BAB I

PENDAHULUAN

A. Latar Belakang

Kemajuan teknologi kecerdasan buatan, khususnya *Deep Learning*, telah mendorong pengembangan algoritma yang mampu menganalisis data visual secara akurat. Salah satu algoritma yang menonjol adalah *Convolutional Neural Network* (CNN). Menurut Hawari dkk. (2024), CNN dirancang untuk mengolah data gambar dua dimensi, mengekstraksi fitur penting, dan mengidentifikasi objek berdasarkan pola yang dipelajari dari dataset. CsNN banyak digunakan dalam berbagai bidang, termasuk pengenalan objek, klasifikasi citra, dan segmentasi gambar, karena kemampuan adaptasinya terhadap data visual yang kompleks.

Pada implementasinya, CNN sering dikembangkan menggunakan *TensorFlow*, sebuah pustaka *open-source* yang populer untuk *Deep Learning*. *TensorFlow* mendukung pemrosesan data dalam skala besar dan memungkinkan pengembangan aplikasi berbasis kecerdasan buatan yang dapat diterapkan pada berbagai *platform*, termasuk perangkat *mobile*. Penelitian oleh Hidayat dan Lusiana (2022) menunjukkan bahwa CNN, yang diimplementasikan menggunakan *TensorFlow*, dapat mengklasifikasikan jenis sayuran dengan tingkat akurasi hingga 86%. Dalam penelitian tersebut, *TensorFlow* memungkinkan integrasi algoritma CNN dengan aplikasi berbasis Android, sehingga memberikan solusi praktis dalam pengklasifikasian gambar melalui teknologi smartphone.

Di sisi lain, biji kakao *(Theobroma cacao L.)*, yang merupakan komoditas penting dalam industri cokelat, memiliki karakteristik warna yang berubah selama proses fermentasi dan pengeringan. Warna biji kakao menjadi salah satu indikator utama dalam menentukan tingkat kekeringan biji, yang berpengaruh langsung terhadap kualitas produk akhir. Menurut Rachmatullah, dkk (2021) pada jurnal "Karakteristik Biji Kakao Hasil Fermentasi dengan Ukuran Wadah Berbeda," warna biji kakao bervariasi dari putih berlendir pada tahap biji mentah, cokelat muda pada biji setengah kering, hingga cokelat tua pada biji kering, tergantung pada proses fermentasi dan pengeringan.

Dalam penelitian ini, algoritma CNN diterapkan untuk mendeteksi tingkat kekeringan pada biji kakao berbasis *machine learning*, dengan dukungan *TensorFlow* sebagai kerangka kerja utama. *TensorFlow* digunakan untuk melatih dan mengoptimalkan model CNN, serta mengintegrasikan model tersebut ke dalam aplikasi berbasis Android. Aplikasi ini dirancang untuk memanfaatkan kemampuan CNN dalam mengklasifikasikan warna biji kakao pada berbagai tingkat kekeringan. Output dari tingkat kekeringan yang dimaksud dalam penelitian ini adalah label kategori yaitu basah, setengah kering, dan kering, bukan berupa persentase.

Berdasarkan uraian diatas maka penulis ingin mengusulkan judul "**Rancang Bangun Aplikasi Pendeteksi Tingkat Kekeringan pada Biji Kakao Berbasis Android**".

B. Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah yang akan dibahas adalah sebagai berikut:

1. Bagaimana cara mengimplementasikan *Convolutional Neural Network* (CNN) untuk melakukan pelatihan model menggunakan dataset biji kakao dengan menggunakan bahasa *Python*?
2. Bagaimana cara mengklasifikasikan warna biji kakao menggunakan *Convolutional Neural Network* (CNN)?
3. Bagaimana cara mengintegrasikan model *TensorFlow Lite* (.tflite) hasil pelatihan ke dalam aplikasi android menggunakan bahasa *Kotlin*?
4. Bagaimana hasil pengujian deteksi aplikasi terhadap biji kakao?

C. Tujuan Penelitian

Berdasarkan rumusan masalah, maka tujuan penelitian ini adalah sebagai berikut:

1. Untuk mengetahui cara mengimplementasikan *Convolutional Neural Network* (CNN) untuk melakukan pelatihan model menggunakan dataset biji kakao dengan menggunakan bahasa *Python*.
2. Untuk mengetahui cara mengklasifikasikan warna biji kakao menggunakan *Convolutional Neural Network* (CNN).
3. Untuk mengetahui cara mengintegrasikan model *TensorFlow Lite* hasil pelatihan ke dalam aplikasi android menggunakan bahasa *Kotlin*.
4. Untuk mengetahui hasil pengujian deteksi aplikasi terhadap biji kakao.

D. Batasan Masalah

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

1. Deteksi tingkat kekeringan biji kakao pada aplikasi ini dilakukan melalui kamera perangkat menggunakan metode pengambilan gambar *(image capture)*. Proses deteksi tidak dilakukan secara *real-time* (video), sehingga pengguna harus mengambil gambar secara langsung pada saat itu. Aplikasi ini juga tidak mendukung deteksi dengan cara mengunggah gambar dari galeri atau sumber eksternal lainnya.
2. Sistem hanya mendeteksi tingkat kekeringan berdasarkan karakteristik warna visual yang ada dalam dataset model *TensorFlow*. Parameter lain seperti tekstur fisik atau tingkat kelembapan non-visual tidak menjadi bagian dari deteksi.
3. Aplikasi ini hanya dapat mendeteksi kelompok biji kakao yang termasuk dalam kategori: Basah, Setengah Kering, dan Kering secara terpisah.
4. Aplikasi ini tidak dapat mendeteksi biji kakao yang berada dalam kondisi gabungan, seperti Basah, Setengah Kering, dan Kering dalam satu kelompok atau gambar yang sama.
5. Output dari aplikasi ini hanya menampilkan tingkat kekeringan dalam kategori: Basah, Setengah Kering, dan Kering, tanpa memberikan hasil dalam bentuk persentase atau parameter lainnya.

E. Manfaat Penelitian

Besar harapan peneliti akan adanya manfaat dari penelitian ini untuk semua kalangan. Adapun manfaat penelitian adalah sebagai berikut:

1. Terhadap penulis, penelitian ini memberikan kesempatan bagi penulis dalam mengasah keterampilan teknis, seperti merancang, mengembangkan, dan mengimplementasikan algoritma *Convolutional Neural Network* (CNN) menggunakan *framework TensorFlow*. Penulis memperoleh pemahaman mendalam mengenai proses *image classification* untuk mendeteksi tingkat kekeringan biji kakao melalui pengembangan aplikasi berbasis Android.
2. Terhadap akademis, penelitian ini menjadi referensi bagi mahasiswa atau akademisi yang ingin mengeksplorasi penerapan teknologi CNN dan *TensorFlow* dalam pengembangan aplikasi berbasis *machine learning*. Selain itu, penelitian ini memperkaya literatur akademis di bidang informatika dengan studi kasus yang konkret tentang klasifikasi citra untuk mendukung proses agrikultur modern. Kajian ini juga dapat menjadi dasar bagi penelitian lanjutan dalam meningkatkan efisiensi dan akurasi teknologi serupa.
3. Terhadap teknologi, Penelitian ini berkontribusi pada kemajuan teknologi dengan memanfaatkan *TensorFlow* untuk membangun dan mengoptimalkan model CNN yang dapat diimplementasikan pada perangkat Android. *TensorFlow* mendukung proses pelatihan dan pengoptimalan model, sementara *TensorFlow Lite* memungkinkan integrasi model ke dalam perangkat *mobile*.

BAB II

TINJAUAN PUSTAKA

A. Kajian Teori

1. Kakao

Kakao merupakan tanaman perkebunan yang memiliki nama ilmiah *Theobroma cacao L*. Kakao, memiliki nama famili *Sterculiaceae*. Tanaman ini berasal dari Amerika Selatan yang saat ini banyak ditanam di berbagai kawasan tropika (Bulandari, 2019). Buah kakao memiliki bentuk oval dengan kulit yang tebal dan berwarna kuning hingga merah ketika matang. Di dalam buahnya, terdapat biji-biji yang dilapisi oleh pulp putih yang manis. Setiap buah kakao mengandung sekitar 20-50 biji cokelat. Biji-biji inilah yang kemudian diolah menjadi berbagai produk cokelat.



Gambar 2.1 Buah kakao
Sumber: Katarina (intisari.grid.id)

Proses pengolahan biji kakao dimulai dengan fermentasi dan pengeringan. Proses fermentasi dan pengeringan sangat penting karena merupakan faktor-faktor yang menentukan kualitas biji kakao kering dan berperan penting dalam pembentukan cita rasa dan aroma biji kakao. Proses fermentasi dengan kotak kayu

besar memiliki efektifitas lebih baik dibandingkan kotak kayu kecil. Hal ini dikarenakan perlakuan kotak kayu besar dapat mencapai suhu optimal fermentasi dan warna biji kakao yang dihasilkan lebih optimal (Rachmatullah et al., 2021).

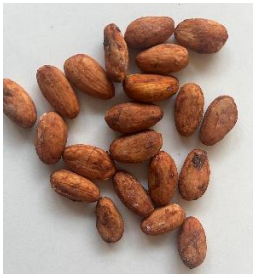
Proses pengeringan merupakan suatu cara yang digunakan dalam teknologi pangan untuk memperpanjang masa simpan produk dengan cara menguapkan sebagian besar kadar air tertentu dengan menggunakan energi panas sehingga menghambat laju kerusakan bahan akibat aktifitas biologis dan kimia (Hatta et al., 2019). Biji kakao selama proses pengeringan menunjukkan warna yang semakin gelap pada kedua perlakuan. Akan tetapi warna biji kakao pada perlakuan kotak kayu besar mengarah ke warna cokelat tua, sedangkan perlakuan kotak kayu kecil mengarah ke warna cokelat muda. Perbedaan warna tersebut berhubungan dengan keberhasilan proses fermentasi sebelumnya (Rachmatullah et al., 2021).

Menurut Rachmatullah, dkk (2021), dalam penelitian "Karakteristik Biji Kakao Hasil Fermentasi dengan Ukuran Wadah Berbeda,". Berikut perbandingan warna biji kakao mulai dari mentah hingga proses pengeringsan.

Tabel 2.1 Perbandingan warna biji kakao

Pengeringan hari ke -	Gambar	Warna	Keterangan
Pengeringan hari ke 1-2		Putih Berlendir	Basah
Pengeringan hari ke 3 - 5		Cokelat Muda	Setengah Kering

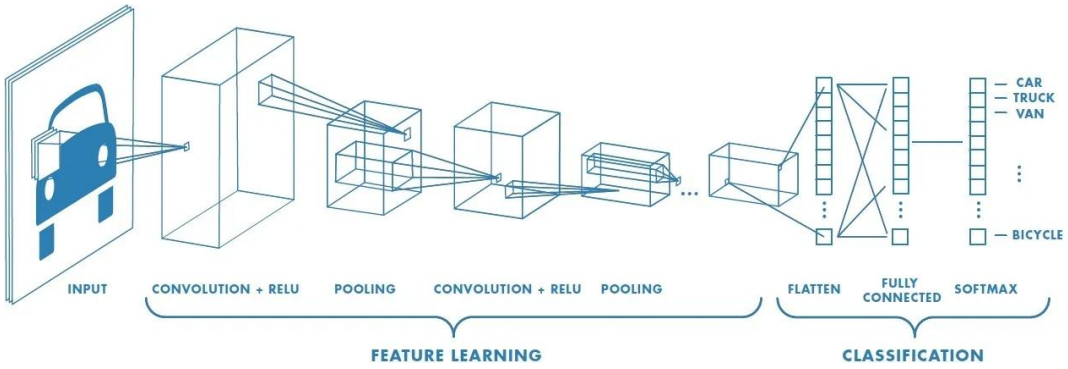
Lanjutan Tabel 2.1

Pengeringan hari ke -	Gambar	Warna	Keterangan
Pengeringan hari ke- 6 - 7		Cokelat Tua	Kering

Selama pengeringan, warna biji kakao mengalami perubahan bertahap. Pada hari pertama hingga kedua, biji kakao berwarna putih berlendir dan masih basah. Pada hari ketiga hingga kelima, biji berubah menjadi cokelat muda dengan kondisi setengah kering. Akhirnya, pada hari keenam hingga ketujuh, warna biji menjadi cokelat tua yang menunjukkan bahwa biji telah benar-benar kering. Perubahan warna ini dipengaruhi oleh keberhasilan fermentasi sebelumnya, di mana kotak fermentasi besar cenderung menghasilkan warna cokelat tua yang lebih optimal dibandingkan kotak kecil.

2. *Convolutional neural network* (CNN)

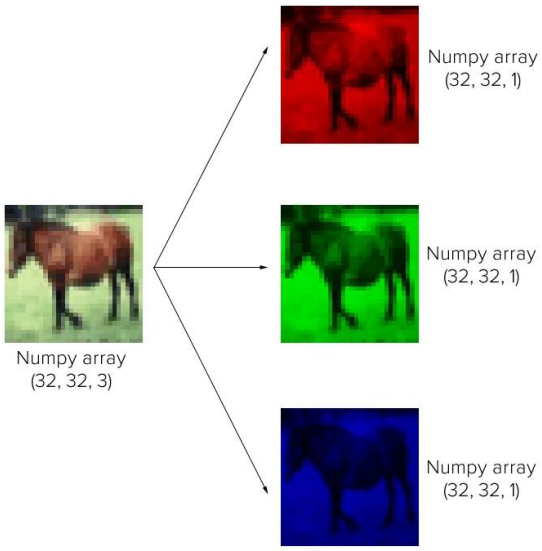
Convolutional Neural Network (CNN) adalah pengembangan dari Multilayer Perceptron (MLP) yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Neural Network* karena kedalaman jaringan yang tinggi dan banyak diaplikasikan pada data citra (I Wayan Suartika E. P, 2016). CNN terdiri dari beberapa lapisan utama, yaitu lapisan konvolusi *(convolutional layer)*, lapisan pooling, lapisan aktivasi, dan lapisan sepenuhnya terhubung *(fully connected layer)*.



Gambar 2.2 Ilustrasi arsitektur CNN

Sumber: medium.com

Lapisan konvolusi adalah inti dari CNN yang bertugas mengekstrak fitur dari citra dengan menggunakan filter (kernel). Filter ini digeser secara sistematis melintasi citra untuk menghasilkan peta fitur *(feature map)*. Setiap filter bertugas mendeteksi karakteristik tertentu seperti tepi horizontal, vertikal, atau pola lainnya.



