek menggunakan kamera Android memiliki reliabilitas yang cukup, tetapi faktor pencahayaan dapat menjadi penurunan kehandalan algoritma.

4.4.4. Analisis *Object Classification Class* 4 *Black Defect*

Pada penelitian selanjutnya yang mengacu pada parameter dari *object classification class 4 (Black Defect)*, pada parameter ini hampir sama dengan analisa sebelumnya yakni membahas parameter hasil dari confussion matrix didapatkan dari hasil *training* pada *web tools teachable machine* Berdasarkan Tabel 4.4 dibawah ini.

Tabel 4.4 *Object Classification Class* 4 (*Black Defect*)

Epoch	***Accuracy***	***Precision***	***Recall***	***F1-score***
50	98,88%	0,97	0,99	0,98
100	98,88%	0,99	0,99	0,98
150	98,51%	0,97	0,97	0,97
200	98,51%	0,97	0,97	0,97
250	98,52%	0,97	0,97	0,97
300	98,52%	0,97	0,97	0,97
$\bar{X}$	98,63%	0,973	0,976	0,973

Berdasarkan Tabel 4.4 merupakan hasil dari pengukuran dari parameter object classification pada class 4 dengan metode CNN. Berdasarkan Tabel tersebut dapat diketahui bahwa nilai rata-rata dari akurasi yang didapatkan pada model bagian sample black coffea sebesar 98,63%, sadangkan evaluasi untuk *f1-score* nilai rata-rata nya sebesar 0,973. untuk nilai rata-rata *precision* dan recall ialah sebesar 0,973 dan 0,976. Berdasarkan data Berdasarkan Tabel 4.4 dapat ditentukan grafik dari perbandingan pada Gambar 4.7.

Gambar 4.7 Performa Model pada *Object Classification Class* 4 *Black Defect*

Gambar 4.7 merupakan grafik performa dengan membandingkan *f1-score* dan juga *accuracy*. Berdasarkan gambar di atas ialah hasil dari performa model, dapat disimpulkan bahwa besar nilai *f1-score* akan sama dengan *accuracy*. Titik tertinggi pada nilai *f1-score* dan *accuracy* terdapat pada *dataset* 205 dan 300.

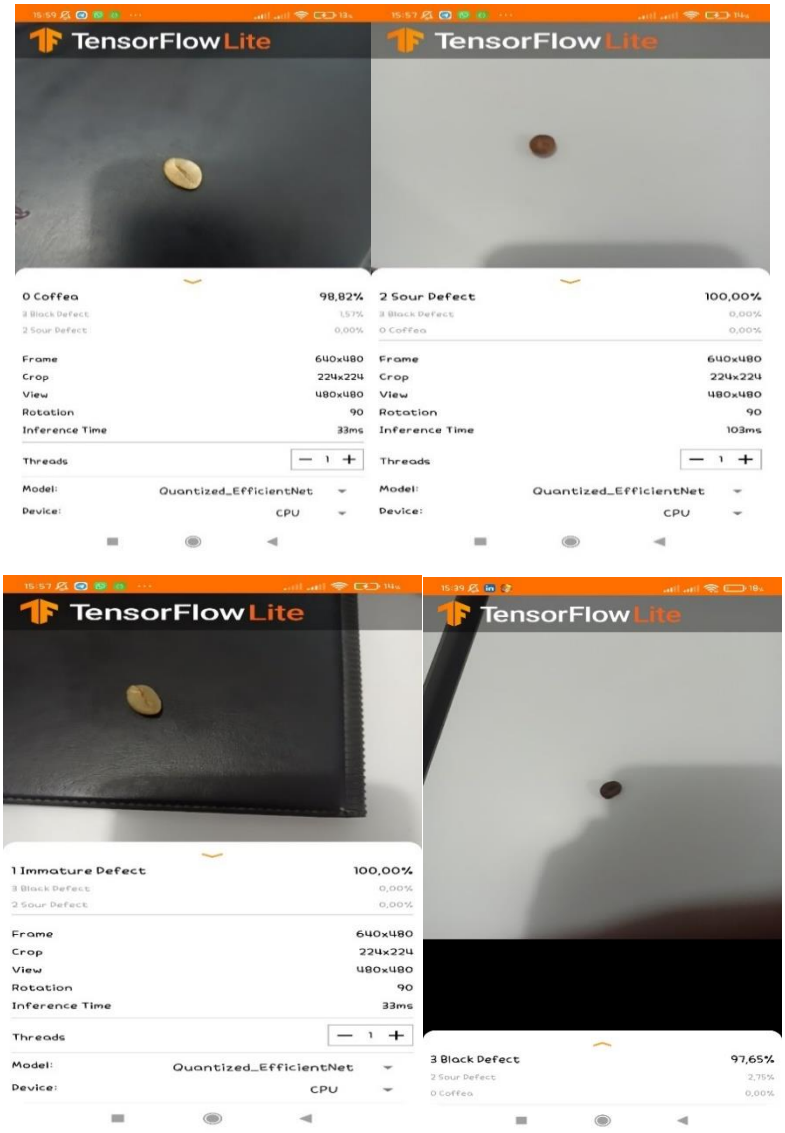
Berdasarkan keseluruhan model yang didapatkan pada objek ini memiliki rata-rata *accuracy* sebesar 0,9863, nilai rata-rata *precision* sebesar 0,973, nilai rata-rata *recall* sebesar 0,976, dan nilai rata-rata *f1-score* sebesar 0,921. Berdasarkan nilai rata-rata yang didapatkan pada penelitian ini ialah dapat disimpulkan bahwa model yang diperoleh memiliki hasil *learning* dan *prediction* yang cukup baik.

