BAB I
PENDAHULUAN

1.1 Latar Belakang

Buah semangka (Citrullus Vulgaris Schrad) Merupakan buah yang tumbuh merambat berkembang pesat di Indonesia, Indonesia salah satu negara penghasil semangka terbesar di dunia kementerian pertanian menyebutkan bahwa hasil panen semangka memiliki kontribusi produksi nasional sebesar 3,30%. Data tersebut membuktikan jumlah produksi buah semangka di Indonesia cukup besar dan diminati oleh masyarakat.(Putri, 2021).

Semangka merupakan buah yang banyak di konsumsi yang menjadikan semangka menjadi buah yang dicari oleh masyarakat. Kesamaan antara semangka mentah, setengah matang dan matang membuat sulit untuk mengidentifikasi kematangan buah semangka. Karena orang secara subyektif menentukan tingkat kematangan buah semangka, maka penilaian tingkat kematangan buah semangka berbeda-beda pada setiap orang. Proses klasifikasi kematangan buah semangka dapat dilakukan dengan cara mengidentifikasi citra dari warna dan corak buah tersebut. Bentuk dan warna yang mirip membuat buah ini susah dibedakan oleh visualisasi manusia, sehingga sulit bagi konsumen memilih buah semangka yang matang untuk dikonsumsi (M. Zanuroini, 2023).

Mengidentifikasi dan mengklasifikasikan kematangan semangka secara efisien dan akurat adalah tantangan dalam industri pertanian dan pangan. Salah satu solusi menarik adalah penerapan Convolutional Neural Network (CNN), sebuah pendekatan yang menerapkan teknologi deep learning untuk analisis gambar dan pengenalan pola (Rahman Sya'ban et al., 2022).

Untuk mengatasi tantangan ini, pengidentifikasian kematangan semangka dapat dilakukan dengan analisis tekstur kulit dan warna buah berdasarkan citra visual. Pendekatan ini memungkinkan klasifikasi kematangan dilakukan secara lebih akurat dan konsisten. Dengan demikian, pengembangan metode otomatis untuk mengklasifikasikan kematangan semangka tidak hanya bermanfaat bagi petani dan produsen, tetapi juga memberikan dampak positif bagi rantai pasokan dan konsumen akhir. Konsumen akan lebih mudah mendapatkan semangka yang matang sehingga meningkatkan pengalaman konsumsi.

Convolutional Neural Network (CNN) adalah arsitektur jaringan saraf tiruan yang telah menunjukkan keberhasilan dalam analisis gambar. Berbeda dengan pendekatan tradisional, CNN dapat secara otomatis mengekstraksi fitur-fitur penting dari gambar, memungkinkan untuk identifikasi pola yang kompleks CNN telah terbukti mendapatkan hasil yang baik, termasuk untuk mendektesi objek dan pengenalan gambar (Nana et al., 2022).

Proses penerapan CNN untuk mengklasifikasikan kematangan semangka melibatkan beberapa langkah kunci. Pertama, data citra semangka dikumpulkan dan diolah untuk menciptakan dataset pelatihan yang mencakup variasi kematangan.

Langkah pra-pemrosesan mungkin diperlukan untuk memastikan data dalam format yang cocok untuk masukan model CNN. Pelatihan model melibatkan proses belajar di mana model diberi tahu bagaimana mengenali tingkat kematangan semangka berdasarkan dataset pelatihan.

Pada tahap pengujian, model yang telah dilatih diuji pada data pengujian yang tidak pernah dilihat sebelumnya. Performa model dievaluasi menggunakan metode confusion metrik mencari nilai precision, recall, akurasi serta f1 score (Yanto et al., 2021). Hasil pengujian ini akan memberikan pandangan yang jelas tentang sejauh mana model dapat mengklasifikasikan kematangan semangka.

Implementasi sukses dari CNN dalam mengklasifikasikan kematangan semangka akan memberikan manfaat yang signifikan bagi konsumen, industri pertanian dan pangan. Meskipun demikian, tantangan seperti keberadaan data pelatihan yang cukup dan penyesuaian parameter yang tepat harus diatasi. Dengan mengatasi hambatan ini, implementasi teknologi ini memiliki potensi untuk meningkatkan kualitas dari buah semangka, membawa keuntungan ekonomi serta memenuhi harapan konsumen akan produk berkualitas tinggi.

Berdasarkan uraian diatas penulis kemudian mengangkat sebuah penelitian dengan judul **"Klasifikasi Kematangan Buah Semangka Menggunakan Convolutional Neural Network"**. Untuk membangun sistem yang dapat mengidentifikasi kematangan buah semangka.

1.2 Rumusan Masalah

Adapun rumusan masalah yang menjadi dasar penelitian, yaitu:

a. Bagaimana cara mengidentifikasi kematangan buah semangka secara otomatis dengan menggunakan metode *Convolutional Neural Network* (CNN)?
b. Bagaimana tingkat keakuratan sistem klasifikasi kematangan buah semangka menggunakan metode *Convolutional Neural Network* (CNN)?

1.3 Tujuan

Berdasarkan permasalahan yang telah dirumuskan sebelumnya, maka tujuan dari penelitian yang akan dilakukan adalah:

a. Untuk mengembangkan suatu sistem yang dapat mengidentifikasi kematangan buah semangka secara otomatis dengan menggunakan metode Convolutional Neural Network (CNN).
b. Untuk mengetahui tingkat keakuratan pengelohan citra menggunakan Teknik CNN dalam mengidentifikasi kematangan buah semangka.

1.4 Manfaat

Dengan dilakukannya penelitian ini, diharapkan manfaat yang didapatkan antara lain:

1. Adanya kemudahan dalam mengidentifikasi tingkat kematangan buah semangka secara otomatis, sehingga dapat meningkatkan produktivitas pada proses pengelohan dan idetinfikasi buah.
2. Dapat mengurangi tingkat kesalahan dalam pemilihan buah semangka karena proses identifikasi tingkat kematangan dilakukan secara akurat dan objektif.
3. Dapat memberikan informasi yang berguna bagi peneliti lain yang ingin melakukan penelitian tentang kematangan semangka dalam bidang pengenalan citra atau pengelohan buah.

1.5 Ruang Lingkup

Adapun ruang lingkup dalam penelitian ini adalah sebagai berikut:

1. Jenis buah yang akan digunakan adalah semangka jenis baginda.
2. Objek penelitian difokuskan pada citra warna kulit dan corak buah.
3. Sistem klasifikasi menggunakan CNN dengan arsitektur *Xception*.
4. Pengambilan citra dari atas menggunakan kamera *smartphone.*
5. Data diambil dengan cara memotret setiap buah secara individu, dan proses klasifikasi juga dilakukan perbuah untuk memastikan akurasi identifikasi tingkat kematangan yang akurat.

1.6 Teori

1.6.1 Semangka

Semangka adalah tanaman merambat yang berasal dari daerah gurun kalahari di Afrika dan kini tumbuh pesat kemudian menyebar ke segala penjuru dunia, terutama di daerah tropis dan sub tropis mulai dari Jepang, Cina, Taiwan, Thailand, India, Jerman, Belanda bahkan Amerika. Buah semangka (citrullus lanatus) termasuk dalam golongan labu-labuan dan melon. Buah semangka merupakan buah yang banyak digemari banyak orang karena rasanya yang manis dan baik bagi kesehatan. Buah semangka banyak terdapat kandungan zat-zat yang sangat berguna bagi kesehatan tubuh manusia. Kandungan dari zat-zat tersebut dapat bermanfaat untuk melindungi jantung, memperlancar pengeluaran urine, dan menjaga kesehatan kulit serta mengandung zat-zat yang efektif dalam membunuh sel-sel kanker. Fungsinya tidak sekadar penghilang dahaga tetapi juga sebagai antioksidan yang baik. Buah semangka dapat diandalkan sebagai penetralisir radikal bebas dan mengurangi kerusakan sel dalam tubuh karena memiliki kadar antioksidan yang tinggi.

Di Indonesia sebagian besar semangka berharga murah dan tingkat pertumbuhannya tinggi, Sehingga semangka dapat menjadi ekspor yang menjanjikan. Agar dapat masuk menjadi komoditas ekspor, Semangka harus memenuhi syarat

kuantitas dan kualitas. Ditinjau dari segi kuantitas Pada tahun 2021 produksi semangka sebesar 414.242 Ton. Sentra produksi semangka mulai dari Jawa Timur, Jawa Tengah, Sumatera Utara, NTB, Sumatera Barat, Sumatera Selatan, Bali, Kalimantan Selatan, Sulawesi Selatan dan Lampung. Hal ini menunjukkan syarat kuantitas sudah terpenuhi. Akan tetapi yang menjadi masalah adalah dalam segi kualitas, pengecekan masih dilakukan secara manual yang tentunya sulit untuk diterapkan pada sistem yang besar.(Mariani et al., 2018).

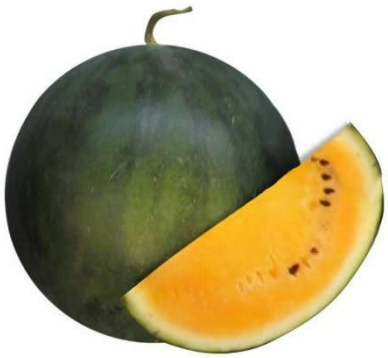
Jenis semangka. Ada dua jenis semangka yang sering ditanam di daerah takalar diantaranya adalah:

1. Semangka baginda (semangka merah).

Gambar 1 Semangka Baginda

Semangka Baginda F1 merupakan semangka hibrida yang cocok untuk ditumbuhkan di dataran rendah. Semangka Baginda F1 merupakan tipe semangka oblong yang cukup tahan penyakit layu fusarium, vigor. Buah yang dihasilkan berukuran besar, daging buahnya merah dan manis, berbiji, warna kulit hijau muda belurik. Berat buahnya sekitar 5,5-7,5 kg, dapat dipanen dalam 55-60 hari setelah tanam (HST). Potensi hasil panen adalah sekitar 25 sampai 35 ton per hektar.

2. Semangka passport (semangka kuning).



Gambar 2. Semangka passport

Semangka Passport adalah varietas semangka hibrida F1 Varietas ini cocok ditanam di dataran rendah dengan umur panen sekitar 64-65 hari setelah tanam. Buahnya berbentuk oval dengan kulit berwarna hijau tua berloreng hitam, daging buahnya berwarna oranye kekuningan dengan tekstur renyah dan rasa manis, Setiap buah memiliki berat rata-rata sekitar 6 kg, dengan potensi hasil panen mencapai 20 sampai 33 ton per hektar.

Dalam penelitian ini, semangka Baginda dipilih sebagai objek penelitian karena ketersediaannya yang melimpah di pasar lokal serta tingginya tingkat konsumsi dan permintaan di kalangan masyarakat. Selain itu, benih semangka Baginda memiliki harga yang lebih terjangkau dibandingkan dengan semangka Passport (semangka kuning) menjadikannya pilihan yang lebih ekonomis untuk dibudidayakan.

Perbedaan tingkat kematangan semangka. Ada beberapa perbedaan yang dapat dilihat dalam menentukan tingkat kematangan buah semangka baginda, Diantaranya adalah:

1. Semangka mentah memiliki warna kulit hijau tua. Pola garis-garis atau corak hijau pada kulitnya tipis dan hijau gelap, menandakan bahwa semangka ini masih mentah dan belum mencapai kematangan yang diinginkan.

Gambar 3. Contoh data semangka mentah

2. Semangka setengah matang memiliki warna kulit hijau muda. Garis-garis hijau pada kulitnya terlihat tebal dan jelas, namun warna kulitnya tidak seputih atau sekusam semangka matang. Kulit semangka ini masih terasa agak keras saat ditekan, menandakan bahwa daging buah di dalamnya sedang dalam proses menuju kematangan penuh.

Gambar 4. Contoh data semangka setengah matang

3. Semangka matang memiliki warna kulit hijau keputihan atau hijau kusam. Warna ini menunjukkan bahwa semangka telah mencapai tingkat kematangan optimal. Garis-garis atau corak hijau pada kulitnya mulai pecah dan melebar memberikan penampilan yang khas pada semangka matang.

Gambar 5. Contoh data semangka matang

1.6.2 Artificial Intelligence

Artificial Intelligence (AI) merupakan bagian dari ilmu komputer yang mempelajari bagaimana menjadikan mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan manusia, bahkan bisa lebih baik. Artificial Intelligence (AI) menurut John McCarthy (1956) yaitu untuk mengetahui atau memodelkan proses berpikir manusia dan mendesain mesin sehingga bisa menirukan perilaku manusia. Mesin bisa bertindak seperti manusia dengan dibekali pengetahuan serta kemampuan menalar yang baik.

1.6.3 Machine Learning

Machine Learning adalah cabang ilmu komputer yang memungkinkan komputer belajar memahami sesuatu tanpa program eksplisit. Machine Learning dapat didefinisikan sebagai metode komputasi eksperimental untuk meningkatkan kinerja atau membuat prediksi yang akurat. Adapun yang dimaksud dengan pengalaman adalah informasi yang ada yang dapat digunakan sebagai data pelatihan. Machine Learning adalah cabang dari kecerdasan buatan yang bertujuan membuat komputer belajar seperti manusia.

Pembelajaran mesin dirancang untuk membantu manusia menganalisis data dan membuat prediksi. Machine Learning melakukan pelatihan data untuk mempelajari, menghasilkan pola dan fitur, lalu melakukan klasifikasi.
Dalam pembelajaran machine learning, terdapat beberapa skenario-skenario seperti:

1. Supervised Learning

   Penggunaan skenario supervised learning, pembelajaran menggunakan masukan data pembelajaran yang telah diberi label. Setelah itu membuat prediksi dari data yang telah diberi label.

2. Unsupervised Learning

   Penggunaan skenario Unsupervised Learning, pembelajaran menggunakan masukan data pembelajaran yang tidak diberi label. Setelah itu mencoba untuk mengelompokkan data berdasarkan karakteristik-karakteristik yang ditemui.

3. Reinforcement Learning

   Pada skenario reinforcement learning fase pembelajaran dan tes saling dicampur. Untuk mengumpulkan informasi pembelajar secara aktif dengan berinteraksi ke lingkungan sehingga untuk mendapatkan balasan untuk setiap aksi dari pembelajar.