Gambar 2.3 *Image* RGB

Sumber: Medium.com

Gambar di atas adalah sebuah gambar RGB (merah, hijau, biru) berukuran 32 x 32 piksel, yang pada dasarnya merupakan array multidimensi berukuran 32 x 32 x 3 (di mana 3 melambangkan jumlah saluran). Lapisan konvolusional terdiri dari neuron yang diatur untuk membentuk filter dengan dimensi panjang dan tinggi (piksel). Sebagai contoh, lapisan pertama yang digunakan untuk mengekstrak fitur biasanya adalah lapisan konversi berukuran 5 x 5 x 3, dengan panjang 5 piksel, tinggi 5 piksel, dan ketebalan/jumlah 3 yang berkaitan dengan saluran pada gambar. Ketiga filter tersebut diterapkan ke seluruh bagian gambar. Setiap pergerakan melakukan operasi "titik" antara input dan nilai filter untuk menghasilkan keluaran yang umumnya disebut peta aktivasi atau peta fitur.

Lapisan pooling bertugas mereduksi dimensi dari peta fitur yang dihasilkan oleh lapisan konvolusi tanpa menghilangkan informasi penting. Proses pooling dapat dilakukan dengan teknik seperti max pooling, yang mengambil nilai maksimum dari area tertentu, atau average pooling, yang mengambil nilai rata-rata. Pengurangan dimensi ini membantu mengurangi kompleksitas komputasi dan meningkatkan ketahanan model terhadap variasi kecil dalam data, seperti rotasi atau translasi.

Lapisan aktivasi, seperti fungsi ReLU *(Rectified Linear Unit)*, diterapkan setelah setiap operasi konvolusi untuk memperkenalkan non-linearitas ke dalam model. Fungsi aktivasi ini penting karena memungkinkan jaringan mempelajari hubungan non-linear yang kompleks dalam data. Sementara itu, lapisan sepenuhnya terhubung *(fully connected layer)* menghubungkan setiap neuron di lapisan sebelumnya dengan neuron di lapisan berikutnya. Bagian ini bertugas

menggabungkan informasi yang diekstrak dari lapisan sebelumnya untuk menghasilkan output akhir seperti label klasifikasi.

Selain itu, CNN memanfaatkan mekanisme seperti *dropout* untuk mencegah overfitting, yaitu kondisi ketika model terlalu cocok dengan data pelatihan tetapi tidak bekerja dengan baik pada data baru. CNN juga dilengkapi dengan fungsi optimasi, seperti *stochastic gradient descent* (SGD) untuk meminimalkan kesalahan prediksi. Dengan arsitektur yang modular dan fleksibel, CNN telah digunakan secara luas dalam aplikasi seperti pengenalan wajah, analisis medis, pengenalan pola, dan banyak lagi, menjadikannya salah satu metode paling populer dalam deep learning.

CNN juga memiliki aplikasi yang sangat luas di berbagai bidang. Dalam bidang medis, CNN digunakan untuk mendeteksi penyakit dari citra radiologi, seperti kanker pada hasil CT scan atau MRI. Di sektor teknologi, CNN banyak diterapkan pada sistem pengenalan wajah, deteksi objek, dan pemrosesan video. Selain itu, CNN juga diterapkan dalam pengembangan mobil otonom untuk mengenali rambu lalu lintas dan mendeteksi objek di jalan. Beragamnya aplikasi menunjukkan fleksibilitas CNN dalam menangani berbagai jenis data visual.

Meskipun memiliki banyak keunggulan, CNN juga memiliki beberapa tantangan. Salah satunya adalah kebutuhan akan sumber daya komputasi yang besar, terutama untuk dataset besar dan arsitektur yang dalam. Selain itu, proses pelatihan CNN dapat memakan waktu lama dan memerlukan hyperparameter tuning yang cermat untuk mendapatkan hasil optimal. Namun, dengan kemajuan teknologi perangkat keras seperti GPU dan TPU, serta algoritma optimisasi yang

terus berkembang, CNN terus menjadi alat yang semakin kuat untuk menangani masalah dalam pemrosesan citra dan data visual.

3. Android

Android merupakan sistem berbasis *Linux* yang dirancang khusus untuk digunakan pada ponsel pintar, tablet, dan jam pintar. Sistem operasi Android, yang dikembangkan oleh *Google*, kini menjadi salah satu *platform* yang paling banyak digunakan secara global. Android menawarkan berbagai fitur dan kemampuan yang mencakup antarmuka pengguna yang intuitif, kemampuan multitasking, akses ke jutaan aplikasi melalui *Google Play Store*, dan integrasi yang erat dengan layanan *Google*.

Salah satu daya tarik utama dari Android adalah fleksibilitasnya. Sistem operasi ini memiliki sifat *open-source*, yang berarti kode sumbernya dapat diakses oleh pengembang dan dimodifikasi sesuai kebutuhan. Hal ini memungkinkan adanya berbagai varian Android yang dikembangkan oleh produsen perangkat seluler, sehingga pengguna dapat memilih perangkat dengan tampilan dan fitur yang sesuai dengan preferensi mereka. Fleksibilitas ini juga memungkinkan pengembangan aplikasi yang kaya dan inovatif untuk memenuhi kebutuhan pengguna.

Dengan popularitas dan adopsi yang luas, Android telah menjadi ekosistem yang dinamis dan berkembang pesat. Pengguna Android dapat menikmati berbagai fitur canggih, mengakses ribuan aplikasi yang beragam, dan menjelajahi dunia digital dengan mudah. Keberadaan Android telah mengubah cara kita berkomunikasi, bekerja, dan bersosialisasi, serta memberikan pengalaman yang

terus berkembang dalam dunia perangkat seluler. Dalam beberapa tahun terakhir, Android juga telah memperluas dukungan untuk teknologi baru seperti kecerdasan buatan, *augmented reality (AR)*, dan *virtual reality (VR)*. Dengan integrasi yang lebih dalam dengan teknologi ini, Android terus menghadirkan pengalaman yang semakin imersif dan interaktif bagi pengguna.



Gambar 2.4 Logo android
Sumber: android.com

Secara keseluruhan, perkembangan Android telah memainkan peran yang signifikan dalam mengubah cara kita menggunakan perangkat seluler. Dengan inovasi terus menerus, kolaborasi yang luas, dan kemampuan yang berkembang, Android terus memperkaya dan memperluas ekosistemnya, memberikan pengalaman yang lebih baik dan membuka peluang baru dalam dunia teknologi.

4. Android studio IDE

Android Studio adalah *Integrated Development Environment* (IDE) resmi untuk membuat aplikasi berbasis Android yang tersedia secara gratis atau sebagai open source (Juansyah, 2015). Android Studio menawarkan berbagai fitur yang memudahkan proses pengembangan, seperti editor kode yang canggih, alat *debugging*, emulator untuk pengujian aplikasi, dan integrasi dengan berbagai pustaka serta alat pengembangan lainnya. Pada konferensi Google I/O 2013 pada

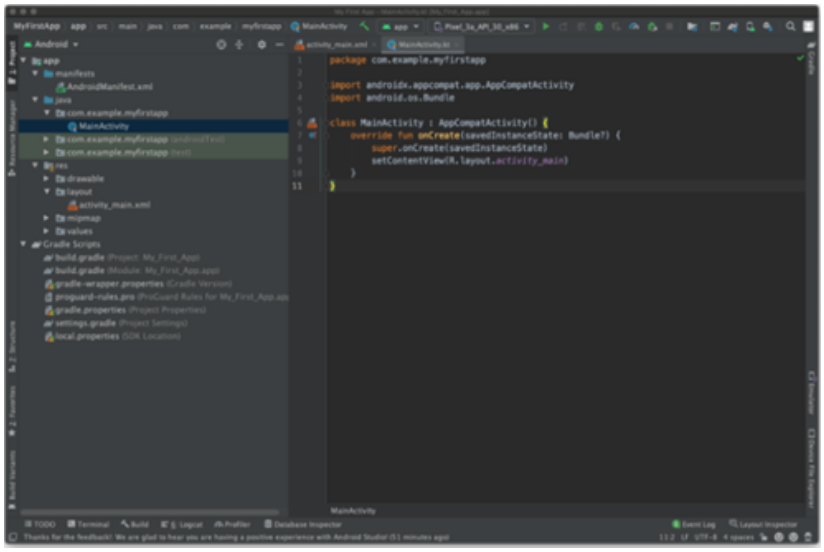
16 Mei 2013, Google mengumumkan peluncuran Android Studio. Sejak saat itu, Android Studio menjadi IDE resmi untuk pengembangan aplikasi Android menggantikan Eclipse.

Berbeda dengan Eclipse, Android Studio didasarkan pada Intellij IDEA dan dilengkapi dengan *plugin* ADT *(Android Development Tools)*. Beberapa fitur utama Android Studio meliputi:

1. Pembangunan proyek berbasis Gradle.
2. *Refactoring* dan perbaikan kesalahan.
3. Dengan *tools* lint, para pengembang dapat dengan cepat memantau kecepatan, manfaat, dan daya saing aplikasi mereka.
4. Keamanan melalui *Proguard* dan S*igning* aplikasi.
5. Kemudahan GUI Aplikasi Android. (Juansyah, 2015).



Gambar 2.5 Tampilan awal android studio
Sumber: Android developer (developer.android.com)



Gambar 2.6 Halaman kerja android studio
Sumber:Android developer (developer.android.com)

5. *Kotlin*

Kotlin adalah bahasa pemrograman lintas platform yang sepenuhnya kompatibel dengan Java. Versi dari *Java Virtual Machine* (JVM) pada pustaka standar *Kotlin* didasarkan pada Pustaka Kelas Java, namun penggunaan inferensi tipe membuat sintaksnya lebih mudah. Pada 7 Mei 2019, *Google* mengumumkan bahwa *Kotlin* sekarang menjadi bahasa yang lebih disukai bagi para pengembang aplikasi Android. Sejak dirilisnya Android Studio 3.0 pada Oktober 2017, *Kotlin* telah menjadi compiler Java alternatif.



Gambar 2.7 Logo *kotlin*
Sumber: mailslurp.com

Kompiler Android *Kotlin* menghasilkan bytecode Java 8 secara default (yang berjalan di JVM yang lebih baru), tetapi memungkinkan pemrogram memilih untuk menargetkan Java 9 hingga 18, untuk pengoptimalan, atau mengizinkan lebih banyak fitur; memiliki dukungan interoperabilitas kelas rekaman dua arah untuk JVM, diperkenalkan di Java 16, dianggap stabil sejak *Kotlin* 1.5. *Kotlin* memiliki dukungan untuk web; dengan mengkompilasi ke *JavaScript* (yaitu, *Kotlin/JS* dengan *back-end* klasik, dinyatakan stabil sejak versi 1.3), sedangkan *Kotlin/JS* (berbasis IR) yang lebih baru dalam versi beta pada versi 1.5.30. *Kotlin/Native* dianggap beta sejak versi 1.3.

6. *Python*

(Rahman et al., 2023). *Python* adalah bahasa pemrograman yang menggunakan interpreter untuk menjalankan kode programnya. Interpreter tersebut dapat menerjemahkan kode secara langsung, dan *Python* bisa berjalan pada berbagai *platform* seperti *Windows*, *Linux*, dan sebagainya. *Python* merupakan bahasa pemrograman tingkat tinggi yang mendapatkan popularitas pesat dalam dunia pengembangan perangkat lunak, dirancang dengan sintaksis yang mudah dipahami dan bersifat dinamis. Ini menjadikannya pilihan utama baik untuk pengembang pemula maupun yang berpengalaman, dengan mendukung paradigma pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional, *Python* memberikan tingkat fleksibilitas yang tinggi dalam menangani berbagai jenis proyek pengembangan perangkat lunak.

Tabel 2.2 Contoh penulisan bahasa *python*

```
@app.route('/start_camera', methods=['POST'])
def start_camera():
    global cap
    if not cap.isOpened():
        cap = cv2.VideoCapture(0)
        cap.set(4, 240)
    return redirect(url_for('index'))

@app.route('/stop_camera', methods=['POST'])
def stop_camera():
    global cap
    if cap.isOpened():
        cap.release()
    return redirect(url_for('index'))
```

Keunggulan *Python* tidak hanya terletak pada kejelasan sintaksisnya, tetapi juga pada ekosistemnya yang kaya dengan berbagai pustaka dan *framework*. *Python* digunakan secara luas dalam berbagai bidang seperti pengembangan web, pengolahan data, kecerdasan buatan, dan pengembangan perangkat lunak umum. Dengan berbagai modul seperti NumPy, Pandas, dan Django, *Python* menjadi pilihan kuat untuk berbagai proyek.

Selain itu, *Python* memiliki komunitas pengguna yang besar dan aktif, memberikan pengembang akses mudah ke dukungan dan sumber daya untuk mengatasi tantangan pengembangan yang dihadapi. Dengan filosofi "baterai termasuk," *Python* mendorong pengembang untuk fokus pada pemecahan masalah daripada mengelola detail teknis. Kombinasi kesederhanaan sintaksis, kekuatan fungsional, dan dukungan komunitas menjadikan *Python* bahasa yang sangat efisien dan cepat untuk prototip, pengujian, dan pengembangan perangkat lunak pada skala besar.

7. *Tensorflow*

TensorFlow adalah kerangka kerja komputasi untuk membangun model pembelajaran mesin. *TensorFlow* menyediakan berbagai *toolkit* yang memungkinkan untuk membuat model pada tingkat abstraksi yang disukai dan dapat menjalankan grafik pada beberapa *platform hardware*, termasuk CPU, GPU, dan TPU (Nasha Hikmatia A.E. & Zul, 2021). *TensorFlow* adalah sebuah platform open-source yang dikembangkan oleh Google untuk keperluan *machine learning* (pembelajaran mesin) dan *deep learning* (pembelajaran mendalam). Pertama kali dirilis pada tahun 2015, *TensorFlow* memungkinkan para peneliti dan pengembang untuk membangun dan menerapkan model *machine learning* dengan berbagai tingkat kompleksitas. *Platform* ini mendukung berbagai bahasa pemrograman, termasuk *Python*, *C++*, dan *JavaScript*, serta menyediakan ekosistem alat yang komprehensif untuk pengembangan model, pelatihan, dan *deployment*.



