Jurnal Sifo Mikroskil (JSM)
Volume 26, No 1, April 2025 – Hal 1 - 20
DOI: https://doi.org/10.55601/jsm.v26i1.1500

e-ISSN: 2622-8130
ISSN: 1412-0100

Pengembangan Aplikasi Deteksi Kematangan Buah Pisang Berbasis Web Menggunakan Model CNN-LSTM

Cindy Sintiya[1], Erin Gunawan[2], Dhea Romantika Marpaung[3], Farrell Rio Fa[4], Frans Mikael Sinaga*[5]
[1,2,3,4,5]Universitas Mikroskil, Jl. Thamrin No. 112, 124, 140, Telp. (061) 4573767
[1,2,3,4,5]Fakultas Informatika, Teknik Informatika, Universitas Mikroskil, Medan
e-mail: [1]211110347@students.mikroskil.ac.id, [2]211110101@students.mikroskil.ac.id,
[3]211110933@students.mikroskil.ac.id, [4]211111610@students.mikroskil.ac.id, [5]frans.sinaga@mikroskil.ac.id

Dikirim: ***03-02-2025*** | *Diterima:* ***17-03-2025*** | *Diterbitkan:* ***30-04-2025***

Abstrak

Klasifikasi tingkat kematangan buah merupakan salah satu tantangan dalam penerapan teknologi kecerdasan buatan di sektor pertanian. Penelitian ini mengusulkan sistem deteksi tingkat kematangan pisang menggunakan arsitektur *deep learning* berbasis *Convolutional Neural Network* (CNN) dan *Long Short-Term Memory* (LSTM). Dataset citra pisang diproses melalui tahapan *preprocessing* yang mencakup normalisasi, segmentasi warna (*mask* kuning dan hijau), serta deteksi tepi, untuk menonjolkan fitur visual yang relevan. Model yang diimplementasikan mampu mengklasifikasikan tingkat kematangan pisang ke dalam kategori "matang" dan "mentah". Sistem ini diintegrasikan dengan antarmuka berbasis *web* menggunakan Streamlit, memungkinkan prediksi dilakukan secara *real-time*. Hasil pengujian menunjukkan bahwa model mencapai akurasi 100% pada dataset uji, dengan *precision*, *recall*, dan *F1-score* sempurna. Penelitian ini membuktikan efektivitas pendekatan CNN-LSTM dalam klasifikasi tingkat kematangan buah yang diharapkan dapat membantu memberikan kontribusi terhadap otomatisasi di sektor pertanian.

Kata kunci: CNN-LSTM, Klasifikasi Kematangan Buah, Pembelajaran Mendalam

Abstract

Classifying fruit ripeness levels is a key challenge in applying artificial intelligence technology to the agricultural sector. This study proposes a banana ripeness detection system using a deep learning architecture based on Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). Banana image datasets are processed through a preprocessing pipeline, including normalization, color segmentation (to get yellow and green masks), and edge detection, to highlight relevant visual features. The implemented model classifies banana ripeness into "ripe" and "unripe" categories. The system is integrated with a web-based interface using Streamlit, enabling real-time predictions. Testing results show that the model achieves 100% accuracy on the test dataset, with perfect precision, recall, and F1-score. This study demonstrates the effectiveness of the CNN-LSTM approach for fruit ripeness classification which is expected to help contribute to automation in agriculture.

Keywords: *CNN-LSTM, Ripeness Classification, Deep Learning*

1. PENDAHULUAN

Kecerdasan buatan dan teknologi pembelajaran mendalam (*deep learning*) telah membawa revolusi dalam berbagai sektor, termasuk sektor pertanian [1]. Salah satu aplikasi *deep learning* dalam sektor pertanian yang berkembang pesat adalah deteksi kematangan buah, yang sangat berguna untuk

menentukan waktu panen yang optimal, meningkatkan kualitas hasil pertanian, dan mengurangi potensi pemborosan produk [2], [3]. Buah yang matang pada saat yang tepat memiliki nilai gizi dan kualitas terbaik, sementara buah yang dipanen atau didistribusikan dalam keadaan belum matang atau terlalu matang sering kali menimbulkan kerugian ekonomi [2].

Pisang, sebagai salah satu buah yang banyak dikonsumsi masyarakat, memiliki ciri khas siklus kematangan yang cepat. Hal ini menimbulkan tantangan tersendiri dalam menentukan tingkat kematangan yang tepat, terutama jika dilakukan secara manual. Proses manual ini sering kali kurang efisien dan membutuhkan keahlian khusus dalam mengidentifikasi tingkat kematangan berdasarkan tampilan visual buah, seperti warna, tekstur, dan bentuknya.

Oleh karena itu, diperlukan suatu teknologi yang dapat membantu mengklasifikasikan kematangan pisang dengan cepat dan akurat. Pendekatan *deep learning* memberikan peluang untuk menciptakan sistem deteksi otomatis yang mampu mengenali tingkat kematangan buah berdasarkan analisis citra visualnya. Dengan memanfaatkan dataset citra buah pisang pada berbagai tahap kematangan, model pembelajaran mendalam dapat dilatih untuk mengklasifikasikan kematangan dengan akurasi tinggi, yang nantinya dapat bermanfaat bagi petani, pengecer, dan konsumen.