Saat ini telah banyak pendekatan machine learning yang digunakan untuk deteksi spam, Optical character recognition (OCR), pengenalan wajah, deteksi penipuan online, NER (Named Entity Recognition), Part-of-Speech Tagger (NURHIKMAT, 2018).

1.6.4 Deep Learning

Deep Learning (DL) merupakan salah satu bagian dari Machine Learning yang memanfaatkan jaringan syaraf tiruan (JST) untuk membantu menyelesaikan pada dataset yang besar. Teknik DL menjadikan arsitektur menjadi sangat kuat untuk Supervised Learning (Data pembelajaran mencakup keluaran yang sudah ditentukan). Penambahan lebih banyak lapisan menjadikan model pembelajaran yang bisa mewakili citra berlabel dengan lebih baik. Aplikasi konsep JST yang lebih dalam atau yang memiliki banyak lapisan dapat ditangguhkan pada algoritma Machine Learning yang sudah ada sehingga komputer bisa belajar dengan skala yang besar, kecepatan, dan akurasi. Prinsip tersebut semakin berkembang hingga DL semakin sering digunakan pada komunitas industri dan riset dalam menyelesaikan masalah data besar seperti pada Computer Vision, Speech Recognition, dan Neural Languange Processing.

Apabila dihadapkan dengan kasus klasifikasi, salah satu fitur DL yang sering kali digunakan adalah Feature Engineering yang merupakan teknik yang penting untuk mencapai hasil yang baik pada tugas prediksi dengan cara mengekstrak pola yang berguna dari data, sehingga bisa lebih mudah dalam membedakan kelas. Akan tetapi, fitur tersebut lebih sulit dipelajari dikarenakan kumpulan data dan berbagai jenis data yang berbeda perlu diselesaikan dengan pendekatan teknik yang berbeda pula. Dalam hal ini, metode Convolutional Neural Network sangat bagus untuk menemukan fitur yang baik pada citra ke lapisan berikutnya dalam kasus DL. Metode ini membentuk hipotesis

nonlinear sehingga bisa meningkatkan kekompleksitasan sebuah model (tiara et al., 2018).

1.6.5 Klasifikasi

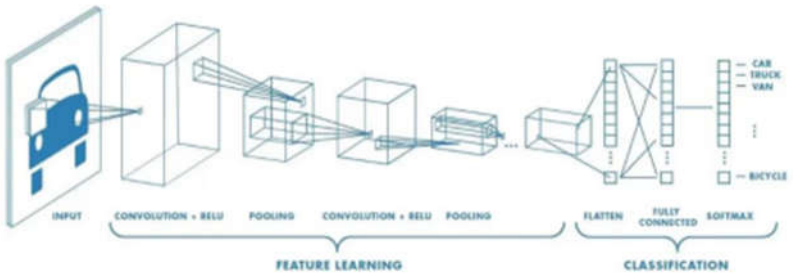
Dalam bidang pengolahan citra ada beberapa algoritma yang dapat digunakan, diantaranya adalah seperti Naive Bayes, Support Vector Machine, dan Neural Network. Algoritma yang umum digunakan adalah neural network. Jaringan saraf dikembangkan berdasarkan cara kerja jaringan saraf di otak manusia. Seiring dengan perkembangan teknologi saat ini, algoritma pengolahan citra digital juga terus berkembang sesuai dengan kebutuhan. Salah satu perkembangan deep learning adalah convolutional neural network.

Klasifikasi adalah proses menemukan model yang dapat membagi data berdasarkan kelas, dan dibagi menjadi dua fase: fase pelatihan (learning) dan pengujian (testing) untuk memahami bagaimana kategori data diketahui meningkat. Fase penilaian eksposur model merupakan hasil dari fase pelatihan yang menggunakan data baru sebagai data uji. Hasil dari fase ini adalah tingkat akurasi/kinerja model saat memprediksi data kelas yang tidak diketahui, khususnya data uji (Nana et al., 2022).

1.6.6 Convolutional Neural Network

Arsitektur CNN. Convolutional Neural Network (CNN) merupakan perkembangan dari multilayer perceptron (MLP) yang dirancang khusus untuk memproses data berdimensi dua, terutama dalam format citra. CNN termasuk dalam kategori Deep Neural Network karena memiliki kedalaman yang tinggi dan sering digunakan untuk tugas-tugas yang melibatkan data citra. Meskipun MLP dapat digunakan untuk klasifikasi citra secara dasar, CNN lebih sesuai karena mampu menyimpan informasi spasial dari data citra. Ini berbeda dengan MLP yang menganggap setiap piksel sebagai fitur independen, mengakibatkan hasil yang kurang optimal. Pendekatan CNN didasarkan pada penelitian awal oleh Hubel dan Wiesel (Hubel & Wiesel, T, 1968), yang membahas visual cortex pada indera penglihatan kucing. Secara teknis, CNN adalah suatu arsitektur yang dapat di-latih dan terdiri dari beberapa tahap. Setiap tahap memiliki masukan (input) dan keluaran (output) berupa beberapa array yang dikenal sebagai feature map. Setiap tahap juga terdiri dari tiga layer, yaitu konvolusi, fungsi aktivasi layer dan pooling layer (NURHIKMAT, 2018).

Alur proses pada Convolutional Neural Network (CNN) dalam pengolahan citra mulai dari proses masukan sampai klasifikasi citra menjadi hasil atau keluaran dapat dilihat pada Gambar 6.



Gambar 6. Struktur CNN

Berdasarkan ilustrasi pada Gambar 6, langkah awal dalam arsitektur CNN adalah tahap konvolusi. Langkah ini dilakukan dengan menerapkan kernel berukuran tertentu, dan jumlah kernel yang digunakan tergantung pada jumlah fitur yang ingin dihasilkan. Setelah itu, proses melanjutkan ke fungsi aktivasi, biasanya menggunakan fungsi aktivasi ReLU (Rectifier Linear Unit). Setelah keluar dari tahap fungsi aktivasi, langkah selanjutnya adalah melalui proses pooling. Proses ini diulang beberapa kali hingga diperoleh peta fitur yang memadai untuk langkah selanjutnya, yaitu masuk ke dalam jaringan saraf terhubung penuh (fully connected neural network), dan hasil akhir dari jaringan terhubung penuh ini merupakan output kelas.

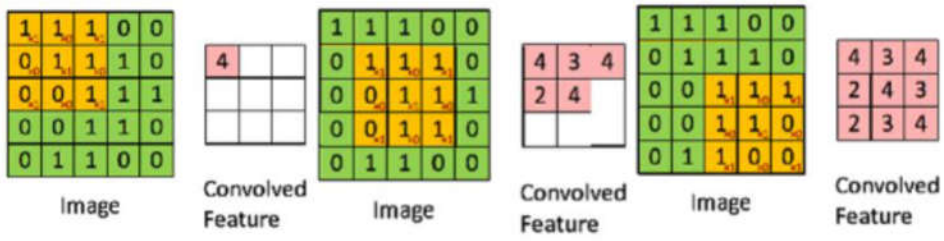
Convolutional Neural Network (CNN) memiliki lima lapisan atau layer utama, yaitu:

1. Lapisan Masukan (Input Layer)

   Lapisan masukan berfungsi sebagai penampungan dari nilai piksel citra yang menjadi input atau masukan. Input menyesuaikan dengan ukuran dan channel warna dari citra. Seperti contoh jika terdapat citra yang berukuran 64x64 dan mempunyai 3 channel warna yaitu RGB (Red, Green, Blue), maka yang menjadi input adalah piksel array yang memiliki ukuran 64x64x3.

2. Lapisan Konvolusi (Convolution Layer)

   Convolution layer merupakan pondasi dari CNN. berupa kernel/filteryang awalnya memiliki bobot acak, yang mana bobot ini akan diperbarui (update) pada saat training. Hasil yang dihasilkan dari convolution layer adalah citra baru yang memperlihatkan fitur-fitur dari citra masukan. Set iap citra yang dijadikan sebagai masukan melewati proses menggunakan filter convolution layer. Struktur convolutional layer terdiri dari neuron yang disusun sedemikian rupa untuk membentuk suatu filter dengan panjang, tinggi (pixel), dan jumlah (channel) tertentu. (Peryanto et al., 2019)



Gambar 7. Proses convolution layer

Kotak berwarna hijau secara keseluruhan mencakup citra yang akan mengalami proses konvolusi, kernel bergerak dari sudut kiri atas ke kanan bawah. Tujuan utama dari proses konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input (misalnya tepi, sudut, tekstur, dll). Output dari convolution layer biasa disebut feature map. Konvolusi menghasilkan transformasi linear dari data input, mencerminkan informasi spasial yang terdapat dalam data tersebut. Oleh karena itu, hasil konvolusi dari citra tersebut terlihat pada Gambar 5 pada Convolved Feature. Bobot pada lapisan tersebut menentukan spesifikasi kernel konvolusi yang digunakan, memungkinkan kernel konvolusi untuk dilatih berdasarkan input pada CNN.

3. Lapisan Aktivasi (*Activation Layer*)

Layer aktivasi atau activation layer, memiliki peran dalam memasukkan feature map ke dalam fungsi aktivasi. Fungsi aktivasi beroperasi untuk mengubah nilai-nilai pada feature map agar sesuai dengan rentang tertentu yang ditentukan oleh jenis fungsi aktivasi yang digunakan. Hal ini dilakukan dengan tujuan untuk meneruskan nilai-nilai yang mencerminkan fitur-fitur dominan dari citra, sehingga nilai tersebut dapat diarahkan ke lapisan berikutnya. Dalam penelitian ini, digunakan fungsi aktivasi ReLU dan Softmax.

a. Fungsi Aktivasi ReLu

Fungsi aktivasi ReLU (Rectified Linear Unit) adalah fungsi yang menghasilkan nilai nol jika inputnya kurang dari nol, dan menghasilkan nilai input itu sendiri jika input lebih besar dari nol, dengan kemiringan yang konstan sebesar 1 untuk nilai input yang positif. ReLU berfungsi untuk memperkenalkan nonlinearitas dalam model serta meningkatkan kemampuan representasi model. Secara matematis, fungsi aktivasi ReLU dapat dituliskan sebagai $f(x) = max\,(0, x)$ (Heaton, 2015). Output dari neuron yang menggunakan fungsi ini adalah nol untuk input negatif, dan sama dengan nilai input jika input tersebut positif (Kim, dkk., 2016).

Salah satu keunggulan dari fungsi aktivasi ReLU adalah kemampuannya untuk mempercepat proses pelatihan dengan menggunakan gradien stokastik, lebih cepat dibandingkan dengan fungsi aktivasi lain seperti sigmoid atau tanh. Hal ini disebabkan oleh bentuk linier dari ReLU yang tidak memerlukan operasi eksponensial seperti yang ada pada sigmoid/tanh, memungkinkan fungsi ini untuk melakukan komputasi

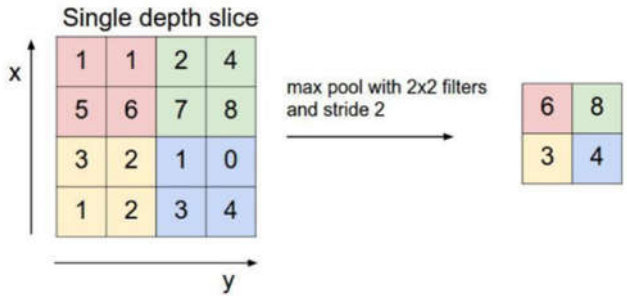
matriks aktivasi dengan efisien ketika nilai ambang batasnya adalah nol. Namun, fungsi ReLU juga memiliki kelemahan, yaitu rentan terhadap masalah "matinya neuron" selama proses pelatihan. Masalah ini terjadi karena gradien yang besar mengalir melalui fungsi ReLU, yang dapat menyebabkan pembaruan bobot yang tidak efektif, sehingga neuron menjadi tidak aktif pada titik data tertentu. Jika hal ini terjadi, gradien yang mengalir ke unit tersebut akan menjadi nol secara permanen, dan unit ReLU tersebut bisa mati secara ireversibel, menyebabkan kehilangan kemampuan model dalam memproses data manifold.

b. Fungsi Aktivasi Softmax

Fungsi aktivasi Softmax digunakan untuk menghasilkan hasil klasifikasi. Penggunaan aktivasi Softmax dianggap lebih efektif dibandingkan dengan fungsi aktivasi lainnya karena fungsi ini menghitung probabilitas untuk setiap kelas target terhadap kelas target yang memungkinkan. Selain itu, fungsi ini mengubah output dari lapisan terakhir dalam neural network menjadi distribusi probabilitas dasar, dan memastikan bahwa rentang probabilitasnya berada antara 0 hingga 1. Seluruh probabilitas yang dihasilkan oleh fungsi Softmax memiliki jumlah total 1, menciptakan suatu distribusi probabilitas yang lengkap (Syahputra & Wibowo, 2020). Fungsi ini biasanya diterapkan pada lapisan terakhir dari *Convolutional Neural Network*.

4. *Pooling Layer*

Pooling merujuk pada pengurangan ukuran matriks atau biasa disebut subsampling, Ada dua jenis pooling yang sering digunakan, Pertama yaitu avarage pooling yang mengambil nilai dari rata-rata, Sedangkan yang kedua yaitu *max pooling* mengambil nilai dari nilai maksimal. Penempatan lapisan pooling di antara lapisan konvolusi secara berurutan dalam struktur model CNN dapat secara bertahap mengecilkan dimensi volume keluaran pada peta fitur. Hal ini bertujuan untuk mengurangi jumlah parameter dan perhitungan dalam jaringan, serta untuk mengatasi masalah *overfitting*.