Gambar 2.8 Logo *tensorflow*
Sumber: *tensorflow*.org

TensorFlow Lite adalah versi ringan dari *TensorFlow* yang dirancang khusus untuk perangkat mobile dan embedded systems. *TensorFlow Lite* memungkinkan model machine learning dijalankan pada perangkat dengan sumber daya terbatas, seperti *smartphone*, mikroprosesor, dan perangkat *IoT (Internet of Things)*. Keunggulan utama dari *TensorFlow Lite* adalah ukurannya yang lebih

kecil dan efisiensi komputasinya yang lebih tinggi, sehingga ideal untuk aplikasi yang membutuhkan inferensi cepat dan penggunaan daya yang rendah. *TensorFlow Lite* mendukung berbagai format model dan optimasi, seperti model kuantisasi dan pemangkasan untuk mengurangi ukuran dan mempercepat waktu inferensi.

Kemampuan untuk menjalankan model machine learning pada perangkat edge, *TensorFlow Lite* membuka banyak peluang baru dalam pengembangan aplikasi mobile dan IoT. Misalnya, pengenalan gambar, pemrosesan bahasa alami, dan deteksi anomali dapat dijalankan langsung di perangkat tanpa perlu mengirim data ke server, yang mengurangi latensi dan meningkatkan privasi data. Dukungan komunitas dan dokumentasi yang kuat juga membuat *TensorFlow Lite* menjadi pilihan populer di kalangan pengembang aplikasi mobile dan embedded.

8. *Image classification*

Image classification adalah salah satu bidang penting dalam computer vision yang bertujuan untuk mengkategorikan gambar ke dalam kelas-kelas yang telah ditentukan (Dharmadi, 2018). Teknik ini memanfaatkan model pembelajaran mesin, terutama deep learning, untuk mengenali pola dan fitur dalam gambar. Model convolutional neural network (CNN) sering digunakan karena kemampuannya yang luar biasa dalam menangani data gambar. CNN secara otomatis dapat mengidentifikasi fitur-fitur penting seperti tepi, tekstur, dan bentuk yang membedakan satu kelas gambar dari yang lain. Proses ini melibatkan tahap pelatihan di mana model dilatih pada dataset besar yang berisi gambar yang sudah diberi label.

Dalam aplikasi praktis, image classification memiliki banyak kegunaan. Di bidang medis, misalnya, model klasifikasi gambar dapat digunakan untuk mendiagnosis penyakit dengan menganalisis citra medis seperti X-ray atau MRI. Dalam industri otomotif, teknologi ini diterapkan pada sistem pengenalan objek untuk mobil otonom, membantu kendaraan mengenali rambu lalu lintas, pejalan kaki, dan kendaraan lain. Selain itu, di sektor e-commerce, image classification digunakan untuk mengatur dan mengelompokkan produk berdasarkan gambar, memudahkan pelanggan menemukan barang yang mereka cari.

Pengembangan model klasifikasi gambar melibatkan beberapa langkah penting. Pertama, diperlukan dataset yang representatif dan beragam untuk melatih model. Data ini harus melalui tahap preprocessing seperti resizing, normalisasi, dan augmentasi untuk meningkatkan kualitas dan variasi data. Selanjutnya, model dibangun dan dilatih menggunakan framework seperti TensorFlow atau PyTorch. Setelah pelatihan, model diuji menggunakan dataset yang berbeda untuk mengevaluasi akurasi dan performanya. Model yang telah terlatih dan diuji kemudian dapat diimplementasikan dalam aplikasi nyata, memberikan prediksi klasifikasi gambar secara otomatis.

9. *Google Colab*

Google Colab adalah Salah satu tools pengembangan machine learning yang dinilai user-friendly. Google Colaboratory atau Google Colab merupakan executable documents yang bisa pengguna manfaatkan untuk menyimpan, menulis, dan membagikan program yang sudah mereka tulis melalui Google Drive. Salah satu manfaat dari Google Colab adalah built in machine learning yang lengkap,

yang merupakan salah satu fitur paling populer dari Google Collab. Dengan fitur ini, pengguna bisa melakukan berbagai aktivitas seperti mengimport set data gambar atau melakukan evaluasi model melalui beberapa baris kode saja. Jadi, notebook yang akan melakukan eksekusi kode tersebut pada server cloud milik Google. Google Collab memiliki banyak perpustakaan machine learning, seperti PyTorch, TensorFlow, dan Keras.

10. *Unified modeling language* (UML)

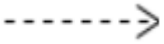
Unified Modeling Language (UML) adalah sebuah bahasa pemodelan perangkat lunak yang telah distandardisasi sebagai media penulisan cetak biru (*blueprints*) perangkat lunak (Pressman). UML bisa saja digunakan untuk visualisasi, spesifikasi, kontruksi dan dokumentasi beberapa bagian-bagian dari system yang ada dalam perangkat lunak. Dalam kata lain, seperti halnya seorang arsitek dalam membuat dokumen cetak biru yang digunakan oleh perusahaan konstruksi untuk membangun sebuah bangunan, arsitek perangkat lunak membuat diagram-diagram UML untuk membantu programmer/ developer membangun perangkat lunak. Untuk selanjutnya, semakin kita mengetahui beberapa kosakata yang digunakan UML, kita akan semakin mudah dalam memahami spesifik (Sumiati et al., 2021). *Unified Modelling Language* (UML) menyediakan beberapa jenis diagram yang dapat dikelompokkan berdasar sifatnya antara lain:

a. Diagram *Use Case*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor

dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu (Dharwiyanti & Wahono, 2003).

Tabel 2.3 Simbol *use case diagram*

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

Lanjutan Tabel 2.3

No	Gambar	Nama	Keterangan
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemenelemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi.

b. *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. (Dharwiyanti & Wahono, 2003).

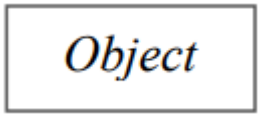
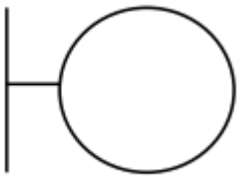
Tabel 2.4 Simbol *activity diagram*

No	Gambar	Nama	Keterangan
1		*Activity*	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi.
2		*Action*	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		*Initial Node*	Bagaimana objek dibentuk atau diawali.
4		*Activity Final Node*	Bagaimana objek dibentuk dan dihancurkan
5		*Fork Node*	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

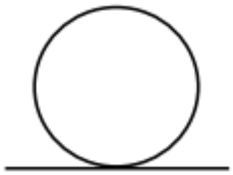
c. *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. (Dharwiyanti & Wahono, 2003).

Tabel 2.5 Simbol *sequence diagram*

No	Gambar	Nama	Keterangan
1		*Object*	*Object* atau biasa juga disebut partisipan merupakan *instance* dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah *class* (kotak) dengan nama objek di dalamnya yang diawali dengan sebuah titik koma.
2		*Actor*	*Actor* juga dapat berkomunikasi dengan *object*, maka *actor* juga dapat diurutkan sebagai kolom.
3		*Life line*	*Life line* mengidentifikasikan keberadaan sebuah *object* dalam basis waktu. Notasi untuk *lifeline* adalah garis putus-putus vertical yang di tarik oleh sebuah *object*
4		*Collaborations*	*Activication* dinotasikan sebagai kotak segi empat yang digambar pada sebuah lifeline action mengidentifikasikan sebuah *object* yang akan beraksi.
5		*Boundary*	*Boundary* terletak diantara system dengan dunia di sekelilingnya. Semua form, laporan-laporan, antara muka ke perangkat keras seperti printer atau scenner dan antara muka ke system lainnya adalah termasuk dalam kategori.

Lanjutan Tabel 2.5

No	Gambar	Nama	Keterangan
6		*Control*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaharui elemen yang bergantun padanya elemen yang tidak mandiri.
7		*Entity*	*Entity* digunakan untuk menengani informasi yang akan disimpan secara permanen.

d. *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain (Dharwiyanti & Wahono, 2003).

Tabel 2.6 Simbol *class diagram*

No	Gambar	Nama	Keterangan
1		*Assosiation*	Hubungan statis antar kelas. Asosiasi menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui eksistensi kelas lain. Asosiasi biasanya disertai dengan *multiplicity*
2		*Directed Assosiation*	asosiasi dengan makna kelas yang satu digunakan oleh kelas yang lain. Asosiasi bearah juga biasanya disertai dengan *multyplicity*
3		*Generalization*	relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus) atau untuk menyatakan hubungan *inheritance*
4		*Dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
5		*Agregation*	Hubungan yang menyatakan bahwa suatu kelas menjadi atribut bagi kelas lain
6		*Composition*	Bentuk khusus dari agregasi dimana kelas yang menjadi bagian diciptakan setelah kelas menjadi *whole*. Misalnya kelas *whole* dihapus, maka kelas yang menjadi part ikut musnah
7		*Realization*	Hubungan antar kelas dimana sebuah kelas memiliki keharusan untuk mengikuti aturan yang ditetapkan oleh kelas lainnya.

11. *Flowchart*

Flowchart adalah adalah suatu bagan dengan simbol-simbol tertentu yang menggambarkan urutan proses secara mendetail dan hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program (Rahmat Arifianto, 2014). *Flowchart* merupakan salah satu diagram yang digunakan untuk merepresentasikan proses atau alur kerja dari suatu sistem atau program. *Flowchart* Dalam bidang informatika, digunakan untuk merepresentasikan alur kerja dari suatu program atau sistem komputer. *Flowchart* dapat mempermudah pengembangan program karena dapat membantu pengembang dalam memahami alur kerja dari program yang dibangun.

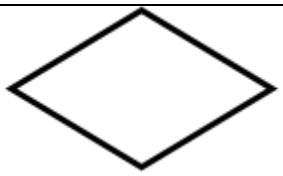
Menurut buku "*Information Systems Development: Reflections, Challenges and New Directions*" karya Robert D. Galliers dan Wendy L. Currie, *flowchart* merupakan alat bantu yang sangat penting dalam pengembangan sistem informasi. *Flowchart* dapat digunakan untuk merepresentasikan proses bisnis atau alur kerja dari suatu sistem informasi. *Flowchart* juga dapat digunakan sebagai alat untuk mengevaluasi dan mengoptimalkan proses bisnis atau sistem informasi yang sudah ada.

Flowchart juga dapat digunakan untuk memudahkan komunikasi antara pengembang program dengan stakeholder atau pengguna akhir. *Flowchar* dalam buku "*System Analysis and Design: An Object-Oriented Approach with UML*" karya Alan Dennis, Barbara Haley Wixom, dan David Tegarden, disebutkan bahwa *flowchart* dapat digunakan untuk memudahkan komunikasi antara pengembang program dengan pengguna akhir. *Flowchart* dapat membantu pengguna akhir

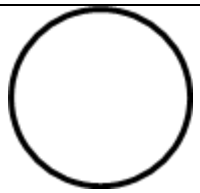
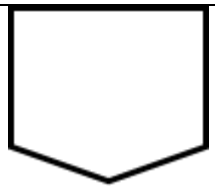
dalam memahami alur kerja dari program yang dibangun sehingga dapat memberikan masukan yang lebih baik dalam proses pengembangan program.

Berikut ini adalah beberapa simbol yang digunakan dalam menggambar suatu *flowchart* dapat dilihat pada tabel 2.7.

Tabel 2.7 Simbol *flowchart*

Simbol	**Nama**	**Fungsi**
	*Terminator*	Permulaan / akhir program
	*Garis alir* (flow line)	Arah aliran program
	*Preparation*	Proses inisialisasi / pemberian harga awal
	Proses	Proses perhitungan / proses pengolahan data
	*Input* / *output* data	Proses input / output data, parameter, informasi
	*Predefined process* (sub program)	Permulaan sub program / proses menjalankan sub program
	*Decision*	Perbandingan pernyataan, penyeleksian data yang memberikan pilihan untuk langkah selanjutnya

Lajutan Tabel 2.7

Simbol	Nama	Fungsi
	*On page connector*	Penghubung bagian-bagian flowchart yang berada pada satu halaman
	*Off page connector*	Penghubung bagian-bagian flowchart yang berada pada halaman berbeda

B. Kajian Hasil Penelitian Terdahulu

Penelitian ini dilakukan tidak lepas dari hasil penelitihan terdahulu yang dilakukan sebagai kajian. Adapun karya ilmiah yang penulis maksud adalah sebagai berikut:

1. Agung Rizqi Hidayat, Veronica Lusiana (2022), Program Studi Teknik Informatika, Universitas Stikubank dalam sebuah jurnal yang berjudul **"Deteksi Jenis Sayuran dengan *Tensorflow* Dengan Metode *Convolutional Neural Network*"**. Penelitian ini memanfaatkan metode *Convolutional Neural Network* (CNN) untuk mengklasifikasikan jenis sayuran. Dengan menggunakan dataset yang terdiri atas 17 jenis sayuran dan 2.550 gambar, penelitian ini menunjukkan bahwa CNN mampu mengidentifikasi objek citra dengan akurasi rata-rata 70%, dan akurasi tertinggi mencapai 86%. Aplikasi berbasis Android dikembangkan menggunakan *Python* dan *TensorFlow*, dirancang untuk mengenali berbagai

jenis sayuran secara tepat dan akurat melalui teknologi *smartphone*, sehingga mempermudah pengguna dalam pengklasifikasian.

2. Khairul Azmi, Sarjon Defit, Sumijan (2023), Ilmu Komputer, Prodi Teknik Informatika, Universitas Putra Indonesia YPTK Padang dalam sebuah jurnal yang berjudul **"Implementasi *Convolutional Neural Network* (CNN) Untuk Klasifikasi Batik Tanah Liat Sumatera Barat"**. Penelitian ini bertujuan mengkaji kemampuan *Convolutional Neural Network* (CNN) sebagai model *deep learning* untuk mengklasifikasikan motif batik tanah liat khas Sumatera Barat, sebuah warisan budaya yang unik dan langka. Penelitian ini menggunakan 400 citra batik yang dibagi ke dalam empat kelas motif, dengan proporsi 320 citra sebagai data latih dan 80 citra sebagai data uji. Data citra diambil melalui survei lapangan dengan kamera DSLR dan *smartphone,* lalu diproses melalui teknik *pre-processing*, seperti *cropping* dan *translation*, guna meningkatkan kualitas dan variasi data. Hasil penelitian menunjukkan bahwa CNN mampu mencapai tingkat akurasi sebesar 98,75% pada data latih dan 62,5% pada data uji. Meskipun akurasi pada data uji lebih rendah, hasil ini cukup baik untuk membangun aplikasi pengenalan motif batik secara umum. Penelitian ini juga mengungkapkan faktor-faktor yang memengaruhi hasil klasifikasi, seperti posisi gambar, kualitas pencahayaan, dan kemiripan motif antarkelas, sehingga memerlukan peningkatan pada proses akuisisi data dan pemodelan lebih lanjut. Dengan keberhasilan implementasi CNN ini, penelitian ini menjadi dasar untuk pengembangan aplikasi klasifikasi motif batik tanah liat Sumatera Barat yang lebih canggih. Penulis juga mengusulkan pengembangan

global untuk klasifikasi motif batik dari berbagai daerah di Indonesia, mengingat keberagaman motif yang menjadi ciri khas budaya Nusantara. Hal ini berpotensi memperkaya promosi dan pelestarian batik sebagai warisan budaya yang diakui dunia.