4.5. Hasil dan Pengujian Implementasi Pada Android

Pengujian ini bertujuan untuk melihat performa dan akurasi yang diperoleh setelah model diimplementasikan pada perangkat Android. Dalam hal ini terdapat empat parameter yang akan dibahas antara lain uji akurasi aplikasi terhadap model dan yang kedua uji coba performa aplikasi.

4.5.1. Uji Coba Akurasi Terhadap Model

Uji coba ini dilakukan untuk mengetahui tingkat akurasi dari aplikasi pengklasifikasian pada biji kopi. Pengujian ini dilakukan secara *real-time* dengan menggunakan citra pada biji kopi yang sebelumnya sudah *di-training*. *Dataset* model yang digunakan pada uji coba ini menggunakan model dengan *dataset* 300. Pada gambar di bawah ini adalah hasil dari uji coba akurasi model menggunakan aplikasi Tensorflow *Lite Image Classification Example* dengan *extension file* nya model.TFlite yang sebelumnya sudah dibuat.



Gambar 4.8 Hasil Tingkat Akurasi Pada Aplikasi

Gambar 4.8 di atas merupakan hasil uji coba aplikasi yang dibuat, pada aplikasi ini merupakan aplikasi yang digunakan untuk mengetahui nilai cacat pada biji kopi. Penelitian ini terdapat 4 *Class* yakni *coffea* untuk kualitas biji yang baik, *immature defect* untuk cacat tingkat satu, *sour defect* untuk cacat tingkat dua, dan yang terakhir *black defect* untuk cacat tingkat tiga.

Berdasarkan keempat *class* ini dapat ditentukan dali nilai presentase nya seperti gambar di atas, yang merupakan *confidence score* atau nilai keyakinan pada aplikasi dalam mengkategorikan gambar. Pada aplikasi ini *confidence score* total mencapai 100%. Di bawah ini merupakan Tabel 4.5 hasi dari uji coba terhadap dua puluh citra biji kopi dengan masing-masing kelas sebanyak lima biji *sample.*

Tabel 4.5 Tingkat Akurasi Pada Aplikasi

Jenis Nilai Cacat	Jumlah Sample	Tingkat Akurasi (%)	Rata-Rata Nilai Keyakinan (%)
Coffea	5	100	96
Immature Defect	5	100	97
Sour Defect	5	100	95
Black Defect	5	100	97

Berdasarkan Tabel 4.5 di atas dapat diketahui bahwa aplikasi dapat mengklasifikasi pada biji kopi dengan sangat baik dengan tingkat akurasi yang didapat mencapai sempurna dan dengan tingkat keyakinan di atas 90%, namun hal ini dipengaruhi oleh pencahayaan yang sesuai dan juga *background* yang sesuai dengan sample sebelumnya. Jika hal tersebut tidak sesuai, maka hasil dari pengambilan nilai keyakinannya tidak akan mencapai 90%, karena dipengaruhi faktor pencahayaan dan juga background pengambilan gambarnya.

4.5.2. Pengaruh Cahaya Terhadap Pengambilan Objek

Pada pengujian ini pengaturan cahaya berpengaruh terhadap hasil dari pengambilan objek gambar, sehingga dalam hal ini penting sekali untuk mengatur intensitas pencahayaan agar kemampuan identifikasi objek, semakin rendah tingkat pencahayaan maka semakin sulit bagi sistem untuk mengidentifikasi objek gambar pada biji kopi dan begitu juga sebaliknya. Berikut ini Tabel 4.6 yang merupakan hasil dari pengambilan objek gambar berdasarkan instensitas pencahayaan.