Gambar 8. Contoh Operasi Max Pooling

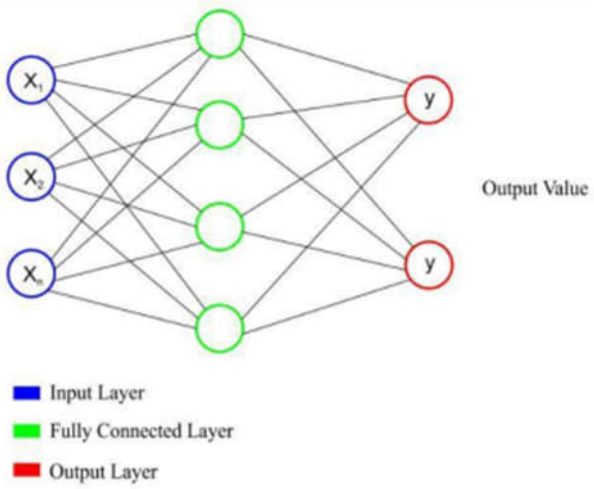
Pada Gambar 8 dapat dilihat pada kotak-kotak berwarna disisi kiri, yaitu merah, kuning, hijau, dan biru, membentuk kelompok yang akan mengambil nilai

maksimum (nilai tertinggi). Hasil dari proses *pooling layer* dengan menggunakan metode *max pooling* dapat dilihat pada kelompok kotak di

sebelah kanan. Langkah ini memastikan bahwa fitur yang diekstraksi tetap sama meskipun objek dalam citra mengalami translasi atau pergeseran. Penerapan pooling layer dalam CNN bertujuan untuk mengurangi ukuran citra, sehingga dapat digantikan dengan convolution layer yang memiliki stride yang sama dengan pooling layer yang bersangkutan. Stride adalah parameter yang menentukan jumlah pergeseran filter. Jika nilai stride adalah satu, filter akan bergerak satu piksel secara horizontal dan vertikal. Semakin kecil nilai stride yang digunakan, semakin detail informasi yang diperoleh dari input, namun memerlukan komputasi yang lebih tinggi dibandingkan dengan stride yang besar (Mardiyah, 2020).

5. *Fully Connected Layer*

Fully-Connected Layer merupakan suatu struktur dimana setiap *neuron* aktivasi dari lapisan sebelumnya memiliki koneksi dengan semua neuron di lapisan berikutnya, mirip dengan jaringan saraf biasa. Pada umumnya, lapisan ini digunakan dalam MLP (*Multi Layer Perceptron*) dengan tujuan melakukan transformasi dimensi data untuk memungkinkan klasifikasi linear. Perbedaan utama antara lapisan *Fully-Connected* dan lapisan konvolusi konvensional adalah bahwa neuron di lapisan konvolusi hanya terhubung ke area tertentu pada input, sedangkan lapisan *Fully-Connected* memiliki koneksi ke semua neuron secara menyeluruh. Meskipun demikian, keduanya tetap menggunakan operasi perkalian matriks (dot product), sehingga fungsinya tidak jauh berbeda (Peryanto et al., 2019). Itu semua dapat dilihat pada Gambar 9 berikut:



Gambar 9. Fully Connected Layer

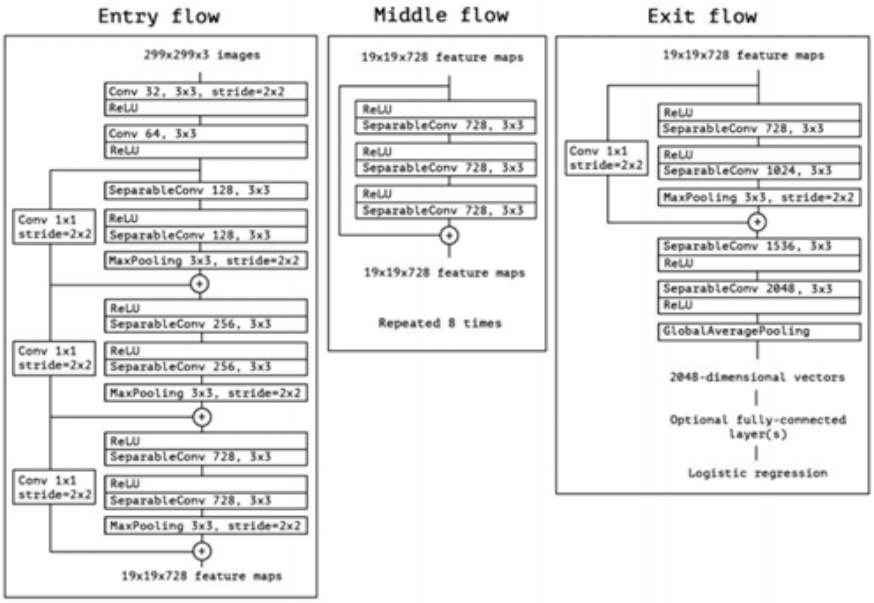
1.6.7 Kelebihan CNN

Menurut Yu dan Shi (2018), deep learning dengan metode CNN memiliki beberapa keunggulan, yaitu:

1. Proses hierarkis pada CNN menyederhanakan fitur data, sehingga mendukung pembelajaran representasi selama pelatihan. Hal ini memungkinkan CNN untuk beradaptasi secara otomatis terhadap data dan tugas prediksi pada bidang tertentu.
2. Peningkatan kekuatan komputasi mendukung pengembangan arsitektur jaringan saraf yang lebih dalam, yang pada akhirnya meningkatkan kemampuan representasi model serta menghasilkan prediksi yang lebih akurat.
3. Proses pembelajaran CNN menggunakan pendekatan pelatihan end-to-end, sehingga tidak memerlukan pengetahuan khusus tentang domain data yang digunakan oleh pengguna.

1.6.8 Xception

Xtreme of Inception (Xception) adalah arsitektur Convolutional Neural Network yang menggunakan metode depthwise separable convolution. Arsitektur ini merupakan hasil dari pengembangan arsitektur Inception yang memiliki 36 lapisan konvolusi yang membentuk basis jaringan ekstraksi fitur. Arsitektur Xception dapat dilihat pada gambar 10.



Gambar 10. Arsitektur Xception

Arsitektur Xception mengintegrasikan lapisan konvolusional dengan jaringan residual untuk meningkatkan efisiensi dalam ekstraksi fitur. Pada blok konvolusional pertama, setiap saluran dipisahkan secara spasial menggunakan konvolusi 1×1, yang berfungsi untuk mengurangi kompleksitas komputasi sekaligus mempertahankan

informasi esensial dari fitur awal. Setelah proses pemisahan, output tersebut dikombinasikan kembali melalui filter concatenation sebelum diproses lebih lanjut oleh lapisan konvolusional berikutnya. Pendekatan ini memungkinkan Xception untuk menangkap pola visual yang lebih kompleks secara lebih efisien, sehingga meningkatkan kemampuan model dalam mengenali fitur dari data input. Pemilihan arsitektur Xception dalam penelitian ini didasarkan pada berbagai pertimbangan yang ada pada tabel 1.

Tabel 1 Keras Applications

Model	Size (MB)	Top -1 Accuracy	Top-5 Accuracy	Parameters
Xception	88	79.0%	94.5%	22.9M
VGG16	528	71.3%	90.1%	138.4M
ResNet50	98	74.9%	92.1%	25.6M
NasNetMobile	23	74.4%	91.9%	5.3M
MobileNetv2	14	77.3%	94.4%	3.5M
EfficientNetB0	29	77.1%	93.3%	5.3M
EfficientNetB1	31	79.1%	94.4%	7.9M

Berdasarkan Tabel 1, terdapat empat alasan utama pemilihan Xception sebagai arsitektur utama dalam penelitian ini yaitu:

a. Xception memiliki akurasi Top-1 sebesar 79.0% dan Top-5 sebesar 94.5% pada dataset ImageNet. Ini lebih baik dibandingkan VGG16 atau ResNet50 dengan jumlah parameter yang lebih besar tetapi akurasi lebih rendah.
b. Xception menggunakan Depthwise Separable Convolution, yang lebih efisien dibandingkan konvolusi standar. Teknik ini memungkinkan model belajar fitur lebih baik dengan parameter yang lebih sedikit dan komputasi lebih cepat dibandingkan CNN konvensional seperti VGG atau ResNet.
c. Ukuran model Xception yang relatif efisien, yaitu 88 MB, lebih kecil dibandingkan dengan VGG16 dan ResNet50.
d. Jumlah parameter yang tidak berlebihan Xception memiliki 22.9 juta parameter, Jumlah parameter yang lebih sedikit dapat meningkatkan efisiensi komputasi tanpa mengorbankan akurasi secara signifikan.

Xception dipilih sebagai arsitektur utama karena menawarkan keseimbangan yang baik antara akurasi, efisiensi, dan jumlah parameter. Model ini lebih ringan dibandingkan arsitektur klasik seperti VGG16 dan ResNet50, namun tetap mampu mempertahankan akurasi yang tinggi. Hal ini menjadikannya pilihan yang ideal untuk penelitian yang membutuhkan kinerja yang baik dengan efisiensi komputasi yang baik. Selain itu, percobaan juga dilakukan dengan beberapa model ringan seperti MobileNetV2, EfficientNetB0, EfficientNetB1, dan NasNetMobile, dengan mempertimbangkan efisiensi pada perangkat seluler. Namun, hasil pelatihan yang diperoleh tidak cukup baik. Hal ini disebabkan oleh karakteristik model ringan yang umumnya lebih efektif pada dataset

berukuran besar, sedangkan jumlah data semangka dalam penelitian ini masih tergolong terbatas, sehingga tidak cukup mendukung kinerja optimal model ringan.

1.6.9 Confusion Matrix

Penentuan baik atau tidaknya performa suatu model klasifikasi dapat dilihat dari parameter pengukuran performanya, yaitu tingkat akurasi, recall, dan presisi. Untuk menghitung faktor-faktor tersebut diperlukan sebuah matrik yang biasa disebut *confusion matriks*. *Confusion matrix* merupakan sebuah table yang terdiri dari banyaknya baris data uji yang diprediksi benar dan tidak benar oleh model klasifikasi (Mardiyah, 2020). Salah satu *confusion matrix* yang kerap digunakan dalam pengukuran dapat dilihat pada Gambar 11.

		Kejadian Sebenarnya	
		P	N
Hipotesis Kejadian	P	True Positive	False Positive
	N	False Negative	True Negative

Gambar 11. Confusion Matrix

Keterangan pada Gambar 11:

A. *True Positive* (TP): Keadaan di mana classifier memprediksi dengan benar kelas positif sebagai positif.
B. *True Negative* (TN): Keadaan di mana classifier memprediksi dengan benar kelas negatif sebagai negatif.
C. *False Positive* (FP): Keadaan di mana classifier salah, yaitu memprediksi kelas negatif sebagai positif.
D. *False Negative* (FN): Keadaan di mana classifier salah, yaitu memprediksi kelas positif sebagai negatif.

1.6.10 Precision, Recall, dan Akurasi

1. Precesion

Precision yaitu tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Untuk menghitung nilai precision menggunakan rumus berikut:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

b. *Recall*

Recall yaitu tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Untuk menghitung nilai recall menggunakan rumus berikut:

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

c. Akurasi

Akurasi yaitu tingkat kedekatan antara nilai prediksi dengan nilai aktual. Akurasi dapat diartikan berupa kebenaran ataupun akurasi kesalahan. Untuk menghitung nilai Akurasi menggunakan rumus berikut:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \quad (3)$$

1.6.11 TensorFlow

Tensorflow merupakan open-source framework yang dapat digunakan untuk mengembangkan, melatih, dan menggunakan model deteksi objek. Sistem ini sudah banyak diterapkan pada berbagai produk Google antara lain pencarian image, deteksi wajah, dan plat nomor kendaraan pada google street, view, google assistant, way mau atau self-driving car, dan lain-lain.

Tensorflow pertama kali diumumkan oleh tim Google Brain pada tahun 2015. Tensorflow sangat memudahkan dan mempercepat penelitian dan penerapan model neural network. Tensorflow merupakan open-source framework yang dapat digunakan untuk mengembangkan, melatih, dan menggunakan model deteksi objek (M. Muttaqin, A. Stefanie, 2023).

1.6.12 Kelebihan TensorFlow

TensorFlow memiliki kelebihan antara lain:

1. Kompatibilitas Platform yang luas: tensorflow dapat digunakan pada berbagai platform seperti desktop, server, *mobile*, dan *edge devices*. Hal ini membuat tensorflow sangat fleksibel dan dapat diintegrasikan ke dalam berbagai aplikasi.
2. Kecepatan dan Skalabilitas: tensorflow dirancang untuk dapat memproses data yang sangat besar dan kompleks dengan kecepatan yang tinggi. Hal ini menjadikan tensorflow sebagai salah satu framework tercepat yang tersedia saat ini.

3. Dukungan komunitas yang luas: tensorflow memiliki komunitas yang besar dan aktif. Hal ini berarti bahwa ada banyak sumber daya dan dukungan yang tersedia untuk para pengguna termasuk dokumentasi, tutorial, dan forum diskusi.
4. Memiliki banyak fitur: tensorflow menyediakan berbagai jenis layer dan model yang dapat digunakan untuk membangun berbagai jenis model *machine learning*, seperti model *deep learning* dan *reinforcement learning*. Tensorflow juga menyediakan berbagai algoritma optimasi yang dapat digunakan untuk meningkatkan performa model.

1.6.13 TensorFlow Lite

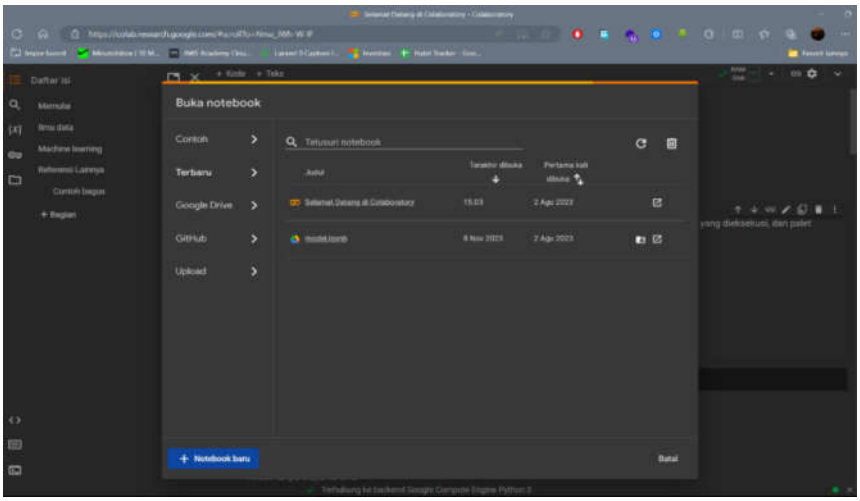
TensorFlow Lite adalah versi ringan (lite) dari *framework machine learning open-source* yang populer, TensorFlow. TensorFlow Lite dikembangkan oleh Google dan dirancang khusus untuk menjalankan model machine learning di perangkat *mobile* dan perangkat berdaya terbatas lainnya. Tujuannya adalah untuk memberikan kinerja yang cepat dan efisien dengan ukuran model yang lebih kecil, sehingga dapat digunakan dengan baik di perangkat bergerak. Hal tersebut juga bermanfaat pada proses klasifikasi, regresi, dan beberapa proses lain tanpa adanya proses pengiriman dan penerima Hal tersebut juga bermanfaat pada proses klasifikasi, regresi, dan beberapa proses lain tanpa adanya proses pengiriman dan penerimaan dengan server.