3. Muhammad Sayyidin Hawibowo, Izzati Muhimmah (2024), Program Studi Informatika, Universitas Islam Indonesia dalam sebuah jurnal yang berjudul "**Aplikasi Pendeteksi Tingkat Kematangan Pepaya menggunakan Metode *Convolutional Neural Network* (CNN) Berbasis Android**". Penelitian ini fokus pada pengembangan aplikasi berbasis Android yang menggunakan CNN untuk mendeteksi tingkat kematangan buah pepaya, yang dibagi menjadi tiga kategori: mentah, setengah matang, dan matang. Dengan akurasi mencapai 96,97%, aplikasi ini dirancang untuk memudahkan petani dalam memantau kematangan pepaya dan mengoptimalkan proses panen. Penelitian ini melibatkan pengumpulan *dataset* dari kebun pepaya di Yogyakarta, menggunakan kamera ponsel dengan resolusi tinggi. Model CNN yang dikembangkan terdiri dari beberapa lapisan konvolusi untuk mengekstraksi fitur gambar, disertai dengan proses pelatihan menggunakan dataset yang telah diproses. Dataset dibagi menjadi data pelatihan, validasi, dan pengujian untuk memastikan akurasi model. Hasil evaluasi menunjukkan kemampuan model dalam mengklasifikasikan tingkat kematangan secara efektif pada data uji. Selain deteksi kematangan, aplikasi ini juga menawarkan fitur prediksi waktu panen dan pengelolaan data melalui pencatatan. Penggunaan format *TensorFlow Lite* memungkinkan integrasi model CNN ke dalam aplikasi

Android, sehingga pengguna dapat dengan mudah memanfaatkan teknologi ini di lapangan. Dengan aplikasi ini, petani tidak hanya dapat meningkatkan efisiensi tetapi juga mengurangi pemborosan hasil panen melalui deteksi yang lebih akurat.

4. Doni Darmawan (2023), Fakultas Sains dan Teknologi, Prodi Sistem Informasi, Universitas Jambi dalam sebuah jurnal yang berjudul **"Implementasi Metode *Convolutional Neural Network* (CNN) dalam Mendeteksi Jenis Sampah"**. Penelitian ini membahas implementasi *Convolutional Neural Network* (CNN) untuk mendeteksi jenis sampah, khususnya untuk mengidentifikasi sampah yang dapat didaur ulang atau tidak. CNN dipilih karena keunggulannya dalam pengolahan citra dan kemampuannya menghasilkan model klasifikasi yang akurat. Peneliti menggunakan teknik *Transfer Learning* dengan pretrained model *InceptionV3*, yang telah terbukti memberikan hasil optimal dalam tugas pengklasifikasian citra. Dataset yang digunakan dalam penelitian ini terdiri dari 15 jenis sampah dengan total 18.733 gambar, diambil dari sumber seperti Kaggle dan GitHub. Penelitian ini membagi data menjadi 90% untuk pelatihan dan 10% untuk pengujian, dengan proses preprocessing seperti resizing gambar untuk memastikan kompatibilitas dengan model. *TensorFlow*, sebuah library *open-source* yang populer untuk komputasi numerik dan *deep learning*, menjadi alat utama dalam membangun model CNN dalam penelitian ini. *TensorFlow* digunakan untuk mendukung semua tahapan proses, mulai dari preprocessing data, pembangunan model CNN, hingga evaluasi performa model. Versi *TensorFlow 2.12* digunakan karena integrasinya dengan Keras,

yang mempermudah pengembangan model dengan *API* yang lebih ramah pengguna. *TensorFlow* juga menyediakan fitur seperti optimasi model dengan *Adam optimizer* dan fungsi aktivasi seperti ReLU dan Softmax untuk meningkatkan akurasi klasifikasi. Peneliti juga memanfaatkan *TensorFlow* untuk memproses data dalam batch mini sehingga mempercepat proses pelatihan dan mengurangi kebutuhan memori. Hasil penelitian menunjukkan bahwa penggunaan *TensorFlow* memungkinkan model untuk mencapai akurasi pelatihan hingga 99%, validasi 92%, dan pengujian 81%. Selain itu, model yang dibangun diintegrasikan ke dalam aplikasi berbasis web menggunakan *REST API*, memberikan akses yang fleksibel kepada pengguna untuk mendeteksi jenis sampah. Dengan dukungan *TensorFlow*, penelitian ini tidak hanya menghasilkan model yang efektif tetapi juga aplikasi praktis yang dapat membantu masyarakat dalam memilah sampah, meningkatkan efisiensi daur ulang, dan mendukung upaya pelestarian lingkungan.

5. Zahrotul Ilmi Wijayanti (2024), Prodi Teknik Informatika, Universitas Muhammadiyah Ponorogo dalam sebuah jurnal yang berjudul **"Penerapan Teknologi CNN dalam Proses Pendeteksi Kematangan Buah Stroberi"**. Penelitian ini mengembangkan model *Convolutional Neural Network* (CNN) untuk mendeteksi kematangan buah stroberi secara otomatis berdasarkan citra digital. Model ini dirancang untuk mengatasi tantangan kematangan yang tidak merata pada buah stroberi yang dipengaruhi oleh paparan sinar matahari. Dengan menggunakan dataset sebanyak 521 gambar dari Kaggle, peneliti mengklasifikasikan stroberi ke dalam dua kategori: matang dan belum matang.

Penelitian ini bertujuan meningkatkan efisiensi proses panen dan menjaga kualitas olahan berbahan dasar stroberi. Metode penelitian melibatkan penggunaan CNN yang diimplementasikan melalui *library TensorFlow* dan *Keras* di lingkungan *Google Colab*. Proses meliputi pengumpulan data, preprocessing menggunakan *ImageDataGenerator* untuk augmentasi data, dan pelatihan model dengan parameter seperti resolusi gambar 150x150 piksel dan 30 epoch. Selama pelatihan, model mencapai akurasi validasi sebesar 98% dengan loss serendah 0,0383, menunjukkan kemampuan yang kuat dalam menggeneralisasi data baru. Evaluasi menggunakan data validasi menunjukkan hasil konsisten yang mendukung keandalan model. Hasil akhir penelitian menunjukkan bahwa teknologi CNN dapat mempermudah proses panen stroberi dengan akurasi tinggi dalam membedakan buah matang dan belum matang. Model ini diharapkan membantu petani dan industri pengolahan stroberi meningkatkan efisiensi dan kualitas produk. Selain itu, penelitian ini menyoroti potensi pengembangan lebih lanjut dalam penerapan CNN untuk pengelolaan hasil panen di sektor pertanian.

C. Kerangka Berpikir

Latar belakang penelitian ini untuk mengimplementasikan algoritma *Convolutional Neural Network* (CNN) dalam mendeteksi tingkat kekeringan pada biji kakao berbasis teknologi *machine learning*.

CNN dipilih karena kemampuannya dalam mengolah data visual dan mengenali pola dari gambar secara efektif, terutama dalam tugas klasifikasi.

↓

Penelitian ini memanfaatkan algoritma kecerdasan buatan untuk mengklasifikasikan tingkat kekeringan biji kakao berdasarkan warna biji. Mengintegrasikan model klasifikasi ke dalam aplikasi berbasis Android untuk memberikan solusi yang praktis dan mudah diakses oleh pengguna, termasuk petani dan industri kakao.

↓

Penelitian ini menggunakan teknologi *TensorFlow* untuk melatih model *Convolutional Neural Network* (CNN) dengan dataset berupa gambar biji kakao. Bahasa pemrograman *Python* digunakan untuk melatih model tersebut, sedangkan *Kotlin* digunakan untuk mengintegrasikan *TensorFlow* dengan aplikasi Android.

↓

Hasil penelitian berupa aplikasi android model CNN yang mampu mengklasifikasikan atau mendeteksi tingkat kekeringan biji kakao ke dalam tiga kategori: basah, setengah kering, dan kering

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Penelitian ini menggunakan metode eksperimen dengan pendekatan kuantitatif. Peneliti akan merancang sebuah aplikasi pendeteksi tingkat kekeringan pada biji kakao berbasis android. Setelah itu, akan dilakukan pengujian dan evaluasi terhadap performa aplikasi tersebut.

B. Lokasi dan Waktu

Lokasi penelitian dilaksanakan di wilayah desa Silopo, kecamatan binuang, Kabupaten Polewali Mandar, dan waktu penelitian dilakukan selama +1 (satu) bulan dimulai pada bulan januari sampai dengan bulan juni 2025.

Tabel 3.1 Jadwal pelaksanaan penelitian

No	Uraian Kegiatan	Bulan 2025		
		Januari	Maret	Juni
1	Studi Literatur			
2	Perancangan & Pembuatan Aplikasi			
3	Pengujian Aplikasi			
4	Analisa hasil pengujian			

C. Alat dan Bahan

Untuk melakukan proses penelitian dalam pembuatan aplikasi, diperlukan perangkat keras dan perangkat lunak sebagai alat dan bahan guna mendukung kegiatan penelitian tersebut. Berikut ini merupakan penjelasan dari alat dan bahan yang digunakan.

1. Alat

Alat yang digunakan untuk membuat aplikasi dapat dilihat pada tabel 3.2 sebagai berikut:

Tabel 3.2 Alat

No	Alat	Spesifikasi
1.	Laptop	*Processor*: Intel® Core™ i5-12500H RAM: 16 GB LPDDR5 SSD: 512GB
2.	Android Studio	versi Electric Eel (2022.1.1.)
3.	*TensorFlow*	Versi 2.14.0
4.	Google Colab	*Library*: Mendukung *TensorFlow*
4.	Bahasa Pemrograman: *Kotlin, Python*	Versi 2.1.0 Versi 3.10.9

2. Bahan

Bahan yang digunakan untuk membuat aplikasi dapat dilihat pada tabel 3.3 sebagai berikut:

Tabel 3.3 Bahan

Bahan	Spesifikasi
Dataset	1) Gambar biji kakao basah 2) Gambar biji kakao setengah kering 3) Gambar biji kakao kering

D. Teknik Pengumpulan Data

Penelitian ini akan difokuskan pada perancangan dan pembuatan aplikasi serta melakukan pengujian dan evaluasi yang akan dilakukan di dalam ruangan. Pengumpulan data dilakukan dengan 2 metode:

1. Wawancara

Wawancara dilakukan peneliti untuk mendapatkan penerangan langsung dari narasumber dalam hal ini yaitu petani buah kakao.

2. Dataset

Pengumpulan data dengan dataset ini bertujuan untuk mendapatkan detail gambar yang akan digunakan sebagai bahan training model.

E. Metode Pengujian

Pengujian sistem dilakukan secara uji kode program dan uji implementasi hasil program dengan melakukan *test-case* pada lingkup tempat penelitian (Krysik, 2023). Pengujian yang tak kalah pentingnya adalah pengujian unit. Pengujian unit digunakan untuk menguji setiap modul untuk menjamin setiap modul menjalankan fungsinya dengan baik. Ada 2 metode untuk melakukan unit testing, yaitu :

1. *White box testing*

White box testing merupakan metode perancangan *test case* yang mengunakan struktural untuk mendapatkan *test case*, test ini digunakan untuk meramal cara kerja perangkat lunak secara rinci kepada *logic path* (jalur logika), perangkat lunak di tes dengan kondisi dan perulangan secara fisik (Setiawan, 2021).

2. *Black box testing*

Black box testing dilakukan dengan menguji aplikasi yang telah dibangun untuk mengamati hasil input dan output dari *software* tanpa mengetahui struktur kode dari perangkat lunak (Setiawan, 2021). Pengujian ini dilakukan agar dapat mengetahui bahwa aplikasi tersebut sudah berfungsi dengan baik.