Tabel 4.6 Pengambilan Objek Berdasarkan Intensitas Cahaya dengan Jarak 20 cm

Percobaan	Durasi (detik)	Jarak (cm)	Intensitas (Flux)	Kemampuan Identifikasi (%)
1	15	20	200	99
2	20	20	200	98
3	25	20	200	99
4	30	20	200	98
5	35	20	200	99

Berdasarkan Tabel 4.6 di atas merupakan hasil pengukuran akurasi berdasarkan intensitas cahaya dengan jarak 20 cm. Berdasarkan data tersebut dapat disimpulkan semakin besar intensitas cahaya yang diperoleh semakin besar performa mendeteksi objek. Parameter lain yang dapat mempengaruhi aplikasi dalam mendeteksi objek yaitu *noise* pada objek yang disebabkan oleh besarnya intensitas cahaya, namun itu bisa diatasi dengan mengatur jarak antara pencahayaan dengan objek yang akan di ambil. Berdasarkan Tabel 4.7 merupakan hasil pengukuran dengan variasi jarak sejauh 30 cm.

Tabel 4.7 Pengambilan Objek Berdasarkan Intensitas Cahaya dengan Jarak 30 cm

Percobaan	Durasi (detik)	Jarak (cm)	Intensitas (Flux)	Kemampuan Identifikasi (%)
1	15	30	197	96
2	20	30	197	96
3	25	30	197	97
4	30	30	197	97
5	35	30	197	97

Berdasarkan Tabel 4.7 di atas merupakan hasil pengukuran akurasi berdasarkan intensitas cahaya dengan jarak 30 cm. Berdasarkan Tabel tersebut terdapat lima kali percobaan yang dilakukan. Berdasarkan data tersebut dapat disimpulkan semakin besar intensitas cahaya yang diperoleh semakin besar aplikasi mendeteksi objek. Parameter lain yang dapat mempengaruhi aplikasi dalam mendeteksi objek yaitu *noise* pada objek yang disebabkan oleh besar nya intensitas cahaya, namun hal tersebut dapat diatasi dengan mengatur jarak antara pencahayaan dengan objek yang akan diambil. Berdasarkan Tabel 4.8 merupakan hasil dari pengambilan data pada jarak 40 cm.

Tabel 4.8 Pengambilan Objek Berdasarkan Intensitas Cahaya dengan Jarak 40 cm

Percobaan	Durasi (detik)	Jarak (cm)	Intensitas (Flux)	Kemampuan Identifikasi (%)
1	15	40	134	97
2	20	40	133	97
3	25	40	134	97
4	30	40	134	97
5	35	40	134	96

Berdasarkan Tabel 4.8 di atas merupakan hasil pengukuran akurasi berdasarkan intensitas cahaya dengan jarak 40 cm. Berdasarkan Tabel tersebut terdapat lima kali percobaan yang dilakukan. Berdasarkan data ini dapat disimpulkan semakin besar intensitas cahaya yang diperoleh semakin besar aplikasi mendeteksi objek.

Parameter yang dapat mempengaruhi aplikasi dalam mendeteksi objek yaitu *noise* pada objek yang disebabkan oleh besar nya intensitas cahaya, namun itu bisa diatasi dengan mengatur jarak antara pencahayaan dengan objek yang akan di ambil. Berdasarkan Tabel 4.9 merupakan hasil pengambilan data pada jarak 60 cm.

Tabel 4.9 Pengambilan Objek Berdasarkan Intensitas Cahaya dengan Jarak 60 cm

Percobaan	Durasi (detik)	Jarak (cm)	Intensitas (Flux)	Kemampuan Identifikasi (%)
1	15	60	112	99
2	20	60	113	98
3	25	60	112	99
4	30	60	112	98
5	35	60	112	99

Berdasarkan Tabel 4.9 di atas merupakan hasil pengukuran akurasi berdasarkan intensitas cahaya dengan jarak 60 cm. Berdasarkan Tabel tersebut terdapat lima kali percobaan yang dilakukan, berdasarkan data ini dapat disimpulkan semakin besar intensitas cahaya yang diperoleh semakin besar aplikasi mendeteksi objek.

Parameter lain yang dapat mempengaruhi aplikasi dalam mendeteksi objek yaitu noise pada objek yang disebabkan oleh besar nya intensitas cahaya, namun itu bisa di atasi dengan mengatur jarak antara pencahayaan dengan objek yang akan di ambil.

4.5.3. Uji Coba Performa Aplikasi

Pada pengujian performa, aplikasi Tensorflow *lite*, memiliki fitur-fitur yang dapat digunakan untuk mengubah jenis model TFlite beserta optimisasi yang ada pada Tensorflow *lite*. Disini terdapat dua jenis model TFLite yaitu *Quantized* dan *Unquant*.