TensorFlow Lite dapat dijalankan pada perangkat Android maupun iOS dengan bantuan C++ API dan memiliki Java Wrapper untuk pengembangan berbasis Android. Pada perangkat Android terdapat interpreter atau penerjemah untuk proses akselerasi dengan adanya *Android Neural Networks*. Selain itu penggunaan CPU standar juga dapat digunakan untuk proses eksekusi.

TensorFlow Lite digunakan untuk melatih model yang kemudian akan diproses pada mesin dan dikonversi menjadi format ".tflite", kemudian model dengan format tersebut akan dimuat pada interpreter atau penerjemah seluler (A. Jauhari, 2022).

1.6.14 Google Colaboratory

Google Colaboratory, atau yang umunya dikenal sebagai Google Colab, adalah sebuah layanan yang disediakan oleh Google untuk memfasilitasi pengembangan dan eksekusi kode Python dalam lingkungan cloud berbasis Jupyter Notebook. Colab memberikan akses ke sumber daya komputasi cloud tanpa memerlukan instalasi atau konfigurasi lokal, sehingga pengguna dapat dengan mudah bekerja pada proyek-proyek pengembangan, analisis data, dan machine learning tanpa batasan perangkat keras lokal. Gambar 12 berikut ini merupakan tampilan dari Google Colaboratory.



Gambar 12. Tampilan Google Colaboratory

Penggunaan Google Colaboratory memerlukan akun Google untuk dapat diakses dan digunakan semua fitur yang terdapat di dalamnya. Keuntungan terbesar dari Google Colaboratory adalah bahwa ia memiliki kumpulan *built-in-library machine learning* paling populer yang dapat dimuat dengan mudah dalam notebook (A. Jauhari, 2022).

1.6.15 Android Studio

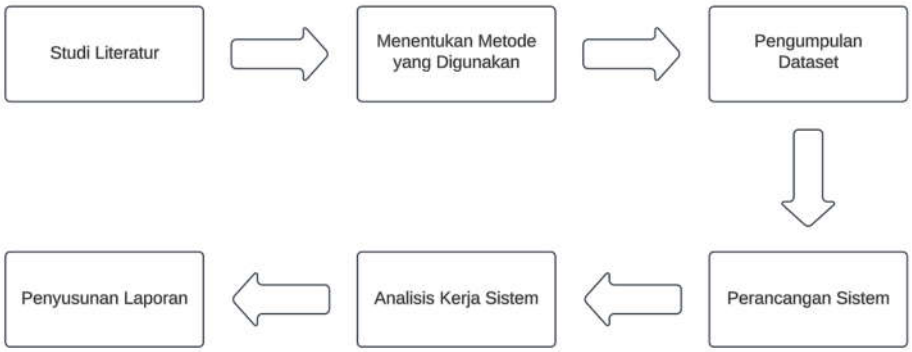
Android Studio adalah Integrated Development Environment (IDE) resmi yang disediakan oleh Google untuk pengembangan aplikasi Android. IDE ini dirancang untuk mempermudah seluruh proses pengembangan, mulai dari desain antarmuka (UI), pengkodean, hingga pengujian. Android Studio dilengkapi dengan berbagai fitur yang meningkatkan produktivitas pengembang dalam membuat aplikasi Android, seperti:

a. Memiliki sistem build berbasis *Gradle* yang fleksibel
b. Memiliki mulator yang cepat dan kaya fitur
c. Memiliki template kode dan integrasi GitHub untuk membantu Anda membuat fitur aplikasi umum dan mengimpor kode sampel
d. Memiliki dukungan C++ dan NDK
e. Memiliki dukungan bawaan untuk Google Cloud Platform, yang memudahkan integrasi Google Cloud Messaging dan App Engine.
f. Memiliki lingkungan terpadu tempat anda bisa mengembangkan aplikasi untuk semua perangkat android.
g. Memiliki *lint tool* untuk merekam performa, kegunaan, kompatibilitas versi, dan masalah lainnya.

BAB II
METODE PENELITIAN

2.1 Tahapan Penelitian

Penelitian “Klasifikasi Kematangan Buah Semangka Menggunakan Convolutional Neural Network” terdiri dari beberapa tahapan. Tahapan yang dilaksanakan sebagaimana ditunjukkan pada Gambar 13.



Gambar 13 Alur kerja penelitian

Berdasarkan Gambar 13, terdapat enam tahapan utama dapat dijelaskan secara ringkas sebagai berikut:

a. Studi Literatur

Pada tahap ini peneliti mengumpulkan literatur serta penelitian-penelitian sebelumnya dari berbagai sumber, yang terkait dengan topik yang diangkat yaitu Klasifikasi kematangan semangka menggunakan CNN.

b. Menentukan Metode yang Digunakan

Pada tahap ini peneliti menentukan metode yang akan digunakan untuk menyelesaikan masalah. Dalam hal ini, peneliti memilih metode Convolutional Neural Network from scratch, yang artinya model yang dibuat akan dibangun dari awal. Model tersebut nantinya akan diconvert ke TensorFlow Lite untuk selanjutnya diaplikasikan pada Android.

c. Pengumpulan Dataset

Pada tahap ini peneliti mengumpulkan dataset berupa citra semangka yang akan digunakan untuk tahap selanjutnya. Objek penelitian diperoleh langsung dari kebun semangka di Mangarobambang, Kota Takalar. Selanjutnya data akan dikumpulkan dengan teknis yang telah ditentukan.

d. Perancangan Sistem

Pada tahap ini peneliti menyusun konsep dan alur program, kemudian diimplementasikan sesuai dengan rancangan yang telah dibuat. Model dibuat dengan bahasa python dengan memanfaatkan library TensorFlow dan akan menggunakan algoritma CNN.

e. Analisis Kinerja Sistem

Pada tahap ini dilakukan analisis terhadap kinerja sistem yang telah dibuat. Tahap ini merupakan proses untuk mendapatkan kesimpulan terhadap penelitian yang telah dilakukan.

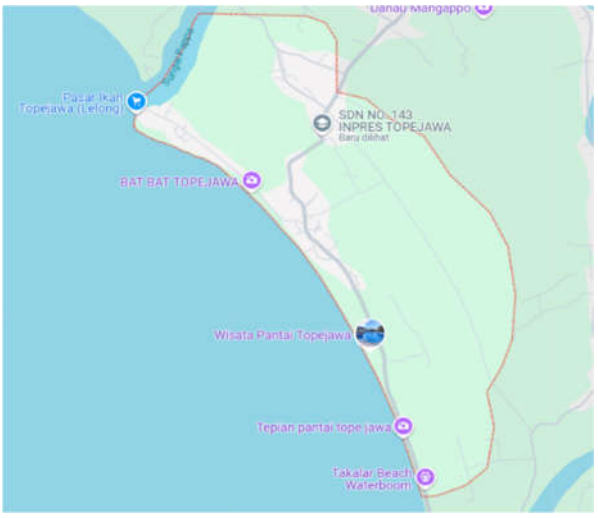
f. Penyusunan Laporan

Pada tahap ini dilakukan penulisan seluruh proses penelitian yang telah dilakukan dalam bentuk skripsi tugas akhir. Laporan ini kemudian akan dijadikan sebagai bahan publikasi maupun acuan untuk penelitian selanjutnya.

2.2 Waktu dan Lokasi Penelitian

Lokasi dan waktu pelaksanaan ditentukan sebagai berikut:

Waktu : 1 September 2023 – 18 Oktober 2024
Tempat : Kelurahan Topejawa Kec. Mangarabombang, Kabupaten Takalar, Sulawesi Selatan 92261



Gambar 14. Lokasi penelitian perkebunan semangka

2.3 Instrumen Penelitian

Instrumen penelitian yang digunakan pada penelitian ini diantaranya:

a. *Software*
   1. *Windows 11 x64*
   2. *Python 3.11.3*
   3. *Google Colaboratory*
   4. *Android Studio*

b. *Hardware*

Kamera : *Handphone* Samsung M52
Laptop : Acer Nitro 5
Laptop yang digunakan memiliki spesifikasi sebagai berikut:

Sistem Operasi	: Windows 11 Home 64-bit
Layar	: 15.6 inci FHD IPS
Processor	: Intel Core i5 Gen 8
RAM	: 16 GB
Storage	: 512 SSD
Kartu Grafis	: NVIDIA® GeForce GTX 1050

2.4 Pengumpulan Dataset

Penulis mengumpulkan data untuk memperoleh data atau informasi yang dibutuhkan untuk melaksanakan penelitian ini. Adapun metode pengumpulan data tersebut meliputi wawancara dan studi pustaka.

A. Wawancara

Data yang diperoleh dari hasil wawancara dengan salah satu pemilik kebun semangka pada kelurahan Mangarabombang.

B. Studi Pustaka

Studi pustaka diperlukan untuk memecahkan masalah yang ada, dengan membaca literatur yang berhubungan tentang penerapan *deep learning* menggunakan metode klasifikasi untuk memperoleh model terbaik berdasarkan hasil training. Adapun studi pustaka yang dilakukan yaitu melalui buku, buku elektronik (*Ebook*) dan jurnal.

2.5 Teknik Pengambilan Dataset

Sampel semangka yang digunakan pertama-tama dibersihkan untuk menghilangkan kotoran yang dapat memengaruhi hasil foto. Semangka kemudian diletakkan pada kertas berwarna hitam polos yang berfungsi sebagai latar belakang. Proses pengambilan gambar dilakukan dari jarak 25 cm antara kamera dengan objek menggunakan kamera *smartphone* Samsung Galaxy M52 yang dipasang pada tripod

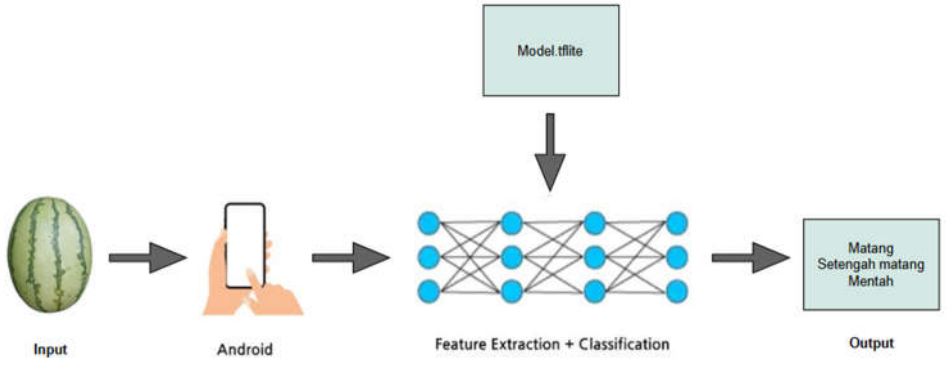
untuk menjaga kestabilan dan konsistensi posisi kamera. Data diambil jam 7 pagi hingga jam 11 siang. Citra diambil dari atas yang nantinya buah akan di putar tiap sisinya untuk mendapatkan data yang banyak, Pengambilan gambar dilakukan dengan memanfaatkan cahaya alami di luar ruangan yang dikontrol menggunakan payung sederhana untuk mengurangi intensitas cahaya langsung. Hasil gambar yang diperoleh secara default tersimpan dalam format .jpg dengan masing-masing dimensi piksel 3060 x 4080. Contoh cara pengambilan gambar dapat di lihat pada Gambar 15.

Gambar 15. Cara pengambilan data semangka

Total data yang diperoleh yaitu sebanyak 1350 citra yang nantinya akan dibagi kedalam tiga class. 1350 citra akan digunakan sebagai dataset dalam pembuatan model dan sisanya digunakan untuk proses uji coba menggunakan android. Data uji coba tidak boleh pernah dilihat oleh model sebelumnya.

2.6 Perancangan Sistem

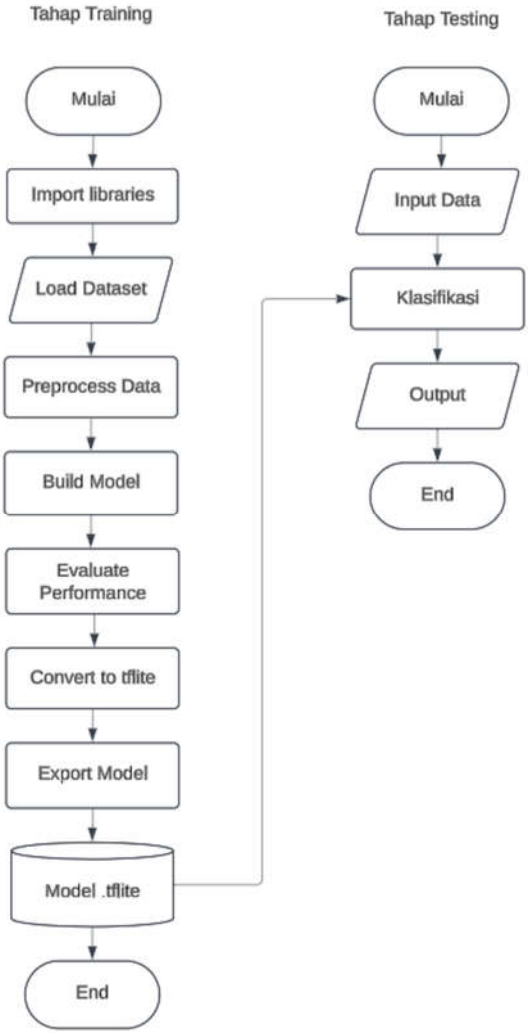
Sistem yang dibangun adalah sistem yang dapat mengidentifikasi kematangan buah semangka baginda dengan menggunakan model TensorFlow lite.