BAB IV

HASIL DAN PEMBAHASAN

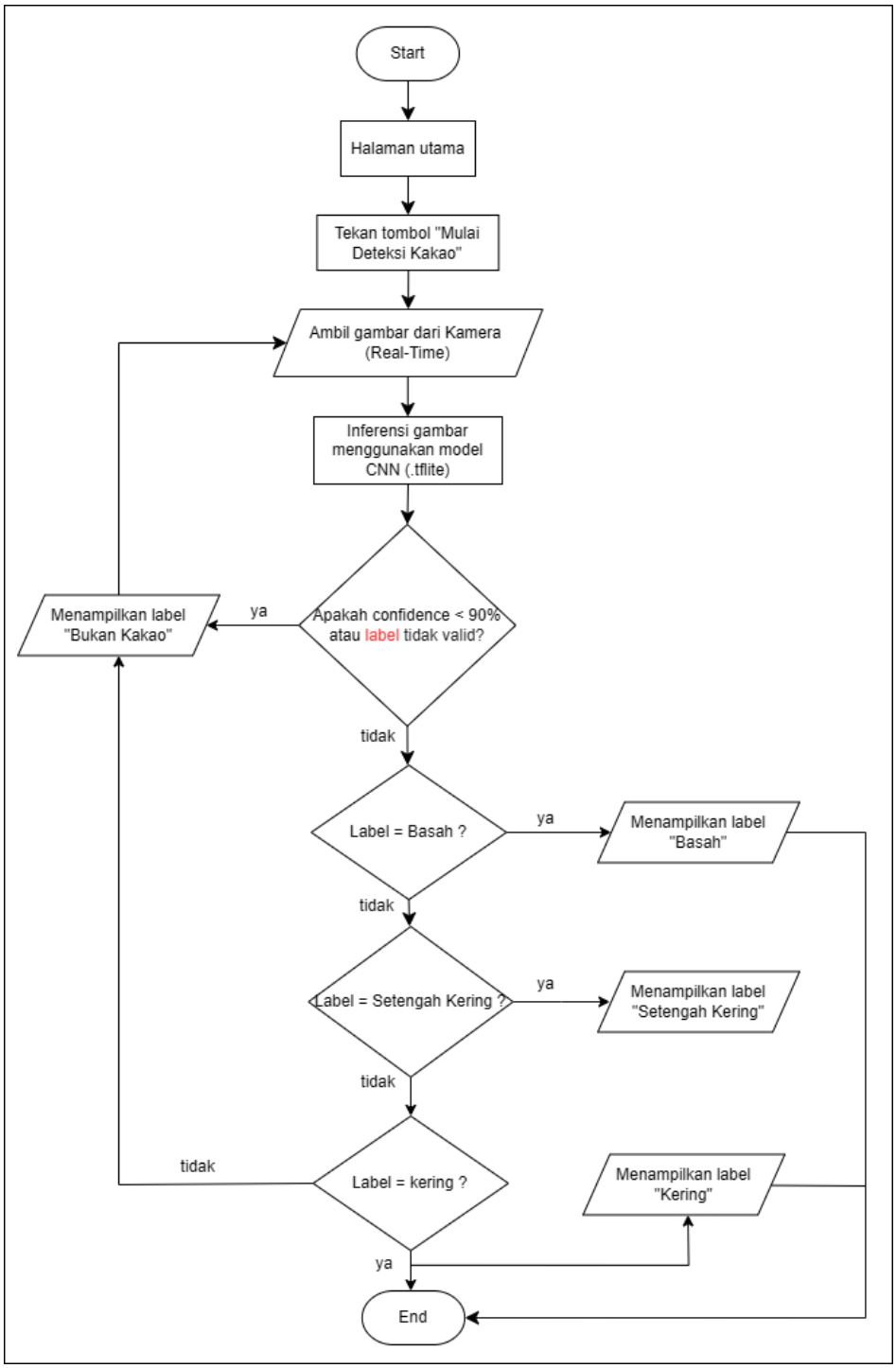
A. Analisis Aliran Data UML

Aliran data dalam UML merujuk pada proses perpindahan informasi atau data antar elemen sistem, seperti komponen, objek, atau aktivitas, meskipun istilah "aliran data" tidak secara eksplisit digunakan sebagaimana pada Data Flow Diagram (DFD). UML lebih menekankan pada pemodelan struktur dan perilaku sistem melalui berbagai jenis diagram, namun konsep aliran data tetap tergambarkan secara implisit melalui interaksi, hubungan, dan pesan antar elemen sistem.

Beberapa diagram UML yang merepresentasikan aliran data secara tidak langsung meliputi Use Case Diagram, Sequence Diagram, Activity Diagram, dan Communication Diagram. Dalam Sequence Diagram, aliran data terwujud melalui pesan yang dikirim antara objek seiring waktu, menunjukkan urutan interaksi antar komponen. Activity Diagram menggambarkan aliran kontrol dan data dalam bentuk aktivitas dan object flow, yang mengilustrasikan bagaimana data berpindah dari satu aktivitas ke aktivitas lainnya.

1. *Flowchart*

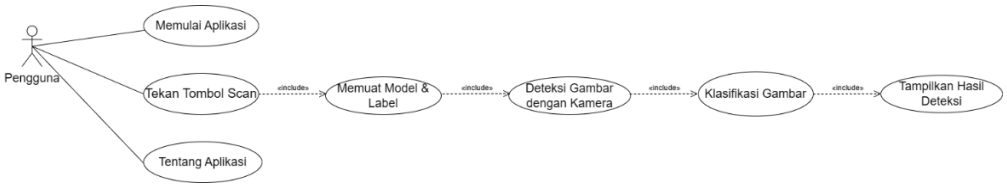
Alur proses jalannya aplikasi digambarkan dengan *flowchart* pada gambar 4.1



Gambar 4. 1 *Flowchart* alur kerja aplikasi

2. *Use case diagram*

Berikut *use case diagram* aplikasi digambarkan pada gambar 4.2



Gambar 4.2 *Use case diagram*

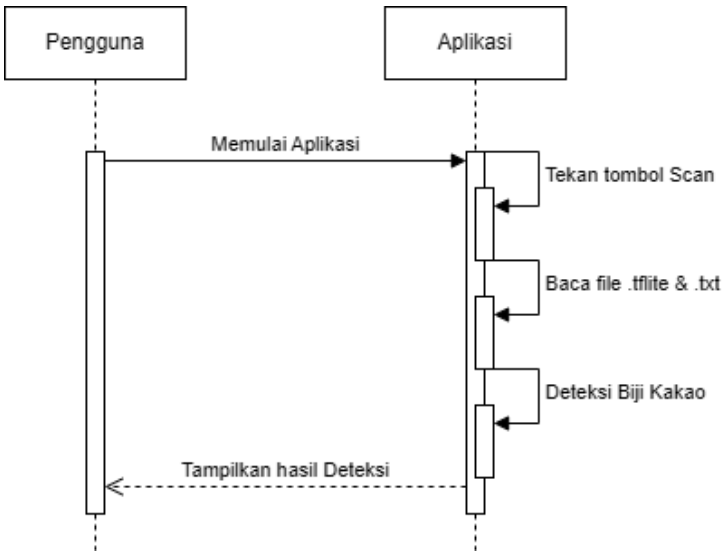
Penjelasan *use case diagram* gambar 4.2 dapat dilihat pada tabel 4.1

Tabel 4.1 *Use case* skenario *user*

Nama Use Case	Deskripsi
Memulai Aplikasi	User membuka dan menjalankan aplikasi untuk memulai proses deteksi.
Tekan Tombol Scan	User menekan tombol scan untuk memulai proses pemindaian gambar biji kakao.
Memuat Model & Label	Aplikasi memuat model CNN dan label klasifikasi yang telah dilatih sebelumnya.
Deteksi Gambar Dengan Kamera	Aplikasi mengakses kamera dan mengambil gambar biji kakao untuk diproses.
Klasifikasi Gambar	Gambar hasil tangkapan kamera diklasifikasikan berdasarkan tingkat kekeringan menggunakan model CNN.
Tampilkan Hasil Deteksi	Aplikasi menampilkan hasil klasifikasi berupa tingkat kekeringan biji kakao.
Tentang Aplikasi	User dapat melihat tentang aplikasi

3. *Sequence Diagram*

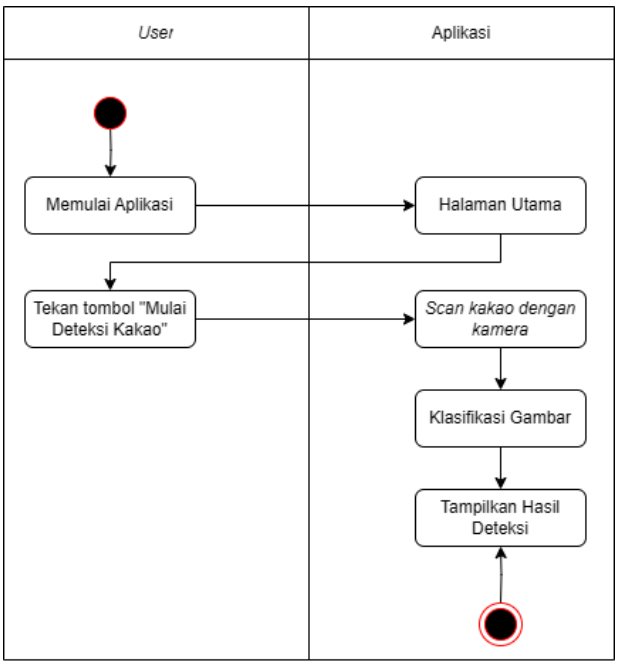
Berikut *sequence diagram* digambarkan pada gambar 4.3



Gambar 4.3 *Sequence diagram user*

4. *Activity Diagram*

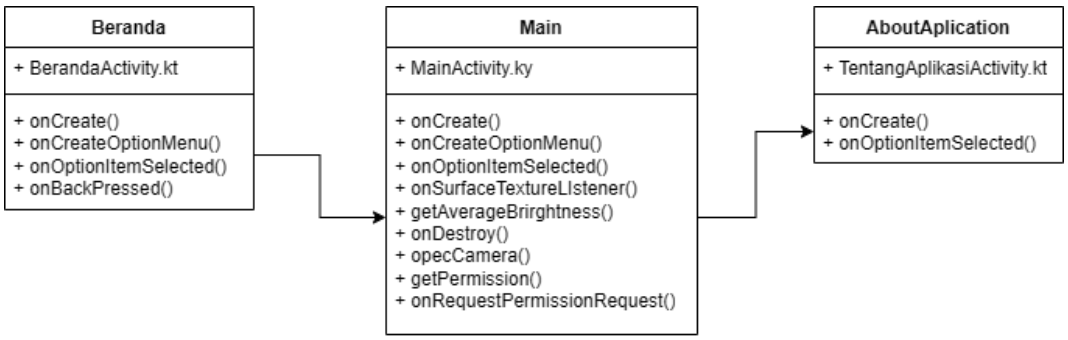
Berikut *activity diagram* digambarkan pada gambar 4.4



Gambar 4.4 *Activity diagram user*

5. *Class Diagram*

Berikut activity diagram digambarkan pada gambar 4.5



Gambar 4.5 *Class Diagram*

Berikut penjelasan class diagram pada gambar 4.5

1. BerandaActivity.kt

   a. Keterangan:

   Kelas ini merepresentasikan halaman utama (beranda) aplikasi yang kemungkinan merupakan titik awal pengguna ketika membuka aplikasi.

   b. Metode:

   1) + onCreate() → Inisialisasi awal saat activity pertama kali dijalankan. Digunakan untuk set layout, toolbar, dan navigasi.
   2) + onCreateOptionsMenu() → Digunakan untuk menampilkan opsi menu di action bar (biasanya ikon titik tiga di pojok kanan atas).
   3) + onOptionsItemSelected() → Menangani aksi ketika salah satu item menu dipilih. Bisa digunakan untuk berpindah ke halaman lain seperti MainActivity atau TentangAplikasiActivity.
   4) + onBackPressed() → Menangani ketika tombol “kembali” ditekan. Bisa digunakan untuk menampilkan dialog konfirmasi keluar.

2. MainActivity.kt

   a. Keterangan:

      Kelas utama untuk deteksi buah kakao menggunakan kamera dan model TFLite. Di sinilah proses real-time classification terjadi.

   b. Metode:

      1) + onCreate() → Setup awal untuk:

         a) Kamera

         b) Toolbar

         c) Model TensorFlow Lite

         d) Listener TextureView

         e) Label dataset

      2) + onCreateOptionsMenu() → Menampilkan menu opsi (seperti tombol info “Tentang Aplikasi”).

      3) + onOptionsItemSelected() → Menangani aksi ketika menu diklik, misalnya navigasi ke TentangAplikasiActivity.

      4) + onSurfaceTextureListener() → Listener untuk TextureView yang memantau perubahan permukaan kamera. Proses klasifikasi dan penampilan hasil dilakukan di sini.

      5) + getAverageBrightness() → Fungsi tambahan untuk menghitung rata-rata kecerahan gambar (digunakan untuk menghindari gambar terlalu putih yang bisa mengganggu deteksi).

      6) + onDestroy() → Menutup model TFLite untuk menghindari memory leak saat activity dihancurkan.

7) + openCamera() → Membuka kamera menggunakan API Camera2, menyiapkan preview ke TextureView.

8) + getPermission() → Meminta izin kamera dari pengguna jika belum diberikan.

9) + onRequestPermissionsResult() → Callback setelah pengguna menjawab permintaan izin, untuk melanjutkan pembukaan kamera jika disetujui.

3. TentangAplikasiActivity.kt (AboutApplication)

   a. Keterangan:

      Kelas ini menampilkan informasi tentang aplikasi (misalnya nama pembuat, versi, deskripsi aplikasi).

   b. Metode:

      1) + onCreate() → Menyiapkan tampilan layout dan toolbar halaman "Tentang Aplikasi".

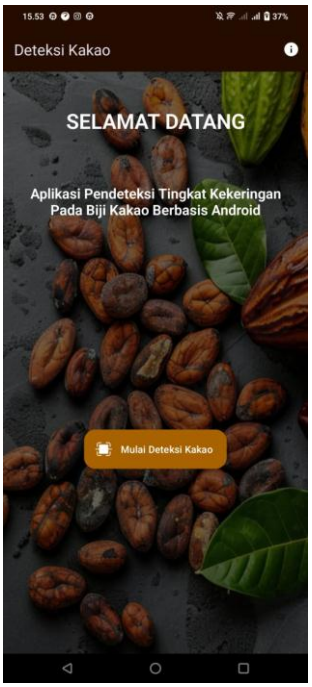
      2) + onOptionsItemSelected() → Menangani ketika tombol back/home (panah kiri di toolbar) ditekan agar kembali ke halaman sebelumnya.

B. Desain Sistem

1. *User*

a. Tampilan Halaman Beranda

Tampilan halaman beranda aplikasi, yang sering juga disebut "home screen" atau "layar beranda", adalah titik awal yang digunakan pengguna untuk mengakses berbagai fitur dan aplikasi di perangkat mereka. Layar ini biasanya menampilkan ikon aplikasi, widget, dan informasi penting lainnya, seperti waktu, notifikasi, dan status baterai.



Gambar 4.6 Tampilan Halaman beranda

Berikut source code untuk tampilan halaman beranda:

```
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.view.Menu
import android.view.MenuItem
import android.widget.Toast
```

```
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.core.content.ContextCompat
import
com.google.android.material.floatingactionbutton.ExtendedFloatingActionB
utton
@Suppress("DEPRECATION")
class BerandaActivity : AppCompatActivity() {
   private lateinit var mulaiscan: ExtendedFloatingActionButton
   private var klikduakali = false
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_beranda)
      mulaiscan = findViewById(R.id.btnscan)
      mulaiscan.setOnClickListener {
         startActivity(Intent(this, MainActivity::class.java))
      }
      window.statusBarColor = ContextCompat.getColor(this,
R.color.warna1)
      val toolbar: Toolbar = findViewById(R.id.toolbar)
      toolbar.title = "Deteksi Kakao"
      setSupportActionBar(toolbar)
   }
   override fun onCreateOptionsMenu(menu: Menu) : Boolean {
      menuInflater.inflate(R.menu.menu_atas, menu)
      return true
   }
   override fun onOptionsItemSelected(item: MenuItem): Boolean {
      return when (item.itemId) {
         R.id.action_tentang_aplikasi -> {
            startActivity(Intent(this, TentangAplikasiActivity::class.java))
            true
         }
         else -> super.onOptionsItemSelected(item)
      }
   }
   @Deprecated("This method has been deprecated in favor of using the\n
{@link OnBackPressedDispatcher} via {@link
#getOnBackPressedDispatcher()}.\n      The OnBackPressedDispatcher
controls how back button events are dispatched\n      to one or more {@link
```

```
OnBackPressedCallback} objects.")
    override fun onBackPressed() {
        if (klikduakali) {
            super.onBackPressed()
            finish()
            return
        }
        klikduakali = true
        Toast.makeText(this, "Tekan sekali lagi untuk keluar",
Toast.LENGTH_SHORT).show()
        Handler(Looper.getMainLooper()).postDelayed({ klikduakali = false },
2000)
    }
}
```

b. Fitur *Scan*

Fitur *Scan* pada aplikasi ini berfungsi untuk mendeteksi tingkat kekeringan biji kakao berdasarkan gambar yang diambil oleh pengguna. Setelah gambar dipindai, sistem akan melakukan analisis menggunakan model CNN *(Convolutional Neural Network)* yang telah dilatih sebelumnya.