Model quantized merupakan model TFLite yang sudah melakukan proses *quantization*. *Quantization* merupakan proses optimisasi yang dimiliki oleh TFLite yang memiliki fungsi mengurangi ukuran dan mempermudah proses komputasi dan mempertahankan akurasinya.

Perbedaannya dengan model *Unquant* yakni dari segi kinerja jenis datanya untuk *quantization* memiliki jenis data berjenis *Integer* 8 bit, sedangkan untuk *Unquant* berjenis data *float* 32, untuk penelitian ini digunakan jenis data *quantized*, hal ini dikarenakan kecapatan pemrosesan datanya lebih cepat. Namun memiliki kekurangan ketelitian pada model sehingga akurasi dari *quantized* lebih kecil dibandingkan denga *unquant.* Perbandingan ukuran *file* pada Gambar 4.4 di bawah.

Gambar 4.9 Ukuran TFlite *file* Model

Pada Gambar 4.9 merupakan perbandingan ukuran dari model yang telah di *generate* dari Teachable Machine. Ukuran dari model *unquant* memiliki ukuran yang lebih besar dengan tipe data float 32.

4.1.1. Uji Coba Performa Aplikasi

Pada pengujian ini dilakukan untuk membandingkan model TFLite *Floating Point* dan juga dengan objek gambar yang sama yang diperoleh dari beberapa situs secara *real time*, performa pada aplikasi dapat dilihat melalui fitur *Interference Time* atau kecepatan aplikasi dalam mengklasifikasikan gambar. Berdasarkan Tabel 4.10 merupakan hasil uji coba performa aplikasi dengan menggunakan NNAPI, GPU dan CPU.

Tabel 4.10 Pengaruh CPU Terhadap Performa Aplikasi

Percobaan	***Interference Time* (ms)**	***Confidence Score* (%)**
1	35	98,36
2	32	97,10
3	30	90,98
4	30	90,40
5	30	90,50

Berdasarkan Tabel 4.10 di atas merupakan hasil dari percobaan untuk mengetahui performa aplikasi pada CPU, dimana percobaan dilakukan sebanyak tiga kali untuk mendapatkan perbandingan yang sesuai. Titik terbesar pada interference time mencapai 35 ms dengan *score* 98,36%, untuk titik terendah sebesar 30 ms dengan *score* 90,40%.

Pada percobaan ini CPU berperan penting dalam proses pengklasifikasian, selain itu cpu merupakan *hardware* dasar dalam mengoptimalkan aplikasi. Sehingga proses pengklasifikasian dapat berjalan sempurna. Berdasarkan Tabel 4.11 merupakan hasil pengaruh NNAPI terhadap performa aplikasi.

Tabel 4.11 Pengaruh NNAPI Terhadap Performa Aplikasi

Percobaan	**Interference Time (ms)**	**Confidence Score**
1	25	85,36%
2	26	86,10%
3	25	87,98%
4	25	87,40%
5	25	87,50%

Berdasarkan Tabel 4.11 di atas merupakan hasil dari percobaan untuk mengetahui performa aplikasi pada NNAP. Nnnpi merupakan *software* yang dirancang untuk menjalankan operasi komputasi intensif bagi *machine learning* pada perangkat Android. Pada percobaan ini terdapat lima kali percobaan agar dapat mengetahui perbandingan yang terjadi.

Dapat disimpulkan bahwa *interference time* pada NNAPI menghasilkan kecepatan yang paling lambat dibandingkan dengan CPU, hal ini disebabkan CPU merupakan *hardware* dasar pada perangkat aplikasi pengklasifikasian. Selain itu ada beberapa faktor yang dapat membuat performa NNAPI lebih lambat dibandingkan dengan CPU salah satu nya ialah karena *driver* NNAPI pada

perangkat aplikasi tidak tersedia, sehingga salah satu nya menjadi penyebab proses pengklasifikasian pada aplikasi berjalan lambat.

Tabel 4.12 pengaruh GPU Terhadap Performa Aplikasi

Percobaan	Interference Time (ms)	Confidence Score
1	45	98,36%
2	45	97,30%
3	47	92,88%
4	47	92,30%
5	45	92,20%

Berdasarkan Tabel 4.12 di atas merupakan hasil dari percobaan untuk mengetahui performa aplikasi pada GPU. Pada percobaan terdapat lima kali percobaan agar dapat mengetahui perbandingan yang terjadi. Dapat disimpulkan bahwa performa GPU menghasilkan akselarasi yang hampir sama dengan CPU. Perbedaan nya dalam hal ini dikarenakan GPU sendiri dikhususkan untuk memproses grafis sedangkan CPU memproses berbagai intruksi umum sementara.