Gambar 16. Gambaran umum proses kerja sistem

Pada Gambar 16 merupakan gambaran umum dari sistem yang dibuat. Citra semangka akan di input menggunakan Android ke dalam sistem, yang kemudian akan menjalani proses *feature extraction* dan *classification* menggunakan model tflite yang telah dibuat hingga menghasilkan output berupa prediksi kelas tingkat kematangan semangka (mentah, setengah matang, atau matang).

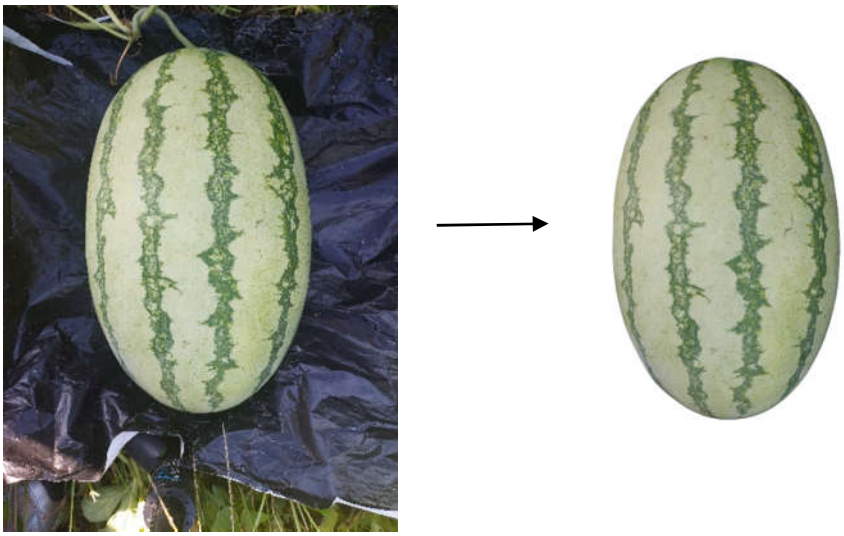
Pada penelitian ini, sebelum data memasuki proses *training*, terlebih dahulu dilakukan proses data preparation untuk mempersiapkan data yang akan digunakan. Selanjutnya, pada proses *training*, sebuah model dilatih menggunakan CNN berdasarkan dataset citra semangka yang telah disediakan. Hasil akhirnya adalah model dengan ekstensi .tflite yang akan digunakan untuk mengidentifikasi tingkat kematangan semangka. Tahap *testing* bertujuan untuk mengevaluasi seberapa baik kinerja sistem dalam memprediksi tingkat kematangan semangka. Flowchart tahap *training* dan *testing* dapat dilihat pada Gambar 17.



Gambar 17. Flowchart tahap training dan testing

2.6.1 Data Preparation

Sebelum data siap digunakan, terlebih dahulu dilakukan proses penghapusan background untuk mengurangi area yang tidak diperlukan dan memfokuskan gambar pada bagian semangka. Background removal dilakukan menggunakan aplikasi diluar dari sistem menggunakan tools dari website https://www.remove.bg/. Contoh data yang sudah mengalami background removal dapat dilihat pada Gambar 18 berikut.



Gambar 18. Citra yang telah dilakukan background removal

Setelah itu, data mengalami proses *cropping* untuk memastikan hanya bagian inti dari citra semangka yang digunakan dalam analisis, Setiap data yang sebelumnya memiliki dimensi piksel 3024 x 4032, setelah proses *cropping* memiliki dimensi yang berbeda-beda seperti pada tabel 2.

Tabel 2 Contoh data penelitian setelah proses cropping

Sebelum di crop	Sesudah di crop
 Matang	 Matang crop
 Mentah	 Mentah crop
 Setengah matang	 Setengah matang crop

Dataset berjumlah 1350 citra yang terdiri dari 3 class. Data kemudian dibagi menjadi beberapa sub folder sesuai dengan kelasnya masing-masing. Setelah persiapan data selesai, selanjutnya dataset disimpan ke dalam google drive untuk nantinya diakses melalui aplikasi google colaboratory untuk pembuatan model.

2.6.2 Tahap Training

1. Import Library

Tahap pertama yaitu mengimpor library yang dibutuhkan untuk proses training. Gambar 19 menunjukkan daftar library yang akan digunakan.

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Dense, Activation,Dropout,Conv2D, MaxPooling2D,BatchNormalization, Flatten,Input
from tensorflow.keras.optimizers import Adam, Adamax
from tensorflow.keras.metrics import categorical_crossentropy
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model, load_model, Sequential
from keras.callbacks import ModelCheckpoint
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from matplotlib.pyplot import imshow
import os
import seaborn as sns
sns.set_style('darkgrid')
from sklearn.metrics import confusion_matrix, classification_report
from IPython.display import display, HTML
```

Gambar 19. Import library

2. Load Dataset

Untuk memastikan bahwa Google Colaboratoy telah terhubung dengan benar ke dataset yang telah disimpan di Google Drive yang digunakan dapat dilakukan proses otentikasi dan mount Google Drive ke dalam Google Colaboratoy menggunakan perintah tertentu yang dapat dilihat pada Gambar 20, yang akan memungkinkan akses langsung ke file dan dataset yang ada di dalamnya, sehingga dapat untuk memanfaatkan data tersebut untuk keperluan analisis atau pemrosesan lebih lanjut.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
sdir =r'/content/drive/MyDrive/Skripsi/newdata/gabung'

filepaths=[]
labels=[]
classlist=os.listdir(sdir)
for klass in classlist:
    classpath=os.path.join(sdir,klass)
    if os.path.isdir(classpath):
        flist=os.listdir(classpath)
        for f in flist:
            fpath=os.path.join(classpath,f)
            filepaths.append(fpath)
            labels.append(klass)
Fseries= pd.Series(filepaths, name='filepaths')
Lseries=pd.Series(labels, name='labels')
df=pd.concat([Fseries, Lseries], axis=1)
print (df.head())
print (df['labels'].value_counts())
```

Gambar 20. Load dataset

3. Membagi porsi dataset

Setelah dataset berhasil diakses dari Google Drive yang telah terhubung dengan Google Colaboratoy, langkah selanjutnya adalah membagi dataset tersebut ke dalam tiga bagian utama, yaitu data latih, data validasi, dan data tes. Pembagian ini dilakukan dengan rasio 80% untuk data latih, 10% untuk data validasi, dan 10% untuk data tes.

Pembagian dataset ini sangat penting untuk memastikan bahwa model yang dikembangkan dapat diuji secara adil dan akurat, sehingga dapat menggeneralisasi dengan baik pada data yang belum pernah dilihat sebelumnya. Potongan source code yang digunakan untuk melakukan pembagian dataset tersebut dapat dilihat pada Gambar 21 yang memperlihatkan implementasi dari proses pembagian ini secara rinci.

```
train_split=.8
test_split=.1
dummy_split=test_split/(1-train_split)
train_df, dummy_df=train_test_split(df, train_size=train_split, shuffle=True, random_state=123)
test_df, valid_df=train_test_split(dummy_df, train_size=dummy_split, shuffle=True, random_state=123)
print ('train_df length: ', len(train_df), '  test_df length: ', len(test_df), '  valid_df length: ', len(valid_df))
```

```
train_df length:  1080   test_df length:  135   valid_df length:  135
```

```
height=224
width=224
channels=3
batch_size=64

img_shape=(height, width, channels)
img_size=(height, width)
length=len(test_df)
test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and length/n<=80],reverse=True)[0]
test_steps=int(length/test_batch_size)
print ( 'test batch size: ' ,test_batch_size, '  test steps: ', test_steps)

gen=ImageDataGenerator(
      rescale=1./255,
    )
train_gen=gen.flow_from_dataframe( train_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                   color_mode='rgb', shuffle=True, batch_size=batch_size)

validgen=ImageDataGenerator(rescale=1./255)
valid_gen=validgen.flow_from_dataframe( valid_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                   color_mode='rgb', shuffle=True, batch_size=batch_size)

testgen=ImageDataGenerator(rescale=1./255)
test_gen=testgen.flow_from_dataframe( test_df, x_col='filepaths', y_col='labels', target_size=img_size, class_mode='categorical',
                                   color_mode='rgb', shuffle=False, batch_size=test_batch_size)

classes=list(train_gen.class_indices.keys())
print (classes)
class_count=len(classes)
```

Gambar 21. Pembagian porsi dataset

Penjelasan pada Gambar 21 mengenai pembagian dataset adalah sebagai berikut:

a. Data train (*training*) merujuk pada data yang digunakan untuk melatih model deep learning. Model akan mempelajari pola-pola dalam data ini untuk meningkatkan kemampuannya dalam membuat prediksi.
b. Data val (*validation*) adalah data yang digunakan untuk memvalidasi model dan mencegah *overfitting*. Setelah model dilatih menggunakan *training dataset*, kinerjanya diuji dengan *validation dataset* untuk memastikan model dapat mengenali pola secara umum. Data ini juga digunakan untuk mengevaluasi akurasi model, dan jika hasilnya belum optimal, *hyperparameter* dapat disesuaikan untuk meningkatkan kinerja model.

Validation dataset bertujuan untuk mengevaluasi apakah bobot (bias dan weight) pada *Neural Network* yang dihasilkan dari training sudah memadai.

c. Data test (*testing*) adalah data yang digunakan untuk menguji model yang telah dibangun. Data ini memberikan gambaran seberapa baik model dapat menggeneralisasi dan memprediksi data yang belum pernah dilihat sebelumnya.

4. Membuat fungsi

Membuat fungsi untuk menampilkan beberapa contoh gambar Untuk melihat beberapa contoh gambar kita dapat menggunakan kode seperti pada Gambar 22 di bawah ini.

```
def show_image_samples(gen):
    test_dict=test_gen.class_indices
    classes=list(test_dict.keys())
    images,labels=next(gen) # get a sample batch from the generator
    plt.figure(figsize=(20, 20))
    length=len(labels)
    if length<25:   #show maximum of 25 images
        r=length
    else:
        r=25
    for i in range(r):
        plt.subplot(5, 5, i + 1)
        image=images[i]
        plt.imshow(image)
        index=np.argmax(labels[i])
        class_name=classes[index]
        plt.title(class_name, color='blue', fontsize=16)
        plt.axis('off')
    plt.show()
```