Gambar 4.7 Deteksi kakao basah



Gambar 4.8 Deteksi kakao setengah kering



Gambar 4.9 Deteksi kakao kering

Berikut source code untuk fitur scan:

```
import android.Manifest
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.*
import android.hardware.camera2.*
import android.os.*
import android.view.*
import android.widget.ImageView
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.core.app.ActivityCompat
import androidx.core.content.ContextCompat
import com.kelas_b.a221280027.ml.CacaoModelWithMetadata
import org.tensorflow.lite.support.common.FileUtil
import org.tensorflow.lite.support.image.ImageProcessor
import org.tensorflow.lite.support.image.TensorImage
import org.tensorflow.lite.support.image.ops.ResizeOp
@Suppress("DEPRECATION")
class MainActivity : AppCompatActivity() {
    private lateinit var labels: List<String>
    private val paint = Paint().apply {
        textAlign = Paint.Align.LEFT
        isAntiAlias = true
    }
    private lateinit var imageProcessor: ImageProcessor
    private lateinit var imageView: ImageView
    private lateinit var cameraDevice: CameraDevice
    private lateinit var handler: Handler
    private lateinit var cameraManager: CameraManager
    private lateinit var textureView: TextureView
    private lateinit var model: CacaoModelWithMetadata
    private var lastDetectedLabel: String = ""
    private var detectionCount: Int = 0
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        labels = FileUtil.loadLabels(this, "labels.txt") // pastikan "Bukan
```

```
Kakao" tidak ada di sini
        imageProcessor = ImageProcessor.Builder()
            .add(ResizeOp(300, 300, ResizeOp.ResizeMethod.BILINEAR))
            .build()
        model = CacaoModelWithMetadata.newInstance(this)
        val handlerThread = HandlerThread("videoThread").apply { start() }
        handler = Handler(handlerThread.looper)
        cameraManager = getSystemService(CAMERA_SERVICE) as
CameraManager
        imageView = findViewById(R.id.imageView)
        textureView = findViewById(R.id.textureView)
        textureView.surfaceTextureListener = surfaceTextureListener
        getPermission()
        window.statusBarColor = ContextCompat.getColor(this,
R.color.warna1)
        val toolbar: Toolbar = findViewById(R.id.toolbar)
        toolbar.title = "Deteksi Kakao"
        setSupportActionBar(toolbar)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)
        supportActionBar?.setDisplayShowHomeEnabled(true)
    }
    override fun onCreateOptionsMenu(menu: Menu): Boolean {
        menuInflater.inflate(R.menu.menu_atas, menu)
        return true
    }
    override fun onOptionsItemSelected(item: MenuItem): Boolean {
        return when (item.itemId) {
            R.id.action_tentang_aplikasi -> {
                startActivity(Intent(this, TentangAplikasiActivity::class.java))
                true
            }
            android.R.id.home -> {
                onBackPressed()
                true
            }
            else -> super.onOptionsItemSelected(item)
        }
    }
    private val surfaceTextureListener = object :
TextureView.SurfaceTextureListener {
```

```
        override fun onSurfaceTextureAvailable(surface: SurfaceTexture,
width: Int, height: Int) {
            if (ContextCompat.checkSelfPermission(this@MainActivity,
Manifest.permission.CAMERA)
                == PackageManager.PERMISSION_GRANTED
            ) {
                openCamera()
            }
        }
        override fun onSurfaceTextureSizeChanged(surface: SurfaceTexture,
width: Int, height: Int) {}
        override fun onSurfaceTextureDestroyed(surface: SurfaceTexture):
Boolean = false
        override fun onSurfaceTextureUpdated(surface: SurfaceTexture) {
            val bmp = textureView.bitmap ?: return
            val avgBrightness = bmp.getAverageBrightness()
            if (avgBrightness > 230) return
            val image =
imageProcessor.process(TensorImage.fromBitmap(bmp))
            val outputs = model.process(image)
            val probability = outputs.probabilityAsCategoryList
            val detected = probability.maxByOrNull { it.score }
            val allowedLabels = listOf("Basah", "Kering", "Setengah Kering")
            val threshold = 0.90f
            val isValid = detected != null &&
                    detected.score > threshold &&
                    detected.label in allowedLabels
            if (isValid) {
                if (detected!!.label == lastDetectedLabel) {
                    detectionCount++
                } else {
                    detectionCount = 0
                }
                lastDetectedLabel = detected.label
                if (detectionCount >= 3) {
                    val mutable = bmp.copy(Bitmap.Config.ARGB_8888, true)
                    val canvas = Canvas(mutable)
                    val h = mutable.height
                    val w = mutable.width
                    paint.textSize = h / 20f
```

```
                paint.strokeWidth = h / 85f
                paint.color = Color.GREEN
                paint.style = Paint.Style.STROKE
                canvas.drawRect(RectF(0f, 0f, w.toFloat(), h.toFloat()), paint)
                paint.style = Paint.Style.FILL
                canvas.drawText(detected.label, (w -
paint.measureText(detected.label)) / 2f, 100f, paint)
                val scoreText = "Confidence: %.2f%%".format(detected.score *
100)
                paint.textSize = h / 35f
                canvas.drawText(scoreText, (w - paint.measureText(scoreText))
/ 2f, 160f, paint)
                imageView.setImageBitmap(mutable)
                return
            }
        } else {
            detectionCount = 0
            lastDetectedLabel = ""
            // tampilkan info "Bukan Kakao"
            val mutable = bmp.copy(Bitmap.Config.ARGB_8888, true)
            val canvas = Canvas(mutable)
            val h = mutable.height
            val w = mutable.width
            paint.textSize = h / 20f
            paint.color = Color.RED
            canvas.drawText("Bukan Kakao", (w - paint.measureText("Bukan
Kakao")) / 2f, 100f, paint)
            imageView.setImageBitmap(mutable)
            return
        }
        imageView.setImageBitmap(bmp)
    }
}
fun Bitmap.getAverageBrightness(): Int {
    var total = 0L
    val pixels = IntArray(width * height)
    getPixels(pixels, 0, width, 0, 0, width, height)
    for (pixel in pixels) {
        val r = Color.red(pixel)
        val g = Color.green(pixel)
```

```
            val b = Color.blue(pixel)
            total += (r + g + b) / 3
        }
        return (total / pixels.size).toInt()
    }
    override fun onDestroy() {
        super.onDestroy()
        model.close()
    }
    private fun openCamera() {
        val cameraId = cameraManager.cameraIdList[0]
        if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) return
        cameraManager.openCamera(cameraId, object :
CameraDevice.StateCallback() {
            override fun onOpened(device: CameraDevice) {
                cameraDevice = device
                val surface = Surface(textureView.surfaceTexture)
                val requestBuilder =
cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_PREVIE
W)
                requestBuilder.addTarget(surface)
                cameraDevice.createCaptureSession(listOf(surface), object :
CameraCaptureSession.StateCallback() {
                    override fun onConfigured(session: CameraCaptureSession) {
                        session.setRepeatingRequest(requestBuilder.build(), null,
handler)
                    }
                    override fun onConfigureFailed(session:
CameraCaptureSession) {}
                }, handler)
            }
            override fun onDisconnected(device: CameraDevice) {}
            override fun onError(device: CameraDevice, error: Int) {}
        }, handler)
    }
    private fun getPermission() {
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA)
```

```
            != PackageManager.PERMISSION_GRANTED
        ) {
            requestPermissions(arrayOf(Manifest.permission.CAMERA), 101)
        }
    }
    override fun onRequestPermissionsResult(requestCode: Int, permissions:
Array<out String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
        if (requestCode == 101 && grantResults.isNotEmpty() &&
grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            if (textureView.isAvailable) openCamera()
            else textureView.surfaceTextureListener = surfaceTextureListener
        } else {
            Toast.makeText(this, "Kamera diperlukan untuk aplikasi ini",
Toast.LENGTH_LONG).show()
            finish()
        }
    }
}
```

Nilai confidence (tingkat keyakinan) mewakili seberapa yakin model terhadap prediksinya untuk sebuah label tertentu. Perhitungan confidence dalam program dilakukan dengan mengambil skor tertinggi dari seluruh label yang diprediksi oleh model. Hal ini direpresentasikan dalam kode berikut:

Setelah gambar diproses oleh model:

```
val outputs = model.process(image)
val probability = outputs.probabilityAsCategoryList
```

Model akan menghasilkan daftar label dan skornya, misalnya:

Label	**Skor**
Basah	0.45678
Setengah Kering	0.9130

Label	**Skor**
Kering	0.87214

Ambil nilai confidence tertinggi:

```
val detected = probability.maxByOrNull { it.score }
```

Hasilnya:

- detected.label = "Setengah Kering"
- detected.score = 0.9130

Cek apakah confidence > threshold (0.90):

```
val isValid = detected != null &&
        detected.score > threshold &&
        detected.label in allowedLabels
```

Karena 0.9130 > 0.90 → valid.

Ubah ke persen dengan dua angka dibelakang koma & tampilkan:

```
val scoreText = "Confidence: %.2f%%".format(detected.score * 100)
```

Perhitungan:

- 0.9130 × 100 = 91.30.
- Format “%.2f" → artinya format jadi dua angka di belakang koma → 91.30
- Ditampilkan di layar:

  ```
  Confidence: 91.30%
  ```