Pada percobaan ini juga terdapat beberapa kendala dalam melakukan percobaan performa aplikasi menggunakan *device* GPU. Hal tersebut disebabkan dalam pemporsesannya membutuhkan spesifikasi perangkat yang sangat *powerfull*, karena pada *device* GPU mengandalakan performa grafis dalam proses pengklasifikasiannya.

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan penelitian yang telah dilakukan yaitu Implementasi Tensorflow Lite Untuk Mengetahui Jenis Cacat Pada Biji Kopi Robusta Berbasis Android, maka dapat disimpulkan, sebagai berikut:

1. Hasil dari pengklasifikasian pada 4 *sample* biji kopi Gunung Karang menggunakan sistem operasi Android dengan Tensorflow Lite sebagai *freamwork*, didapatkan hasil deteksi objek jenis biji kopi, yaitu *premium*, *immature defect*, *sour defect*, dan *black defect.*
2. Perancangan sistem klasifikasi berbasis Android dengan Tensorflow Lite, didapatkan hasil tingkat akurasi penentuan nilai cacat pada biji kopi sebesar 97%.
3. Pembuatan klasifikasi menggunakan Teachable Mechine sebagai *training dataset* dikarenakan mudah dalam proses *input* data dan juga prosesnya lebih cepat dibandingkan proses klasifikasi yang lain, sedangkan Tensorflow Lite sebagai *freamwork* digunakan sebagai aplikasi yang terhubung dalam perangkat *smartphone*, sehingga menjadi solusi aplikatif yang *portable* bagi produsen dalam penentuan *quality control* produk biji kopi.

5.2. Saran

Adapun saran yang dapat diberikan dari hasil penelitian yang telah dilakukan antara lain sebagai berikut.

1. Menambahkan fitur *input* gambar pada aplikasi agar data semakin variatif.
2. Menggunakan banyak sampel gambar dan jumlah *dataset* yang banyak serta menggunakan *learning rate* yang rendah saat proses *training* agar mendapatkan model *neural network* yang akurat pada pengaplikasian Tensorflow.
3. Metode CNN dapat diganti dengan metode lain seperti *artificial neural network* ataupun jenis metode lainnya yang dimungkinkan untuk mengetahui keakuratan dari *object detection*.

DAFTAR PUSTAKA

[1] Parnadi, F., Loisa, R. *Analisis Daya Saing Ekspor Kopi Indonesia Di Pasar Internasional*. Jurnal Manajemen Bisnis Dan Kewirausahaan. 2018. Vol. 2. No. 2, pp. 52-61.

[2] Budihardjo, K., W. Mutiara Fahmi. *Strategi Peningkatan Produksi Kopi Robusta (Coffea L.) Di Desa Pentingsari, Kecamatan Cangkringan, Kabupaten Sleman, Daerah Istimewah Yogyagkarta*. Jurnal Ilmiah Mahasiswa AGROINFO GALUH. 2020. Vol. 7. No. 2, pp.373-379.

[3] Sari, M.N., T., Suhartati, Husniati. *Analisis Senyawa Asam Klorogenat Dalam Biji Kopi Robusta (Coffea Canephora) Menggunakan HPLC*. Analytical and Environmental Chemistry. 2019. Vol. 4. No. 2, pp. 87-93.

[4] As'ad, M.H., J.. Mulyo Aji. *Factors Affecting The Preference of Modern Coffe Shop Consumers in Bondowoso*. Journal of Social and Agricultural Economics. 2020. Vol. 13. No. 2, pp. 182-199.

[5] Arboleda, E.R., Arnel, C.F., Ruji, P.M. *Classification of Coffee Bean Species Using Image Processing, Artificial Neural Network and K Nearest Neighbors*. 2018 IEEE International Conference on Innovative Research and Development. 2018. Thailand.

[6] Manuel, M.N., Adenilton, C.D., Gisele, S.L.,Livia, P.D. *One-class Classification of Special Agroforestry Brazilian coffee using NIR spectrometry and chemometric tools*. Food Chemistry. 2022. Vol 366, pp. 1-6.

[7] Baqueta, M.R., Aline, C. Paulo, H.M., Patricia, V. *Multivariate classification for the direct determination of cup profile in coffee blends via handheld near-infrared spectroscopy*. Talanta. Vol. 222, pp. 1-8.

[8] Putra, W.Y. *Klasifikasi Citra Menggunakan Convolutional Neural Network (CNN) Pada Caltech 101*. Jurnal Teknik ITS. 2016. Vol 5. No. 1.