Gambar 22. Buat fungsi untuk menampilkan contoh gambar

5. Build Model

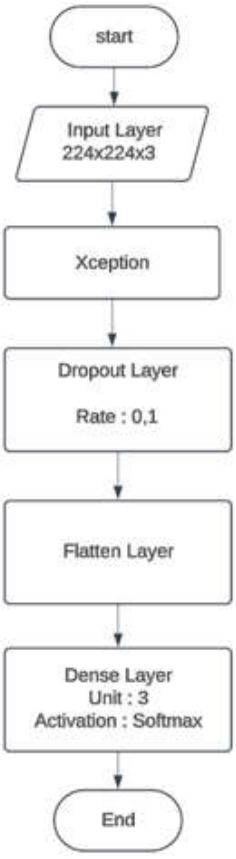
Proses pemodelan dilakukan dalam beberapa kali percobaan dengan hyperprameter yang berbeda untuk menemukan model klasifikasi varietas semangka baginda dengan akurasi terbaik. Secara garis besar arsitektur model CNN terdiri dari beberapa layer seperti *convolutional layer*, *pooling layer*, *dropout layer*, *flatten layer* dan *fully connected layer*.

```
model_name='Semangka'
print("Building model with", base_model)
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(224, 224, 3)),  # Define input layer
    base_model,  # Menggunakan Xception sebagai feature extractor

    tf.keras.layers.Dropout(rate=0.1),  # Regularisasi untuk mencegah overfitting
    tf.keras.layers.Flatten(),  # Mengubah hasil convolusi menjadi vektor 1D
    tf.keras.layers.Dense(3, activation='softmax')  # Output layer untuk klasifikasi 3 kelas
])
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

Gambar 23. Build model

Agar memudahkan dalam memahami cara kerja setiap layer pada model tersebut, maka diambil sampel Model Xception yang telah melalui serangkaian percobaan arsitektur guna memperoleh hasil dengan akurasi terbaik. Model ini dipilih karena memiliki performa yang lebih unggul dibandingkan arsitektur lainnya dalam mengklasifikasikan data secara optimal. Adapun Flowchart arsitektur yang menggambarkan struktur lapisan serta hyperparameter yang digunakan selama percobaan Model Xception dapat dilihat pada Gambar 24 berikut:



Gambar 24. Flowchart percobaan arsitektur model Xception

Tabel 3. Percobaan C - Hyperparameter Model Xception

	Hyperparameter
Split data	80 % Train, 10% Valid, 10% Test
Batch size	64
Layers	4
Epoch	20
Optimizer	Adam
Learning rate	0.0001
Dropout rate	0.1

Data pada penelitian ini dibagi menjadi 80% untuk data train, 10% untuk data validasi, dan 10% untuk data test. Model ini menggunakan batch size sebanyak 64, yang berarti jika dataset memiliki 1350 sampel data, maka model akan mengambil 64 sampel pertama untuk dilatih oleh algoritma CNN, lalu melanjutkan dengan 64 sampel berikutnya, hingga seluruh dataset selesai diproses. Proses ini dilakukan secara bertahap agar efisien dalam penggunaan memori dan mempercepat pelatihan. Arsitektur model terdiri dari 4 layer yaitu base model xception, layer dropout, layer flatten, dan layer fully connected untuk melakukan klasifikasi akhir. Model dilatih menggunakan optimizer Adam, yang bekerja dengan memperbarui bobot secara iteratif berdasarkan data training. Optimizer ini dikenal efisien dalam menangani data besar dan parameter yang kompleks. Learning rate yang digunakan adalah 0.001. Learning rate ini menentukan seberapa besar langkah pembelajaran model pada setiap iterasi nilai yang moderat seperti 0.001 memungkinkan model belajar secara optimal tanpa risiko konvergensi terlalu lambat atau terlalu cepat.

Input Layer. Citra dengan dimensi 224x224x3. Kemudian data citra yang telah di pra proses dimasukkan ke dalam input layer. Nilai input dan output dari layer ini adalah 223x224 dengan nilai matriks [-1,0,1] dan channel RGB.

Menambahkan Model Xception. Xception adalah salah satu model CNN yang tersedia di keras dengan akurasi terbaik saat ini. Model ini juga sering digunakan sebagai arsitektur CNN karena memiliki kelebihan yaitu dalam proses training memerlukan beban komputasi yang tidak terlalu tinggi, serta model yang dihasilkan lebih kecil namun tetap dengan performa terbaik sehingga dapat mempermudah dalam deployment aplikasi ini ke android agar beban kinerja tidak terlalu berat dan juga dapat memuat bobot model saat memasukkannya ke dalam android studio.

Model xception yang telah dilatih sebelumnya melalui transfer learning akan digunakan untuk dataset semangka baginda dengan input berupa citra ukuran 224 dan input channel sebanyak 3. Output pada tahapan konvolusi pertama akan dijadikan input untuk tahapan konvolusi kedua dan seterusnya. Ukuran citra input adalah 224 x 224 x 3 dan output dari tahapan ini adalah 7 x 7 x 2048 dengan total parameter 21,162,539. Sebagaimana yang tertera pada gambar 25:

Layer (type)	Output Shape	Param #
xception (Functional)	(None, 7, 7, 2048)	20,861,480
dropout (Dropout)	(None, 7, 7, 2048)	0
flatten (Flatten)	(None, 100352)	0
dense (Dense)	(None, 3)	301,059

Total params: 21,162,539 (80.73 MB)
Trainable params: 21,108,011 (80.52 MB)
Non-trainable params: 54,528 (213.00 KB)

Gambar 25. Output tahapan menambahkan model Xception

Convolutional Layer. Setelah menambahkan model Xception maka masuk ke tahap selanjutnya yaitu konvolusi. Tahapan ini akan memanfaatkan inputan yang kita dapatkan sebelumnya yaitu output image dari Xception yaitu 7 x 7 x 2048. Konvolusi pada tahapan ini menggunakan filter 32, kernel 3x3, padding same (0), activation relu, strides 1. Untuk memudahkan dalam penjelasan dan penulisan bagaimana proses untuk perhitungan pada convolution layer. Berikut proses konvolusi dengan memberikan nilai filter pada matrix. Pixel citra dengan 3 channel red, green, dan blue diambil data pixel-nya masing-masing dan ditambahkan padding 0 (same) di setiap pixelnya sehingga didapatkan pixel seperti Gambar 26.

Red

0	0	0	0	0	0
0	65	152	134	39	0
0	148	177	179	141	0
0	139	172	178	142	0
0	35	129	132	44	0
0	0	0	0	0	0

Green

0	0	0	0	0	0
0	76	172	155	45	0
0	170	195	196	156	0
0	157	187	190	156	0
0	39	141	144	47	0
0	0	0	0	0	0

Blue

0	0	0	0	0	0
0	51	113	90	23	0
0	123	156	153	101	0
0	110	149	149	109	0
0	25	94	97	31	0
0	0	0	0	0	0

Gambar 26.Nilai pixel RGB dataset

Pada percobaan ini digunakan kernel 3 x 3 dengan nilai seperti pada Gambar 27 di bawah ini :

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

Gambar 27. kernel 3x3

Langkah selanjutnya dilakukan proses perhitungan di setiap channel dengan dikalikan dengan kernel ukuran 3x3 pada Gambar 27. Tahapan ini dilakukan berulang dengan pergeseran kernel sebanyak 1 strides di setiap channelnya sehingga didapatkan perhitungan di setiap channel dengan nilai pada Gambar 28:

Channel red

Posisi 1 di *Channel Red*

0	0	0	0	0	0
0	65	152	134	39	0
0	148	177	179	141	0
0	139	172	178	142	0
0	35	129	132	44	0
0	0	0	0	0	0

((1*0) + (0*0) + ((-1)*0) + (1*0) + (0*65) + ((-1)*152) + (1*0) + (0*148) + ((-1)*177)) = −329

Posisi 2 di *Channel Red*

0	0	0	0	0	0
0	65	152	134	39	0
0	148	177	179	141	0
0	139	172	178	142	0
0	35	129	132	44	0
0	0	0	0	0	0

((1*0) + (0*0) + ((-1)*0) + (1*65) + (0*152) + ((-1)*134) + (1*148) + (0*177) + ((-1)*179)) = −100

Gambar 28. Ilustrasi perhitungan layer konvolusi *channel red*

Proses perhitungan dilakukan pada posisi 1 channel red yang diberi tanda kotak kuning akan dikalikan dengan nilai kernel 3x3 pada gambar 27 sehingga perkalian antara matrix 3x3 yang dikalikan sesuai dengan baris dan kolomnya mendapatkan nilai −329. Kemudian perhitungan dilanjutkan pada posisi 2 channel red dimana kotak kuning akan bergeser sebanyak 1 kolom sesuai dengan stridenya yaitu 1. Kemudian dilakukan perkalian nilai pada kotak kuning

dengan nilai kernel 3x3 pada gambar 27. sehingga perkalian antara matrix 3x3 yang dikalikan sesuai dengan baris dan kolomnya mendapatkan nilai −100. Tahapan ini akan terus berulang hingga posisi ke 16 dan mendapatkan nilai konvolusi untuk channel red pada gambar 29.

−329	−100	149	313
−501	−139	179	491
−478	−167	151	489
−301	−136	115	310

Gambar 29. Hasil tahapan perhitungan konvolusi *channel red*

Selanjutnya, dilakukan perhitungan pada channel hijau dengan mengalikan nilai piksel dengan kernel berukuran 3×3 pada Gambar 27. Proses perhitungan ini mengikuti tahapan yang sama seperti pada channel merah sebelumnya, termasuk metode dan penjelasannya. Hasil perhitungan untuk channel hijau ditampilkan pada Gambar 30 berikut:

Posisi 1 *Channel Green*

0	0	0	0	0	0
0	76	172	155	45	0
0	170	195	196	156	0
0	157	187	190	156	0
0	39	141	144	47	0
0	0	0	0	0	0

((1*0) + (0*0) + ((-1)*0) + (1*0) + (0*76) + ((-1)*172) + (1*0) + (0*170) + ((-1)*195)) = −367

Posisi 2 *Channel Green*

0	0	0	0	0	0
0	76	172	155	45	0
0	170	195	196	156	0
0	157	187	190	156	0
0	39	141	144	47	0
0	0	0	0	0	0

((1*0) + (0*0) + ((-1)*0) + (1*76) + (0*172) + ((-1)*155) + (1*170) + (0*195) + ((-1)*196)) = −105

Gambar 30. Hasil tahapan perhitungan konvolusi *channel Green*

Setelah perhitungan di channel green maka didapatkanlah hasil sebagaimana pada Gambar 31 berikut:

−367	−105	166	351
−554	−138	197	541
−523	−164	164	530
−328	−138	125	334

Gambar 31. Hasil tahapan perhitungan konvolusi *channel green*