c. Halaman Tentang Aplikasi



Gambar 4.10 Halaman tentang aplikasi

Berikut source code untuk halaman tentang aplikasi:

```
import android.os.Bundle
import android.view.MenuItem
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.widget.Toolbar
import androidx.core.content.ContextCompat
class TentangAplikasiActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_tentang_aplikasi)
      window.statusBarColor = ContextCompat.getColor(this,
R.color.warna1)
      val toolbar: Toolbar = findViewById(R.id.toolbar)
      toolbar.title = "Tentang Aplikasi"
      setSupportActionBar(toolbar)
      supportActionBar?.setDisplayHomeAsUpEnabled(true)
      supportActionBar?.setDisplayShowHomeEnabled(true)
   }
   override fun onOptionsItemSelected(item: MenuItem): Boolean {
```

```
        return when (item.itemId) {
            android.R.id.home -> {
                onBackPressed()
                true
            }
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

C. Pengujian Sistem

1. *Blackbox testing*

Pengujian *Black box testing* dilakukan agar dapat mengetahui bahwa aplikasi tersebut sudah berfungsi dengan baik.

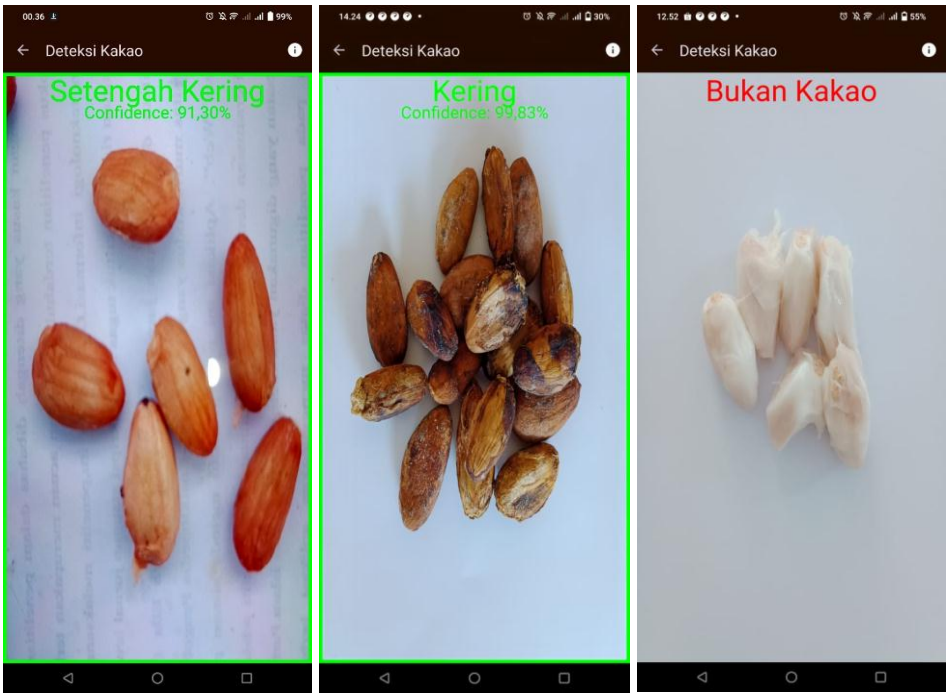
a. *Blackbox testing* tampilan halaman utama *(User)*

Tabel 4.2 Tampilan halaman utama *user*

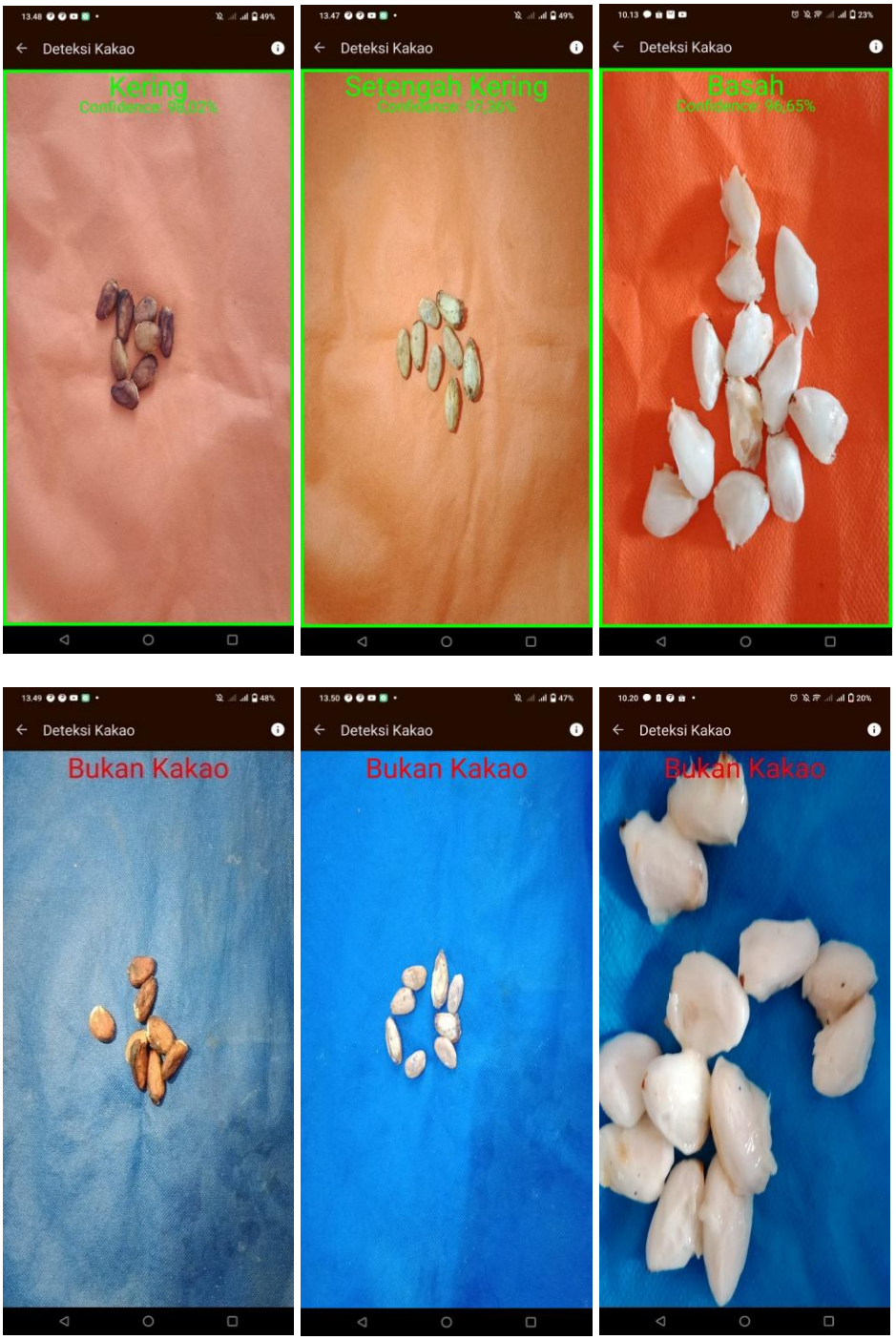
Test Faktor	Hasil	Keterangan
Masuk ke halaman utama	✓	Sistem akan menampilkan halaman utama
Screenshot		
		

b. *Blackbox testing scan* kakao *(User)*

Tabel 4.3 Pengujian *scan* kakao berdasarkan latar menggunakan kamera 108 MP dengan jarak 25 cm

Kategori Kakao	Latar putih	Latar biru	Latar orange	Keterangan
Basah	✗	✗	✓	Sistem berhasil menampilkan label basah pada latar orange, sedangkan latar putih dan biru gagal.
Setengah Kering	✓	✗	✓	Sistem berhasil menampilkan label setengah kering pada latar putih dan orange, sedangkan latar biru gagal.
Kering	✓	✗	✓	Sistem berhasil menampilkan label kering pada latar putih dan orange, sedangkan latar biru gagal
Screenshot				
				

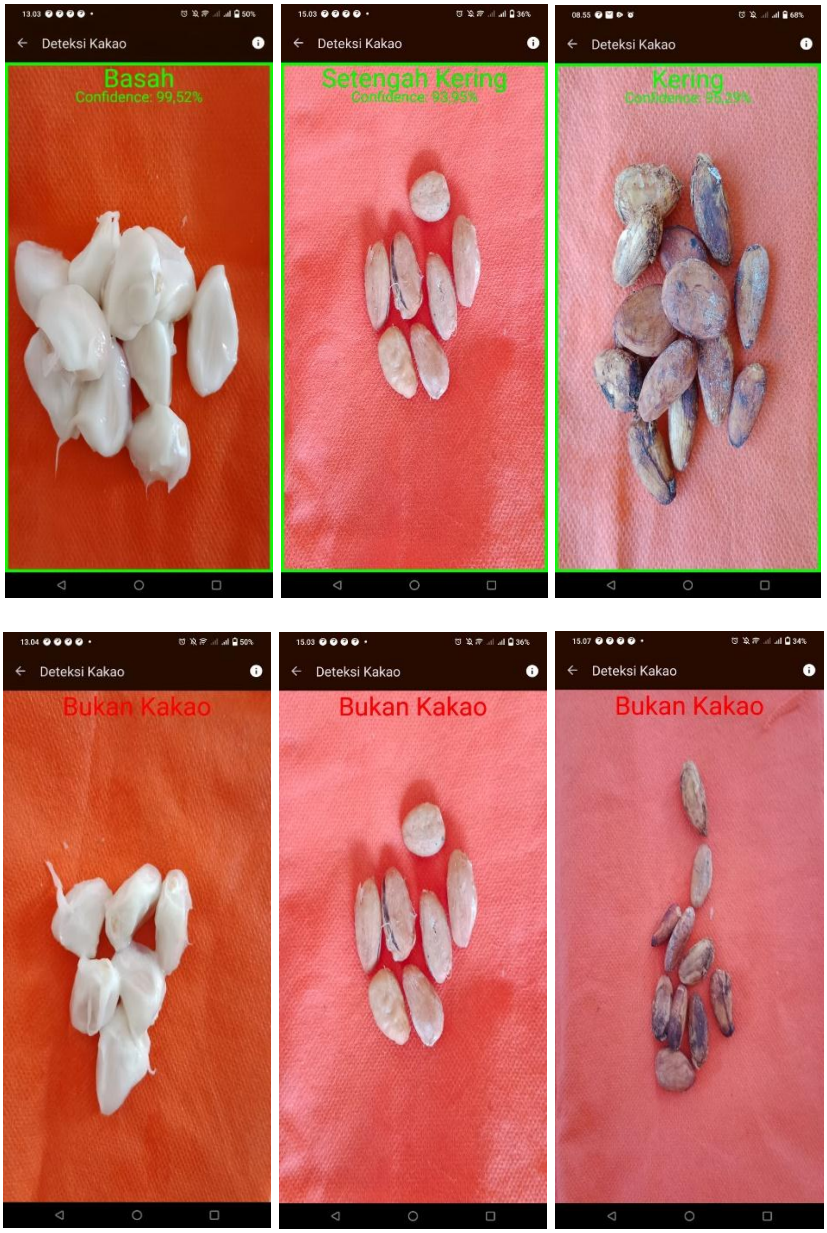
Lanjutan Tabel 4.3

Screenshot


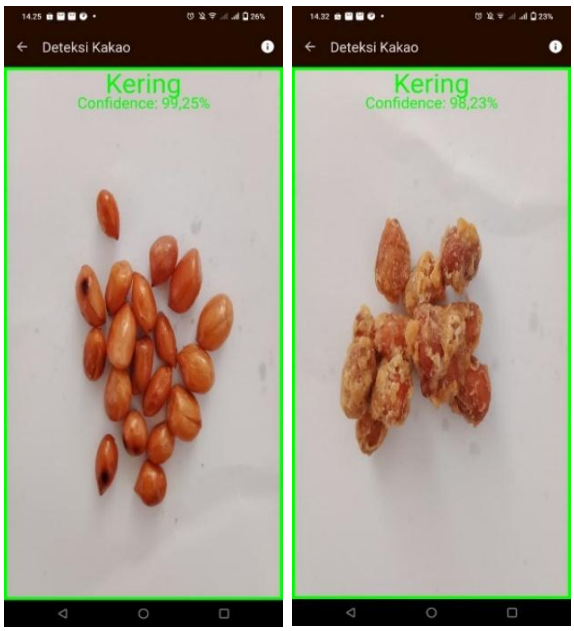
Tabel 4.4 Pengujian *scan* kakao berdasarkan resolusi kamera (MP) dan jarak (cm) menggunakan latar orange

Resolusi Kamera (MP)	Kategori Kakao	Jarak (cm)					Keterangan
		6	7	10	15	25	
13 MP	Basah	✗	✗	✓	✗	✗	Sistem berhasil mendeteksi semua kategori pada jarak >7-<15cm, dan gagal pada jarak terdekat <8 cm dan jarak terjauh >14 cm.
	Setengah Kering	✗	✗	✓	✗	✗	
	Kering	✗	✗	✓	✗	✗	
50 MP	Basah	✗	✓	✓	✓	✗	Sistem berhasil mendeteksi semua kategori kakao pada jarak 7 – 20 cm dan gagal pada jarak terdekat >7 cm dan jarak terjauh <20 cm.
	Setengah Kering	✗	✓	✓	✓	✗	
	Kering	✗	✓	✓	✓	✗	
108 MP	Basah	✗	✓	✓	✓	✓	Sistem berhasil mendeteksi semua kategori pada jarak 7 – 25 cm dan gagal pada jarak terdekat >7 dan jarak terjauh <25 cm.
	Setengah Kering	✗	✓	✓	✓	✓	
	Kering	✗	✓	✓	✓	✓	

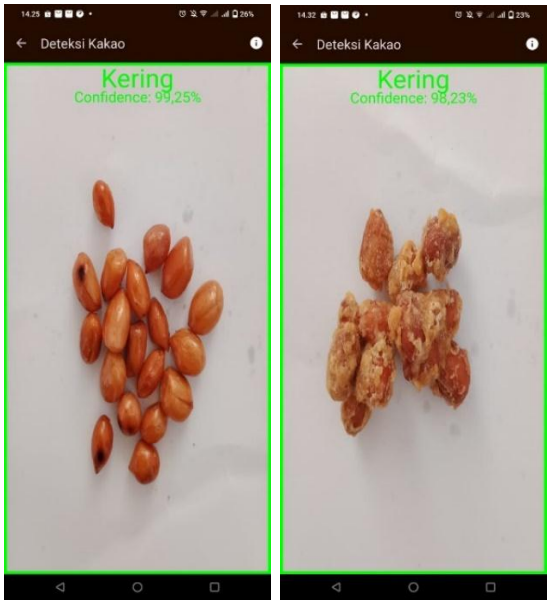
Lanjutan Tabel 4.4

Screenshot


Tabel 4.5 Uji deteksi terhadap objek non-biji kakao

Objek	**Hasil**	**Keterangan**
Kacang Tanah	✗	Sistem gagal mendeteksi karena menampilkan label “Kering” pada objek kacang tanah.
Kue Kacang	✗	Sistem gagal mendeteksi karena menampilkan label “Kering” pada objek Kue Kacang.
Batu	✗	Sistem gagal mendeteksi karena menampilkan label “Kering” pada objek Batu.
Tisu	✗	Sistem gagal mendeteksi karena menampilkan label “Basah” pada objek Tisu.
Screenshot		
		

Lanjutan Tabel 4.5

Screenshot


2. *Whitebox testing*

Whitebox testing bertujuan untuk menguji logika internal dari program secara langsung. Dalam penelitian ini, pengujian dilakukan pada komponen utama aplikasi yang berperan dalam proses klasifikasi gambar, yakni:

a. Fungsi pemrosesan citra kamera

```
1  fun getCapturedImage(): Bitmap {
2      val bitmap = textureView.bitmap
3      return Bitmap.createScaledBitmap(bitmap, 224, 224, true)
4  }
```

Gambar 4.11 Fungsi pemrosesan citra kamera

1) Jalur logika yang diuji:
    a) textureView.bitmap tersedia → gambar berhasil diproses dan diskalakan ke ukuran input model (224x224).
    b) Jika null → penanganan error perlu ditambahkan (pengujian dilakukan untuk mendeteksi potensi null).
2) Hasil:
    a) Gambar berhasil diambil dan dikonversi ke ukuran yang sesuai untuk inferensi CNN.
    b) Fungsi ini penting untuk memastikan input ke model sudah valid dan konsisten.

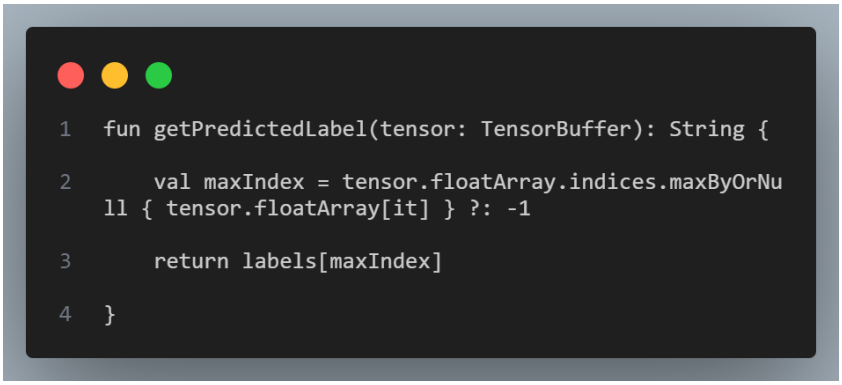
b. Fungsi pemanggilan model *TensorFlow Lite*