[9] Kurniawan, W.M., Khafizh, H. *Penentuan Kualitas Biji Kopi Arabika Dengan Menggunakan Analytical Hierarchy Process (Studi Kasus Pada Perkebunan Kopi Lereng Gunung Kelir Jambu Semarang)*. Jurnal Teknik

Industri Mesin Elektro dan Ilmu Komputer. 2017. Vol. 8. No. 2, pp. 519-528.

[10] Nurfalah, A.A., Surti, Z. Mohamad, B.T. *Pengaruh Kualitas Produk dan Harga Terhadap Kepuasan Konsumen di Kedai Kopi Mustafa85 Pandeglang Banten (Studi Kasus Kedai Kopi Mustafa85 di Pandeglang Banten)*. Jurnal Bina Bangsa Ekonomika. 2020. Vol. 13. No. 02, pp. 313–318.

[11] Mahmuda, S.,Kusrini, Mei. P.K. *Identifikasi Mutu Biji Kopi Arabika Berdasarkan Cacat*. Jurnal Teknologi Informasi dan Komunikasi. 2020. Vol. 10, No. 1, pp. 27–35.

[12] Febriani, A.I., Raden, S. *Klasifikasi Mutu Biji Kopi Robusta Berdasarkan Ciri Warna Dan Tekstur Dengan Metode Backpropagation Neural Network.* Universitas Gadjah Mada Tugas Akhir. 2018.

[13] Vasileios, L., Spyridon, M., Konstantina, K. Sotirios, X. Dimitrios, S., Kiamal, P. *A* Tensorflow *Extension Framework for Optimized Generation of Hardware CNN Inference Engines*. Technologies. 2020. Vol. 8. No. 6, pp. 1-15.

[14] Louis, M.S., Zahra, A. Leila, D., Suyog, G., Pete, W., Vijay, J.R., Ajay J. *Towards Deep Learning using* Tensorflow *Lite on RICS-V*. 2019. CARRV.

[15] Royani, D.N., Gunawan, A. *Implementasi Deep Learning berbasis* Tensorflow *untuk Pengenalan Sidik Jari*. Jurnal Teknik Elektro. 2018, Vol. 8, No 1.

[16] Alam, I.F., Muhammad, I.S., Adha, M.S. *Implementasi Deep Learning dengan Metode Convolutional Neural Network untuk Identifikasi Objek secara Real Time Berbasis Android*. 2020. semanTIK. Vol. 5. No. 2, pp. 12–26.

[17] Hendrawan, Y., Rohmatullah, B., Ilmi, F., Fauzy, R., Damayanti, R., Riza, D.F., Hermanto, B., Sandra. *AlexNet convolutional neural network to classify the types of Indonesian coffee beans*. IOP Conference Series: Earth and Environmental Science. 2021, Vol. 905. No. 1.

[18] Wisnudhanti, K., Feri, C. *Metode Convolutional Neural Network Dalam Klasifikasi Citra Tiga Tokoh Wayang Pandawa*. 2020. Jurnal Online

Mahasiswa. Vol. 7. No. 2, pp. 1–5.

[19] David, R., Jarke, D., Advait, J., Vijay, J.R., Nat, J., Jian, L. Nick, K., Ian, N., Meghna, N., Shlomi, R., Rocky, R., Tiezhen, W., Pete, W. Tensorflow *Lite Micro: Embedded Machine learning on TinyML Systems*. 2020. 4th MLSys Conference. California.

[20] Wikarsa, L., Angdresey, A., Ticoalu, S. *Detection of the Types of Consumable Saltwater Fish in the Coastal Area of Likupang Uses the Convolutional Neural Network Method*. 2022. Vol. 7. No. 2.

[21] Thohari, A.N., Galuh, B.H. *Implementasi Convolutional Neural Network untuk Klasifikasi Pembalap MotoGP Berbasis GPU*. 2018. CENTIVE. pp. 50–55.

[22] Sari, I.R. *Implementasi Convolutional Neural Networks (Cnn) Untuk Klasifikasi Citra Tomat Menggunakan Keras*. Tugas Akhir Universitas Muhammadiyah Semarang. 2021.

[23] Mardiyah, M.I. *Implementasi Deep Learning untuk Image Classification Menggunakan Algoritma Convolutional Neural Network (CNN) Pada Citra Kebun dan Sawah*. Tugas Akhir Universitas Islam Indonesia. 2020.

[24] Riad, R., Olivier, T., David, G., Neil, Z. *Learning strides in convolutional neural networks*. International Conference on Learning Representations. 2022.