Selanjutnya, dilakukan perhitungan pada *channel* biru dengan mengalikan nilai piksel dengan kernel berukuran 3×3 pada Gambar 27. Proses perhitungan ini mengikuti tahapan yang sama seperti pada *channel* merah sebelumnya, termasuk metode dan penjelasannya. Hasil perhitungan untuk *channel* biru ditampilkan pada Gambar 32 berikut:

Posisi 1 *Channel Blue*

0	0	0	0	0	0
0	51	113	90	23	0
0	123	156	153	101	0
0	110	149	149	109	0
0	25	94	97	31	0
0	0	0	0	0	0

((1*0) + (0*0) + ((-1)*0) + (1*0) + (0*51) + ((-1)*13) + (1*0) + (0*123) + ((-1)*156)) = −169

Posisi 2 *Channel Blue*

0	0	0	0	0	0
0	51	113	90	23	0
0	123	156	153	101	0
0	110	149	149	109	0
0	25	94	97	31	0
0	0	0	0	0	0

((1*0) + (0*0) + ((-1)*0) + (1*51) + (0*113) + ((-1)*90) + (1*123) + (0*156) + ((-1)*153)) = −69

Gambar 32. Ilustrasi tahapan perhitungan konvolusi *channel blue*

Setelah perhitungan di *channel blue* maka didapatkanlah hasil seperti Gambar 33 berikut ini:

−169	−69	145	243
−418	−108	185	392
−399	−141	158	399
−243	−111	103	246

Gambar 33. Hasil tahapan perhitungan konvolusi channel blue

Setelah melakukan tahapan konvolusi di setiap layer maka didapatkanlah nilai *red, green* dan *blue* seperti pada Gambar 34:

Red

−329	−100	149	313
−501	−139	179	491
−478	−167	151	489
−301	−136	115	310

Green

−367	−105	166	351
−554	−138	197	541
−523	−164	164	530
−328	−138	125	334

Blue

−169	−69	145	243
−418	−108	185	392
−399	−141	158	399
−243	−111	103	246

Gambar 34. Hasil perhitungan layer konvolusi channel red, green dan blue

Setiap nilai dari channel yang sudah melalui tahapan perhitungan pada Gambar 34 akan dijumlahkan yang menghasilkan output total seperti Gambar 35 :

Total = *Red* + *Green* + *Blue*

−865	−274	460	907
−1473	−385	561	1424
−1400	−472	473	1418
−872	−385	343	890

Gambar 35. Total perhitungan convolutional

Total = *Red* + *Green* + *Blue*

−865	−274	460	907
−1473	−385	561	1424
−1400	−472	473	1418
−872	−385	343	890

Hasil Relu

0	0	460	907
0	0	561	1424
0	0	473	1418
0	0	343	890

Gambar 36. Proses ReLU

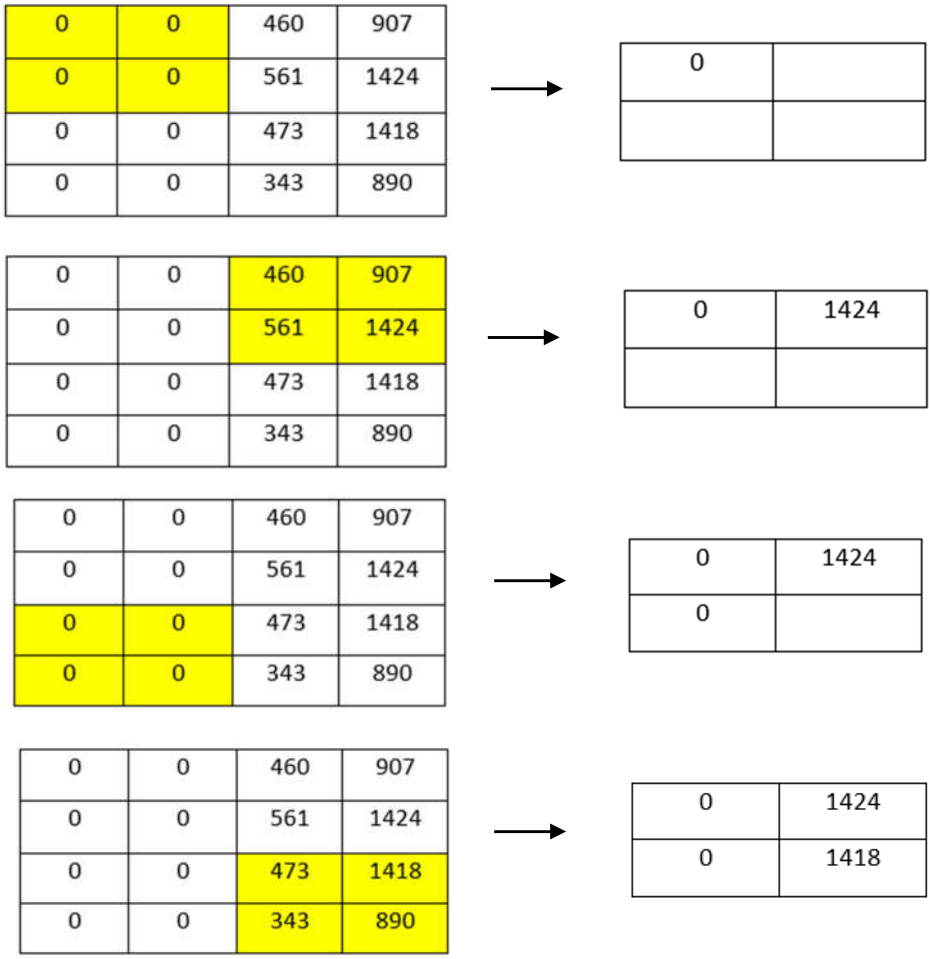
Pada tahap ini masukan dari neuron-neuron berupa bilangan negatif, maka fungsi ini akan menerjemahkan nilai tersebut ke dalam nilai 0, dan jika masukan bernilai positif maka output dari neuron adalah nilai aktivasi itu sendiri. Proses ini dilakukan secara berulang sampai semua citra dengan menggunakan 32 jenis filter yang berbeda sehingga output dari tahapan konvolusi ini akan menghasilkan banyak feature map, Gambar 36 merupakan contoh output dari salah satu feature map.

Pooling Layer.

0	0	460	907
0	0	561	1424
0	0	473	1418
0	0	343	890

Gambar 37. Hasil convolutional #1

Dari hasil pada Gambar 37, output tahapan konvolusi akan dijadikan sebagai input pada *pooling layer*, pada tahapan ini menggunakan *maxpooling* dengan kernel size 2x2 dengan stride 2 dengan output dari tahapan konvolusi. Ilustrasi proses *maxpooling* ditunjukkan seperti Gambar 38:



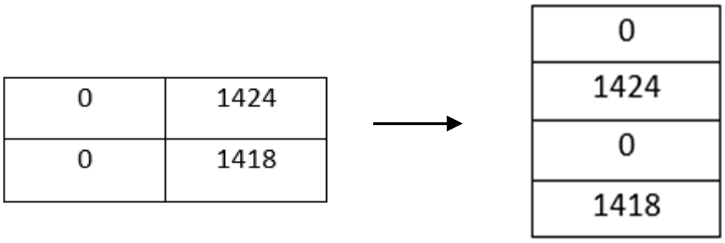
Gambar 38. Ilustrasi perhitungan layer (max pooling 2x2)

Gambar 38 adalah tahapan maxpooling dimana tahapan ini dilakukan secara dengan kernel size 2x2, dari nilai kernel ini akan diambil nilai tertinggi sehingga didapatkanlah nilai pertama yaitu 0 dikarenakan 0 merupakan nilai terbesar dalam kernel tersebut walaupun semua elemen nilainya sama. Langkah selanjutnya dilakukan pergeseran kernel sebanyak 2 karena kita menggunakan strides 2. Sehingga di dapatkanlah nilai kernel 2x2 pada matrix selanjutnya dengan nilai 460, 907, 561, dan 1424 dimana setelah dilakukan maxpooling didapatkan nilai tertinggi yaitu 1424, dan begitu seterusnya untuk proses maxpooling. Sehingga didapatkan output pada layer maxpooling yaitu 2x2x32 dengan nilai 0, 1424, 0 dan 1418 yang mana output ini akan dijadikan inputan pada dropout layer. Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari feature map (downsampling), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi overfitting. Max pooling dilakukan dengan mengambil nilai piksel terbesar di

setiap pooling kernel. Dengan begitu, sekalipun mengurangi jumlah parameter, informasi terpenting dari bagian tersebut tetap diambil.

Dropout Layer. *Dropout* berfungsi untuk mengurangi kompleksitas model yang telah dibangun dengan membuang neural network yang tidak terpakai. Ukuran model akan tetap 2x2x32, tidak ada perubahan.

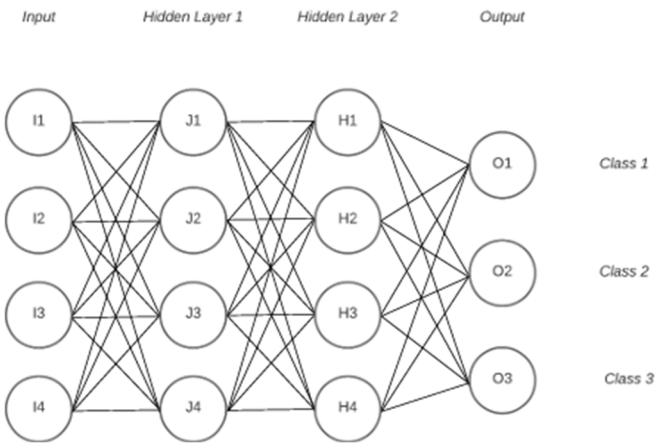
Flatten Layer. *Flatten* merupakan tahapan mengubah matriks ke bentuk vector/matrix 1 dimensi. Dari *output dropout layer* sebesar 2x2x32 akan didapatkan bentuk *vector* baru, gambar ilustrasi dapat dilihat pada Gambar 39 di bawah ini.



Gambar 39. Ilustrasi proses flatten

Gambar 38 adalah proses flatten dimana matrix 2x2 diubah menjadi matrix 4x1 atau vector. Output pada tahapan ini adalah vector 1 dimensi yang akan digunakan sebagai input di fully connected layer (dense).

Fully Connected Layer. Output dari tahapan *flatten layer* yaitu vektor 1 dimensi, yang akan menjadi inputan pada tahapan ini, berikut ilustrasi *fully connected layer* pada Gambar 40 berikut ini:



Gambar 40. Ilustrasi gambar dense + softmax

Tahap Classification menggunakan Fully Connected Layer. Feature map yang dihasilkan dari feature extraction masih berbentuk multidimensional array, sehingga harus melakukan “flatten” atau reshape feature map mejadi sebuah vector agar bisa digunakan sebagai input dari fully-connected layer.

Lapisan Fully-connected adalah lapisan dimana semua neuron aktivitas dari lapisan sebelumnya terhubung semua dengan neuron di lapisan selanjutnya. Dense layer adalah lapisan sederhana dari neuron dimana setiap neuron menerima input dari semua neuron lapisan sebelumnya. Dense layer digunakan untuk mengklasifikasikan citra berdasarkan output dari convolutional layer.

Tahapan terakhir dari fully connected layer pada penelitian ini adalah dense layer yang memiliki 3 neuron dengan fungsi aktivasi softmax. Fungsi aktivasi softmax digunakan ketika kita memiliki dua atau lebih dari dua class. Jika kita memiliki total 3 class, maka jumlah neuron pada lapisan keluaran adalah 3. Setiap neuron mewakili satu class. Fungsi softmax digunakan untuk menghitung nilai probabilitas untuk semua label atau class.

Tabel 4. Contoh probabilitas untuk setiap kelas

0	1	2	Label
0.9546832	0.027077	0.334747	Matang
0.027077	0.9129704	0.070122	Setengah Matang
0.216773	0.048301	0.9694372	Mentah

Pelatihan Model. Epoch atau jumlah training setiap percobaan dilakukan sebanyak tabel epochs dari setiap percobaan. Jumlah epoch dapat mempengaruhi hasil tingkat akurasi, namun epoch yang terlalu tinggi juga akan berpengaruh ke lama waktu proses dalam melakukan training. Oleh karena itu, pemilihan jumlah epoch yang optimal sangat penting untuk mencapai keseimbangan antara akurasi yang tinggi dan efisiensi waktu komputasi. Jika jumlah epoch terlalu sedikit, model mungkin belum sepenuhnya belajar pola dalam data, sehingga menghasilkan akurasi yang rendah. Sebaliknya, jika jumlah epoch terlalu banyak, model berisiko mengalami overfitting, di mana model terlalu menyesuaikan diri dengan data pelatihan dan kurang mampu melakukan generalisasi terhadap data baru.

Percobaan dilakukan dengan 20 epoch atau 20 kali iterasi, di mana setiap epoch merupakan satu siklus penuh pelatihan model menggunakan seluruh data yang tersedia. Setelah pelatihan selesai, model ini berhasil mencapai tingkat akurasi sebesar 96,30%, yang menunjukkan bahwa model mampu mengenali pola dalam data dengan cukup baik. Selain itu, nilai loss dan accuracy yang diperoleh selama proses training dan validasi dapat dilihat secara lebih rinci pada Tabel 5:

Tabel 5. Training Model Percobaan C - Xception Tanpa Lapisan

Epoch	Accuracy	Loss	Val_Accuracy	Val_Loss
1	0.6409	0.7740	0.7333	0.5878
2	0.9735	0.1089	0.6444	0.9014
3	0.9956	0.0141	0.8444	0.4082
4	0.9973	0.0072	0.8074	0.5577
5	0.9985	0.0033	0.8667	0.4256
6	1.0000	0.0017	0.8667	0.4671
7	1.0000	8.8710e-04	0.8963	0.3863
8	1.0000	4.4621e-04	0.9259	0.3271
9	1.0000	9.5369e-04	0.9333	0.3132
10	1.0000	3.2372e-04	0.9407	0.3199
11	1.0000	0.0011	0.9259	0.2683
12	1.0000	5.8198e-04	0.9259	0.3421
13	1.0000	3.5705e-04	0.9481	0.2525
14	1.0000	2.8120e-04	0.9556	0.2395
15	1.0000	1.1348e-04	0.9556	0.2467
16	1.0000	5.9682e-04	0.9407	0.2753
17	1.0000	1.8072e-04	0.9481	0.2700
18	1.0000	1.0489e-04	0.9556	0.2552
19	1.0000	1.0460e-04	0.9556	0.2423
20	1.0000	2.3844e-04	0.9630	0.2310
Total nilai akurasi = 96.30%				

Gambar 41. Hasil Proses Training Percobaan C

Gambar 41 menunjukkan proses pelatihan model dengan pola yang konsisten antara training loss dan validation loss. Training loss (garis merah) terus menurun seiring bertambahnya epochs, menunjukkan bahwa model