```
1  val model = KakaoModel.newInstance(context)
2
3  val inputFeature = TensorBuffer.createFixedSize(int
   ArrayOf(1, 224, 224, 3), DataType.FLOAT32)
4  inputFeature.loadBuffer(convertBitmapToByteBuffer(b
   itmap))
5
6  val outputs = model.process(inputFeature)
7  val outputFeature = outputs.outputFeature0AsTensorB
   uffer
8
```

Gambar 4.12 Fungsi pemanggilan model *TensorFlow Lite*

1) Jalur logika yang diuji:
    a) Model berhasil dipanggil dan diinisialisasi.
    b) Data input berhasil dikonversi ke TensorBuffer.
    c) Fungsi process() mengembalikan hasil prediksi berupa tensor berisi skor per kelas.
2) Hasil:
    a) Fungsi berjalan sesuai alur tanpa error selama model dan input valid.
    b) Output berhasil diperoleh dalam bentuk tensor array.

c. Fungsi interpretasi hasil klasifikasi



Gambar 4.13 Fungsi interpretasi hasil klasifikasi

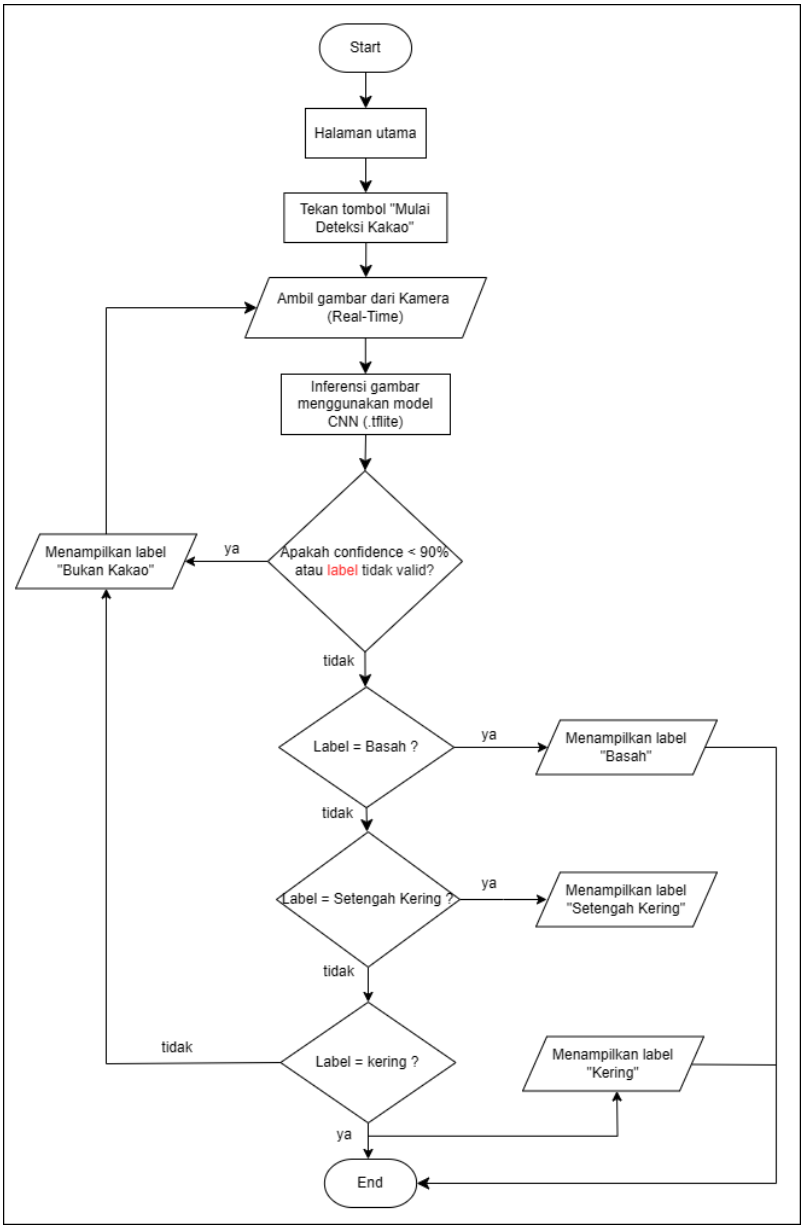
1) Jalur logika yang diuji:
    a) Tensor output berisi array nilai probabilitas → indeks dengan nilai tertinggi diambil.
    b) maxIndex dicocokkan ke dalam list labels[] untuk mendapatkan label kelas akhir.

2) Hasil:

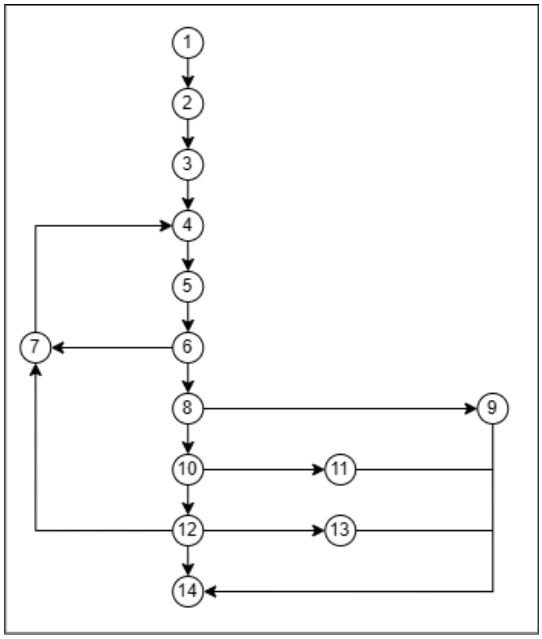
a) Fungsi berhasil menentukan label klasifikasi seperti “Basah”, “Setengah Kering”, atau “Kering”.

b) Penanganan kondisi kosong atau error juga perlu ditambahkan jika model rusak

d. Berikut alur kerja flowchart dan flowgraph aplikasi ini:



Gambar 4.14 *Flowchart* alur kerja aplikasi



Gambar 4.15 *Flowgraph* alur kerja aplikasi

Berdasarkan *flowgraph* pada gambar 4.15, maka dilakukan proses perhitungan sebagai berikut:

a. Menghitung Cyclomatic Complecity V(G) dari Edge dan Node :

   V(G) = E – N + 2

   N (*node*) = 14

   E (*edge*) = 16

   P (*predikat node*) = 4

   Penyelesaian :

   V(G) = E – N + 2

   = 17 – 14 + 2

   = 5

   Predikat (P) = P + 1

   = 4 + 1

   = 5

b. Berdasarkan perhitungan *Cyclomatic Complecity* dari *Flowgraph* diatas memiliki *Region* = 5

c. *Independent Path* pada *Flowgraph* diatas adalah :

Path 1 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 4 – 5 – 6 – 8 – 10 – 12 - 14

Path 2 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 9 – 14

Path 3 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 10 – 11 – 14

Path 4 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 10 – 12 – 7 – 4 – 5 - 6 – 8 – 10 – 12 - 14

Path 5 = 1 – 2 – 3 – 4 – 5 – 6 – 8 – 10 – 12 – 13 – 14

d. Grafik Matriks

Tabel 4.6 Grafik Matriks

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	E - 1
1		1													1 – 1 = 0
2			1												1 – 1 = 0
3				1											1 – 1 = 0
4					1										1 – 1 = 0
5						1									1 – 1 = 0
6							1	1							2 – 1 = 1
7				1				1							2 – 1 = 1
8									1	1					2 – 1 = 1
9										1				1	2 – 1 = 1
10											1	1			2 – 1 = 1
11														1	1 – 1 = 0
12							1						1	1	3 – 1 = 2

13														1	1 – 1 = 0
14															
SUM (E + 1)															7 + 1 = 8

BAB V

PENUTUP

A. Kesimpulan

Berdasarkan hasil penelitian dan pengujian terhadap aplikasi pendeteksi tingkat kekeringan pada biji kakao berbasis Android, maka dapat disimpulkan:

1. Aplikasi berhasil dirancang dan dibangun menggunakan teknologi TensorFlow dan CNN (Convolutional Neural Network) yang mampu mendeteksi tingkat kekeringan biji kakao ke dalam tiga kategori: *basah*, *setengah kering*, dan *kering*.
2. Model CNN yang telah dilatih dengan dataset gambar biji kakao mampu melakukan klasifikasi dengan tingkat akurasi yang layak, bergantung pada kualitas citra yang diambil dari kamera pengguna.
3. Proses integrasi model *TensorFlow Lite* ke dalam aplikasi Android menggunakan *Kotlin* berhasil dilakukan, sehingga pengguna dapat melakukan deteksi secara langsung melalui kamera *smartphone*.
4. Pengujian sistem melalui metode *blackbox* dan *whitebox* menunjukkan bahwa aplikasi berjalan sesuai dengan fungsinya, baik dalam proses *input*, pemrosesan, maupun *output* hasil klasifikasi.

5. Kesimpulan hasil pengujian scan kakao:
    a. Pengaruh Resolusi Kamera:
        1) Kamera 13 MP hanya mampu mendeteksi secara terbatas pada jarak ideal 10 cm, dan gagal di jarak terlalu dekat (<8 cm) maupun terlalu jauh (>14 cm).
        2) Kamera 50 MP dapat mendeteksi semua kategori pada rentang 7–20 cm, tetapi gagal di jarak terdekat dan terjauh.
        3) Kamera 108 MP memberikan performa terbaik, mampu mendeteksi semua kategori kakao pada jarak 7–25 cm, serta stabil dalam berbagai kondisi latar.
    b. Pengaruh Jarak Kamera:
        1) Jarak ideal untuk semua kamera berada pada kisaran 10–15 cm.
        2) Jarak terdekat (<7 cm) dan terjauh (>20 cm) sering menyebabkan sistem gagal mendeteksi.
        3) Kamera 108 MP tetap mampu mendeteksi dengan baik bahkan hingga 25 cm, menunjukkan stabilitas pada jarak jauh.
    c. Pengaruh Warna Latar:
        1) Latar orange memberikan hasil deteksi terbaik untuk semua kategori kakao.
        2) Latar putih hanya efektif untuk kategori setengah kering dan kering, terutama dengan kamera 108 MP.
        3) Latar biru adalah yang paling tidak efektif; semua kategori gagal terdeteksi meskipun menggunakan kamera resolusi tinggi.

d. Deteksi Berdasarkan Kategori Kakao:

    1) Kategori basah paling sulit terdeteksi, terutama pada kamera resolusi rendah dan latar putih/biru.
    2) Kategori setengah kering dan kering lebih mudah dikenali, bahkan dengan kamera 13 MP, selama berada pada jarak optimal.
    3) Kategori kering merupakan kategori yang paling konsisten terdeteksi dalam semua pengujian.

6. Berdasarkan pengujian akurasi aplikasi yang menampilkan hasil uji deteksi terhadap objek non-biji kakao (kacang tanah, kue kacang, batu, dan tisu) dapat disimpulkan bahwa aplikasi masih keliru mendeteksi objek yang bukan biji kakao. Sistem justru memberikan label "Kering" atau "Basah" pada objek lain seperti kacang tanah, kue kacang, batu, dan tisu. Hal ini menunjukkan bahwa aplikasi hanya mampu mengenali objek yang mirip dengan data pelatihan biji kakao dan belum dapat membedakan dengan baik antara biji kakao dan benda lain di luar dataset.
7. Berdasarkan pengujian yang telah dilakukan menunjukkan bahwa implementasi CNN melalui *TensorFlow* cukup akurat mendeteksi tingkat kekeringan pada biji kakao sesuai dataset pelatihan. Namun, model kurang akurat dalam membedakan biji kakao dengan objek lain di luar dataset.

B. Saran

1. Peningkatan kualitas dataset: Untuk memperoleh akurasi yang lebih tinggi, disarankan agar dataset gambar diperluas dengan berbagai variasi pencahayaan, sudut pandang, dan kondisi biji kakao yang lebih beragam.
2. Implementasi real-time detection: Aplikasi ke depan dapat dikembangkan agar mendukung deteksi secara *real-time* melalui video stream, bukan hanya pengambilan gambar statis.
3. Penambahan fitur persentase kekeringan: Sebaiknya aplikasi dilengkapi dengan estimasi tingkat kekeringan dalam bentuk persentase guna memberikan informasi yang lebih detail.
4. Pengembangan UI/UX: Tampilan aplikasi dapat diperbaiki agar lebih ramah pengguna (user-friendly) dan menarik secara visual agar lebih nyaman digunakan oleh kalangan petani maupun pengguna umum.
5. Dukungan multi-platform dan fitur cloud: Untuk pengembangan ke depan, aplikasi dapat ditambahkan fitur cloud storage untuk penyimpanan data hasil klasifikasi serta dikembangkan versi multiplatform seperti untuk iOS atau web.

DAFTAR PUSTAKA

Bulandari, S. (2019). Pengaruh Produksi Kakao Terhadap Pertumbuhan Ekonomi di Kabupaten Kolaka Utara. *Fakultas Ekonomi dan Bisnis Islam Universitas Islam Negeri Alauddin Makassar*, *juli*, 1–85.

Rachmatullah, D., Putri, D. N., Fiki Herianto, & Harini, N. (2021). KARAKTERISTIK BIJI KAKAO (Theobroma cacao L.) HASIL FERMENTASI DENGAN UKURAN WADAH BERBEDA. *VIABEL: Jurnal Ilmiah Ilmu-Ilmu Pertanian*, 15(1), 32–44. https://doi.org/10.35457/viabel.v15i1.1409

Dharmadi, R. (2018). *CNN: Beyond Image Classification*. https://medium.com /nodeflux/cnn-beyond-image-classification-3b9b0af021a9

Dharwiyanti, S., & Wahono, R. S. (2003). Pengantar Unified Modeling LAnguage (UML). *IlmuKomputer.com*, 1–13.

Hasyim, A. H. (2023). *Dasar Pemrograman* (3rd ed.). CV. Bangun Bumitama.

I Wayan Suartika E. P, A. Y. W. dan R. S. (2016). Klasifikasi Citra Menggunakan Convolutional Neural Network (Cnn) pada Caltech 101. *JURNAL TEKNIK ITS*. Vol. 5, No(1): 1–2.

Juansyah, A. (2015). Pembangunan aplikasi child tracker berbasis assisted–global positioning system (a-gps) dengan platform android. *Jurnal Ilmiah Komputer dan Informatika* (KOMPUTA), 1(1): 1-8.

Krysik, A. (2023, Desember 8). *SDLC Guide: Coding and Development Phase in Software Engineering*. From Stratoflow: https://stratoflow.c om/development-phase-sdlc/

Nasha Hikmatia A.E., & Zul, M. I. (2021). Aplikasi Penerjemah Bahasa Isyarat Indonesia menjadi Suara berbasis Android menggunakan Tensorflow. *Jurnal Komputer Terapan*, *7*(1), 74–83. https://doi.org/10.35143/jkt.v7i1.4629

Rahman, S., Sembiring, A., Siregar, D., Khair, H., Gusti Prahmana, I., Puspadini, R., & Zen, M. (2023). Python : Dasar Dan Pemrograman Berorientasi Objek. In *Penerbit Tahta Media*.

Arifianto, R. (2014). *Pengertian Flowchart Dan Jenis – Jenisnya*. https://rahmatarifianto.wordpress.com/2014/11/20/pengertian-flowchart-dan-jenis-jenisnya/

Setiawan, R. (2021, November 20). *White Box Testing untuk Menguji Perangkat Lunak*. From Dicoding.com: https://www. dicoding.com/blog/white-box-testing/

Sumiati, M., Abdillah, R., & Cahyo, A. (2021). *Uml 21*. *11*(2), 79–86.