[25] Hibatullah, A., Irfan, M. *Penerapan Metode Convolutional Neural Network Pada Pengenalan Pola Citra Sandi Rumput*. Library Unikom. 2019.

[26] Nugroho, A, P., Fenriana, I., Rudi, A. *Implementasi Deep Learning Menggunakan Convolutional Neural Network (CNN) Pada Ekspresi Manusia*. Jurnal Algor. 2020. Vol. 2. No. 1.

[27] Tejakumar, D., Mahardi, I-Hung, W., Kuang, L., Shinn, C. *Predicting Surface Roughness using Keras DNN Model*. IEEE Eurasia Conference on IOT, Communication and Engineering. 2020.

[28] Tseng, C., Su, L. *Design of digital differentiator using supervised learning on keras framework*. IEEE 8th Global Conference on Consumer Electronics. 2019.

[29] Skalicly, S., Joshua, M., Andrew, S., Matthew, F. *Hot&Spicy: Improving*

Productiviy with Python and HLS for FPGAS. IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines. 2018.

[30] Goldsborough, P. *A Tour of* Tensorflow. Arxhiv. 2016.

[31] Manor, E., Shlomo, G. *Custom Hardware Inference Accelerator for* Tensorflow *Lite for Microcontrollers*. IEEE Access. 2022. Vol. 10, pp. 73484-73493.

[32] Zeroural, A., Makhlouf, D., Mohamed, A., Atef, B. *Using a Fine-Tuning Method for a Deep Authentication in Mobile Cloud Computing Based on* Tensorflow *Lite Framework.* International Conference on Networking and Advanced Systems. 2019.

[33] Vidaklovic, M., Stefan, C., Ognjen, C., Ivan, K., Gordana, V. *One solution for execution of JavaScript in Java EE application servers*. Zooming Innovation in Consumer Technologies Conference. 2018.

[34] Ejiyi, C.J, Jianhua, D., Thomas, U.E., Adetunji, A. S., Makuachukwu, B.E., Chinoso, G. A. *Design and Development of Android Application for Educational Institutes*. Journal of Physics: Conference Series. 2021. Vol. 1769.

[35] Agustian, D., Pande, P.P., Padma, N.C., Putu, D.N. *Implementation of Machine Learning using Google's Teachable Machine Based on Android.* 3rd International Conference on Cybernethics and Intelligent System. 2021. Indonesia.

[36] Arboleda, E.R., Arnel, C.F., Ruji, P.M. *Classification of coffee bean species using image processing, artificial neural network and K nearest neighbors*. IEEE International Conference on Innovative Research and Development. 2018. Thailand.

[37] Gonzalez, J. D., Jormany, Q.R. *Use of Convolutional Neural Networks in Smartphones for the Identification of Oral Diseases Using a Small Dataset.* Revista Faculta de Ingenieria. 2021.

LAMPIRAN A *Confusion Matrix*

60	11	0	0
9	64	0	0
0	0	56	1
0	0	2	66

Confusion Matrix *Dataset* 50

65	6	0	0
9	64	0	0
0	0	56	1
1	0	1	66

Confusion Matrix *Dataset* 100

65	6	0	0
2	71	0	0
0	2	53	2
0	0	2	66

Confusion Matrix *Dataset* 200

67	4	0	0
2	71	0	0
0	0	55	2
0	0	2	66

Confusion Matrix *Dataset* 250

67	4	0	0
2	71	0	0
0	0	57	2
0	0	2	66

Confusion Matrix *Dataset* 300

LAMPIRAN B Listing Program *ClassifierQuantizedMobileNet*

```
package org.Tensorflow.lite.examples.classification.tflite;

import Android.app.Activity;
import java.io.IOException;
import
org.Tensorflow.lite.examples.classification.tflite.Classifie
r.Device;
import org.Tensorflow.lite.support.common.TensorOperator;
import org.Tensorflow.lite.support.common.ops.NormalizeOp;

/** This Tensorflow Lite classifier works with the quantized
MobileNet model. */
public class ClassifierQuantizedMobileNet extends Classifier
{

  /**
   * The quantized model does not require normalization, thus
set mean as 0.0f, and std as 1.0f to
   * bypass the normalization.
   */
  private static final float IMAGE_MEAN = 0.0f;

  private static final float IMAGE_STD = 1.0f;

  /** Quantized MobileNet requires additional dequantization
to the output probability. */
  private static final float PROBABILITY_MEAN = 0.0f;

  private static final float PROBABILITY_STD = 255.0f;
```

```
  /**
   * Initializes a {@code ClassifierQuantizedMobileNet}.
   *
   *                    @param                    activity
   */
  public ClassifierQuantizedMobileNet(Activity activity,
Device          device,          int          numThreads)
      throws                  IOException                  {
    super(activity,          device,          numThreads);
  }

  @Override
  protected          String          getModelPath()          {
    //    you    can    download    this    file    from
    // see build.gradle for where to obtain this file. It
should                        be                        auto
    //          downloaded          into          assets.
    return                                "model.tflite";
  }

  @Override
  protected          String          getLabelPath()          {
    return                                "labels.txt";
  }

  @Override
  protected  TensorOperator  getPreprocessNormalizeOp()  {
    return    new    NormalizeOp(IMAGE_MEAN,    IMAGE_STD);
  }

  @Override
```

```
  protected TensorOperator getPostprocessNormalizeOp() {
    return new NormalizeOp(PROBABILITY_MEAN, PROBABILITY_STD);
  }
}
```