semakin mampu mempelajari pola dari data pelatihan. Validation loss (garis hijau) juga menunjukkan tren penurunan hingga mencapai titik optimal pada epoch ke-20, yang ditandai sebagai best epoch.

Pada grafik accuracy, training accuracy (garis merah) meningkat secara bertahap dan mencapai nilai mendekati sempurna, menunjukkan bahwa model berhasil memprediksi data pelatihan dengan sangat baik. Validation accuracy (garis hijau) juga meningkat hingga mencapai akurasi terbaik pada epoch ke-20, menunjukkan bahwa model mampu menghasilkan prediksi yang baik terhadap data validasi. Hasil ini menunjukkan bahwa model Xception berhasil belajar secara baik dari data pelatihan dan validasi, dengan performa akurasi yang terus meningkat selama proses pelatihan.

Validasi Sistem. Validasi model menggunakan confusion matrix dengan melihat hasil dari kinerja model berupa nilai akurasi dan loss pada data training dan data validation untuk mendapatkan nilai akurasi, presisi, recall, dan hasil f1. Hasil ini nantinya dijadikan landasan apakah model perlu dilakukan pelatihan ulang atau tidak bergantung pada hasil yang didapat. Berikut persamaan yang digunakan:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (4)$$

$$Presisi = \frac{TP}{TP+FP} \quad (5)$$

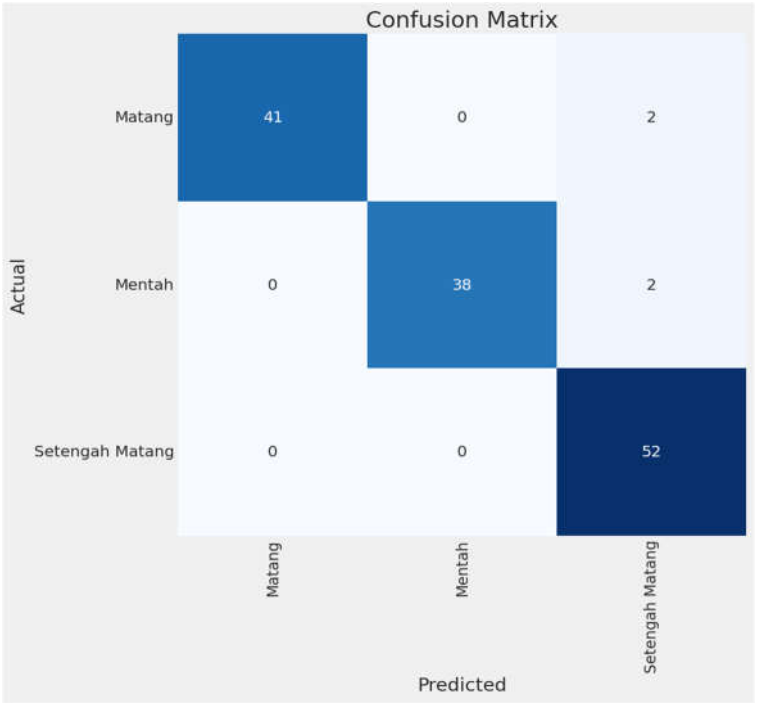
$$Recall = \frac{TP}{TP+FN} \quad (6)$$

$$F1 - Score = 2 \times \frac{precision \times recall}{precision+recall} \quad (7)$$

Keterangan:
True Positif (TP): Data positif diprediksi positif
True Negatif (TN): Data negatif diprediksi negatif)
False Positif (FP): Data negatif namun diprediksi sebagai data positif
False Negatif (FN): Data positif namun diprediksi sebagai data negatif

Tingkat akurasi model yang dikembangkan menggunakan arsitektur CNN mencerminkan sejauh mana model tersebut mampu mengklasifikasikan data dengan tepat. Presisi mengacu pada tingkat kesesuaian antara data yang diinginkan dan hasil prediksi yang dihasilkan oleh model CNN. Recall menggambarkan sejauh mana model CNN berhasil dalam mengidentifikasi kembali informasi yang relevan. Sementara itu, f1-score adalah rata-rata harmonis antara presisi dan recall, yang memperhitungkan keseimbangan keduanya.

Pada percobaan menggunakan model Convolutional Neural Network (CNN) dengan arsitektur Xception, dataset yang digunakan berjumlah sebanyak 1350 sampel. Adapun gambar percobaan tersebut dapat dilihat pada Gambar 42 berikut:



Classification Report:

	precision	recall	f1-score	support
Matang	1.00	0.95	0.98	43
Mentah	1.00	0.95	0.97	40
Setengah Matang	0.93	1.00	0.96	52
accuracy			0.97	135
macro avg	0.98	0.97	0.97	135
weighted avg	0.97	0.97	0.97	135

Gambar 42. Hasil prediksi model Percobaan Xception Tanpa Lapisan

Dari pengujian yang telah dilakukan berdasarkan sampel data uji sebanyak 135. Maka didapatkan presentasi akurasi pengujian klasifikasi confusion matrix multi class dalam mengidentifikasi objek yaitu sebesar 9,33%.

Kelas Matang

TP: 41

FP: 2

FN: 0

Akurasi = $\frac{41}{41+2+0}$ x100 = 95.35%

Presisi = $\frac{41}{41+2}$ = 0.9535

Recall = $\frac{41}{41+0}$ = 1,0

F1 – Score = 2 x $\frac{0.9535 \times 1.0}{0.9535 \times 1.0}$ = 0.976

Kelas Mentah
TP = 38
FP = 2
FN = 0
Akurasi = $\frac{38}{38+2+0}$ x100 = 95%
Presisi = $\frac{38}{38+2}$ = 0.95
Recall = $\frac{38}{38+0}$ = 1,0
F1 – Score = 2 x $\frac{0.95\times1,0}{0.95\times1,0}$ = 0.9744

Kelas Setengah Matang
TP = 52
FP = 0
FN = 2 + 2 = 4
Akurasi = $\frac{52}{52+0+4}$ x100 = 92.86%
Presisi = $\frac{52}{52+0}$ = 1,0
Recall = $\frac{52}{52+4}$ = 0.9286
F1 – Score = 2 x $\frac{1,0\times0.9286}{1,0\times0.9286}$ = 0.9632

Dari pengujian yang telah dilakukan berdasarkan sampel data uji sebanyak 135. Maka didapatkan akurasi pengujian klasifikasi confusion matrix dalam mengidentifikasi objek yaitu sebesar 97.04%%.
Total TP = 41+38+52= 131
Akurasi = $\frac{131}{135}$ x 100 = 97.04%

Convert Ke TensorFlow Lite. Model yang telah dibangun akan di convert ke TensorFlow Lite agar bisa digunakan pada android.

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

Gambar 43. Convert ke TensorFlow lite

Save Model. Model yang telah diconvert ke TensorFLow Lite kemudian disimpan yang nantinya akan digunakan untuk klasifikasi kematangan semangka menggunakan android.

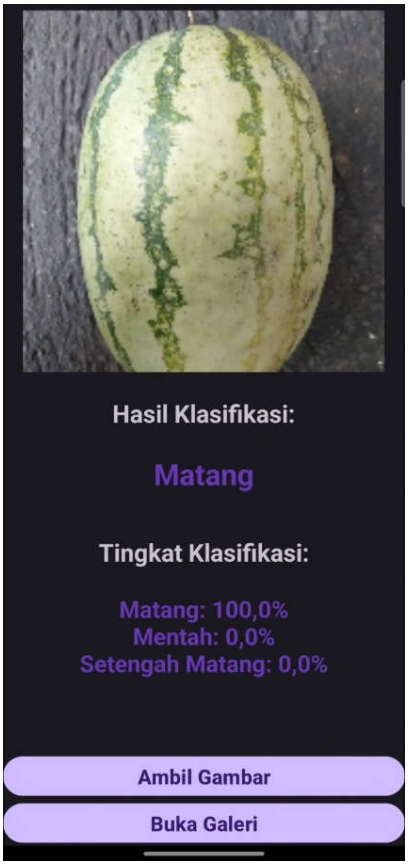
```
with open(output_path, 'wb') as f:
  f.write(tflite_model)
```

Gambar 44. Save

Deployment. Pada tahap ini dilakukan deployment yang berupa aplikasi android. tampilan antarmuka dari model yang dihasilkan oleh pengolahan data menggunakan Python. Deployment ini dilakukan menggunakan android studio. Berikut adalah desain tampilan antarmuka dari watermelon classification menggunakan algoritma Convolutional Neural Network.

Gambar 45. Tampilan layar utama aplikasi

Gambar 45 merupakan tampilan layar utama dari aplikasi android yang di buat. Aplikasi ini memiliki fitur untuk memasukkan gambar, baik melalui kamera dengan menekan tombol 'Ambil Gambar' maupun dari galeri perangkat seluler dengan menekan tombol 'Buka Galeri'. Setelah pengguna memasukkan gambar sistem akan melakukan deteksi berdasarkan model model.tflite yang telah diintegrasikan ke dalam aplikasi. Pada halaman awal hasil klasifikasi belum ditampilkan karena pengguna belum memberikan input berupa gambar yang akan dideteksi.



Gambar 46. Contoh hasil klasifikasi aplikasi

Pada Gambar 46 merupakan contoh gambar yang sudah di input melalui fitur gambar kamera atau galeri dan akan menampilkan hasil klasifikasi sesuai dengan jenis yang diprediksi oleh model terlatih yang sudah diintegrasikan pada aplikasi tersebut.

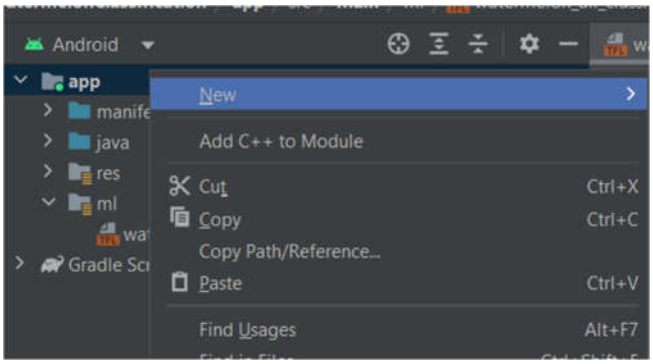
2.6.3 Tahap Testing

a. Tahap testing

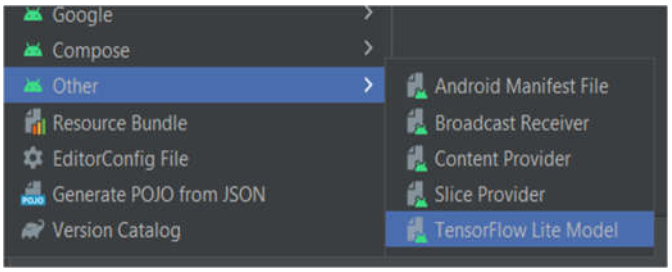
Model yang telah dikembangkan akan dievaluasi dengan mengukur kinerjanya melalui nilai akurasi dan loss pada data testing. Proses pengujian ini dilakukan menggunakan Google Colaboratory untuk memastikan performa model sebelum dikonversi ke format TensorFlow Lite.

b. Testing Dengan Aplikasi Android

Pengujian kedua dilakukan menggunakan aplikasi Android yang telah dikembangkan. Pengujian ini dilakukan pada data baru yang belum pernah digunakan oleh model, baik dalam data training, validation, maupun testing. Aplikasi dirancang menggunakan Android Studio, dengan model yang diimpor dalam format TensorFlow Lite untuk digunakan dalam proses identifikasi.



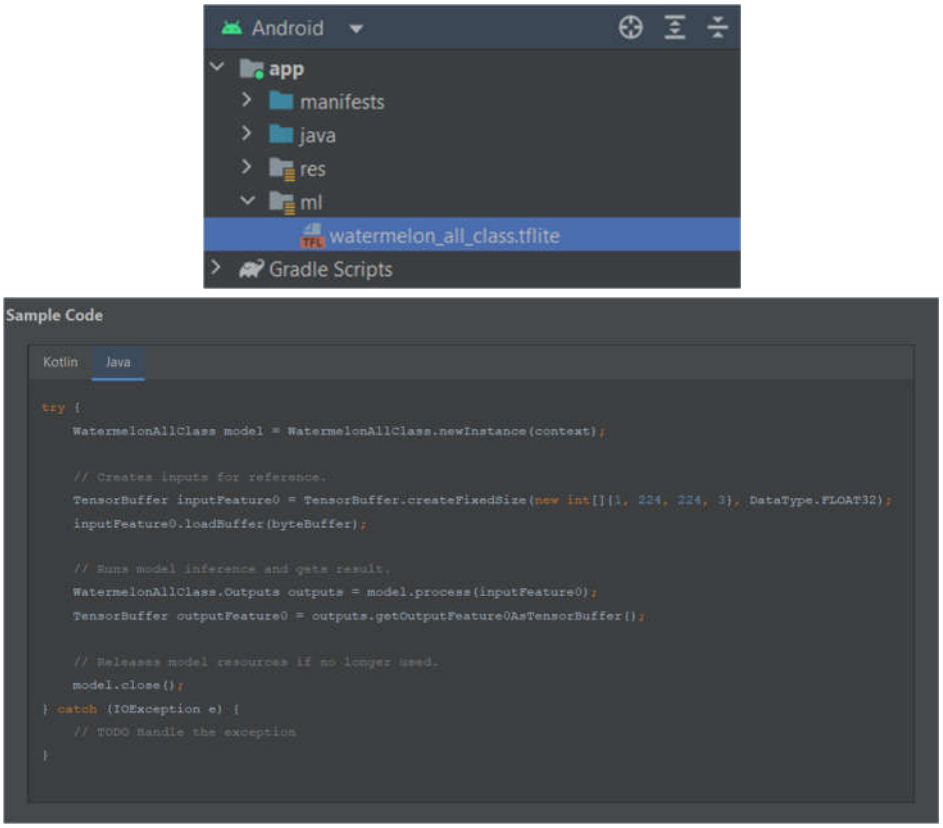
(a) Klik kanan lalu pilih New



(b) Pilih Other lalu pilih TensorFlow Lite Model

Gambar 47. Import TensorFlow Lite Model

Setelah model berhasil di import, akan terbentuk folder baru yang berisi model dengan ekstensi .tflite. Ketika model diklik akan memberikan sample code untuk menggunakan model tersebut pada proses identifikasi.



Gambar 48. Sample code TensorFlow lite model

c. Mengevaluasi Hasil Prediksi

Untuk menghitung nilai akurasi sistem, digunakan metode Confusion Matrix.

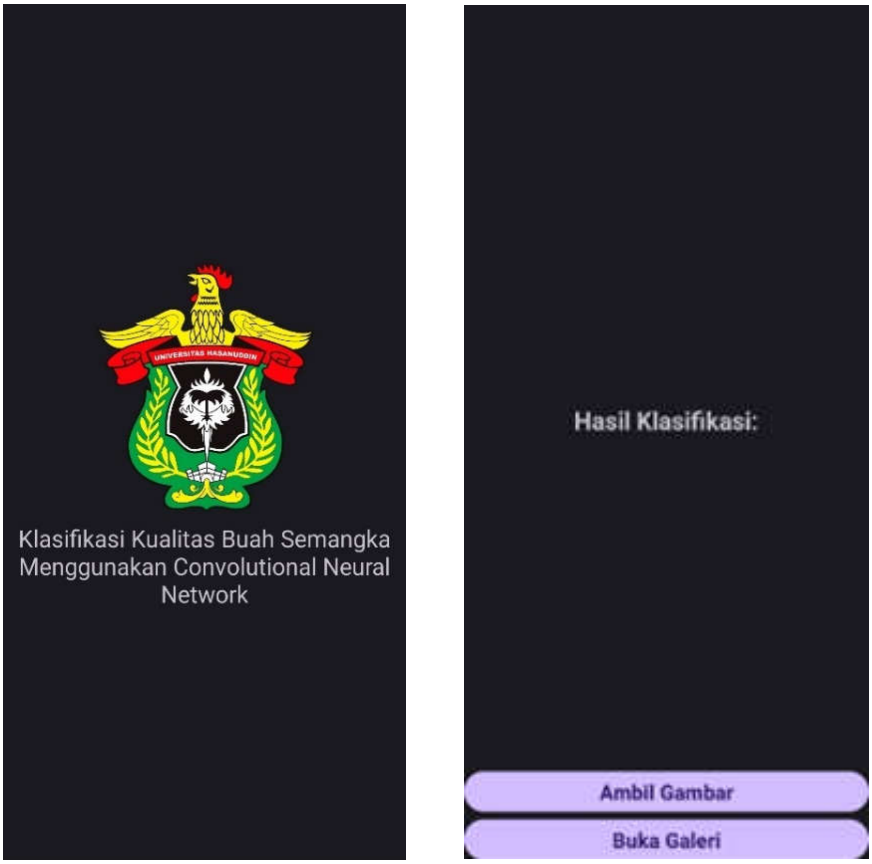
Tabel 6. Tabel confusion matrix yang akan digunakan

		Prediksi		
		Matang	Setengah Matang	Mentah
Aktual	Matang	TP		
	Setengah Matang		TP	
	Mentah			TP

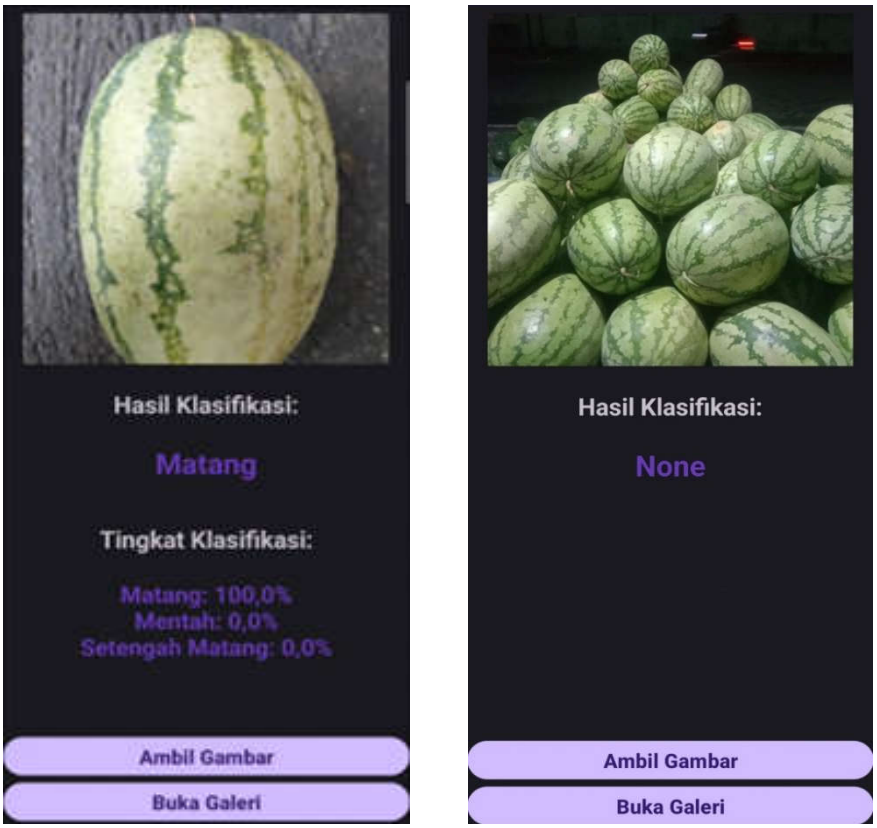
2.6.4 Implementasi Sistem

Sistem yang dikembangkan merupakan aplikasi mobile sederhana. Ketika pengguna pertama kali membuka aplikasi, halaman Splash Screen akan ditampilkan selama 3 detik. Setelah itu, pengguna diarahkan ke halaman utama, yang menyediakan dua opsi untuk menginput gambar menggunakan kamera atau memilih gambar dari galeri perangkat seluler. Gambar yang dipilih akan ditampilkan di layer kemudian sistem akan melakukan proses identifikasi berdasarkan model terlatih yang telah dibuat sebelumnya.

Hasil identifikasi akan ditampilkan bersama persentase klasifikasi untuk setiap kelas. Jika persentase klasifikasi salah satu kelas melebihi 50%, maka aplikasi akan menampilkan hasil klasifikasi kelas tersebut. Namun, jika tidak ada kelas yang memiliki persentase di atas 50%, hasil klasifikasi yang ditampilkan adalah unknown. Rancangan tampilan sistem dapat dilihat pada Gambar 49 & Gambar 50.



Gambar 49. Interface Awal Sistem



Gambar 50. Interface Hasil Klasifikasi

2.6.5 Analisis Kerja Sistem

Setelah seluruh tahap training dan testing dilakukan, berikutnya adalah mengukur performa atau kemampuan model dengan menggunakan confusion matrix. Yang dimana perhitungan tersebut mencakup nilai akurasi, precision, recall dan score f-1 untuk mengetahui tingkat ketepatan prediksi benar dari keseluruhan data.