LAMPIRAN C Listing Program *ClassifierFloatMobileNet*

```
package org.Tensorflow.lite.examples.classification.tflite;

import Android.app.Activity;
import java.io.IOException;
import
org.Tensorflow.lite.examples.classification.tflite.Classifie
r.Device;
import org.Tensorflow.lite.support.common.TensorOperator;
import org.Tensorflow.lite.support.common.ops.NormalizeOp;

/** This TensorflowLite classifier works with the float
MobileNet model. */
public class ClassifierFloatMobileNet extends Classifier {

  /** Float MobileNet requires additional normalization of
the used input. */
  private static final float IMAGE_MEAN = 127.5f;

  private static final float IMAGE_STD = 127.5f;

  /**
   * Float model does not need dequantization in the post-
processing. Setting mean and std as 0.0f
   * and 1.0f, repectively, to bypass the normalization.
   */
  private static final float PROBABILITY_MEAN = 0.0f;

  private static final float PROBABILITY_STD = 1.0f;
```

```
  /**
   * Initializes a {@code ClassifierFloatMobileNet}.

   *
   *                    @param                    activity
   */
  public ClassifierFloatMobileNet(Activity activity, Device
device,                  int                  numThreads)
      throws                IOException                {
    super(activity,          device,          numThreads);
  }

  @Override
  protected          String          getModelPath()          {
    //    you    can    download    this    file    from
    // see build.gradle for where to obtain this file. It
should                        be                        auto
    //          downloaded          into          assets.
    return                          "model_unquant.tflite";
  }

  @Override
  protected          String          getLabelPath()          {
    return                                  "labels.txt";
  }

  @Override
  protected   TensorOperator   getPreprocessNormalizeOp()   {
    return    new    NormalizeOp(IMAGE_MEAN,    IMAGE_STD);
  }
```

```
  @Override
  protected TensorOperator getPostprocessNormalizeOp() {
    return new NormalizeOp(PROBABILITY_MEAN, PROBABILITY_STD);
  }
}
```

LAMPIRAN D Listing Program Build.Gridl Android

```
apply plugin: 'com.Android.application'

Android {
    compileSdkVersion 28
    defaultConfig {
        applicationId
"org.Tensorflow.lite.examples.classification"
        minSdkVersion 21
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-
Android.txt'), 'proguard-rules.pro'
        }
    }
    aaptOptions {
        noCompress "tflite"
    }
    compileOptions {
        sourceCompatibility = '1.8'
        targetCompatibility = '1.8'
    }
    lintOptions {
        abortOnError false
```

```
    }
}

// Download default models; if you wish to use your own models
then
// place them in the "assets" directory and comment out this
line.

apply                                   from:'download.gradle'

dependencies                                                {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation      'Androidx.appcompat:appcompat:1.0.0'
    implementation
'Androidx.coordinatorlayout:coordinatorlayout:1.0.0'
    implementation
'com.google.Android.material:material:1.0.0'

    //Build     off     of     nightly     Tensorflow     Lite
    implementation('org.Tensorflow:Tensorflow-lite:0.0.0-
nightly')       {       changing       =       true       }
    implementation('org.Tensorflow:Tensorflow-lite-
gpu:0.0.0-nightly')     {     changing     =     true     }
    implementation('org.Tensorflow:Tensorflow-lite-
support:0.0.0-nightly')     {     changing     =     true     }
    //       Use       local       Tensorflow       library
    //    implementation    'org.Tensorflow:Tensorflow-lite-
local:0.0.0'

    AndroidTestImplementation
'Androidx.test.ext:junit:1.1.1'
    AndroidTestImplementation
```

```
'com.Android.support.test:rules:1.0.2'
    AndroidTestImplementation 'com.google.truth:truth:1.0.1'
}
```

LAMPIRAN E *Build Model Tensoflow Lite* Menggunakan *Web Tools Teachable machine*