Pendeteksian kematangan buah secara otomatis menggunakan algoritma *machine learning* maupun *deep learning* telah dilakukan dalam penelitian-penelitian terdahulu, seperti yang deteksi kematangan buah *cape gooseberry* atau *goldenberry* dengan pendekatan berbagai algoritma seperti ANN, Decision Tree, SVM, dan KNN pada citra buah yang telah dilakukan pemrosesan warna [2]. Penelitian tentang klasifikasi tingkat kematangan pisang juga dilakukan pada 45 citra pisang dengan metode KNN dan mencapai tingkat akurasi 88,89% [4]. Penelitian yang dilakukan oleh Saragih dan Emanuel menggunakan metode *deep learning* berbasis *pretrained model Convolutional Neural Network*, yakni *MobileNet V2* dan *NASNetMobile* untuk mengklasifikasikan pisang menjadi pisang mentah, matang, setengah matang, dan busuk. Akurasi model CNN *MobileNet V2* dalam penelitian tersebut mencapai 96,18% [3].

Pada penelitian ini, akan dilakukan pembangunan model dan evaluasi model klasifikasi tingkat kematangan pisang dengan metode gabungan CNN dan LSTM. Pemrosesan gambar digunakan untuk mengekstraksi fitur warna, kontur terbesar objek dan garis tepi objek. Fitur warna digunakan untuk mengklasifikasikan pisang menjadi "matang" dan "mentah", sementara kontur terbesar objek digunakan agar garis tepi bebas dari *noise* berupa bintik-bintik hitam dari pisang dan kepadatan garis tepi digunakan untuk mengenali inputan "bukan pisang" agar tidak diproses lebih lanjut oleh model.

Penggunaan CNN dalam penelitian ini berfungsi untuk mengekstraksi fitur spasial dari citra pisang, seperti warna dan tekstur [5]. Namun, model CNN konvensional tidak secara optimal menangkap hubungan antar fitur yang mungkin memiliki pola sekuensial dalam analisis kematangan pisang. Oleh karena itu, LSTM digunakan untuk menangkap dependensi temporal antar fitur yang diekstraksi oleh CNN. Arsitektur LSTM mampu mempertahankan informasi dari fitur yang relevan dalam jangka panjang, sehingga membantu model dalam memahami pola perubahan warna dan tekstur pisang yang terjadi secara bertahap selama proses pematangan. Kombinasi CNN dan LSTM memungkinkan model untuk tidak hanya mengenali karakteristik visual statis tetapi juga memahami pola kematangan dengan lebih baik [6, 7].

Selanjutnya, model yang dikembangkan akan diimplementasikan dalam aplikasi *web* berbasis Streamlit agar dapat diakses oleh pengguna secara langsung untuk mendeteksi kematangan buah pisang berdasarkan model yang telah dibangun.

2. TINJAUAN PUSTAKA

2.1 Pembelajaran Mendalam

Pembelajaran mendalam adalah cabang dari pembelajaran mesin yang berfokus pada algoritma yang mirip dengan versi yang sangat disederhanakan dari otak manusia, yang mampu menyelesaikan berbagai masalah dalam kecerdasan mesin modern. Banyak contoh umum dapat ditemukan dalam ekosistem aplikasi *smartphone* (iOS dan Android), seperti deteksi wajah di kamera, koreksi otomatis

dan teks prediktif pada papan ketik, aplikasi kecantikan yang ditingkatkan oleh AI, asisten pintar seperti Siri, Alexa, dan Google Assistant, serta fitur *Face ID*. Secara keseluruhan, pembelajaran mendalam telah menjadi bagian yang tak terpisahkan dari kehidupan digital kita saat ini [8].

Algoritma dalam pembelajaran mendalam menggunakan arsitektur jaringan saraf tiruan *(artificial neural networks)* yang terdiri dari banyak lapisan ANN *(hidden layers)*. Oleh karena banyak lapisan tersebut, maka algoritma disebut dengan dalam *(deep)*. Data diteruskan melalui laposan-lapisan tersebut dan setiap lapisan dapat melakukan ekstraksi fitur dan meneruskannya ke lapisan berikutnya. Lapisan awal digunakan untuk ekstraksi fitur tingkat rendah, sementara lapisan berikutnya menggabungkan fitur-fitur ini untuk membentuk hasil representasi yang lengkap [9].

2.2 Zero Padding

Zero padding adalah proses penambahan *pixel* bernilai 0 (nol) pada tepi input, dan ini adalah metode yang sederhana serta mudah untuk diterapkan. Dalam berbagai aplikasi *computer vision* seperti klasifikasi dan segmentasi, model jaringan saraf dalam menunjukkan kemampuan adaptasi yang baik terhadap penggunaan *zero padding*. Namun, model yang menggunakan *skip connections* sering kali mengalami penurunan kinerja karena lebih peka terhadap penggunaan algoritma *zero padding* [10].

2.3 Segmentasi

Segmentasi adalah proses mempartisi sebuah gambar utuh *R* menjadi *sub-region* R_1, R_2, …, R_n. Setiap *pixel* di dalam gambar R harus termasuk ke dalam satu *sub-region*. Proses segmentasi dilakukan ketika kita menginginkan pengolahan hanya pada *region* tertentu dalam sebuah gambar. Contoh kasus penggunaannya, untuk memisahkan antara objek *(foreground)* dengan latar belakang *(background)* [11].

Algoritma segmentasi pada gambar *grayscale* secara umum terbagi menjadi dua, yaitu algoritma berbasis diskontinuitas dan algoritma berbasis similaritas. Algoritma diskontinuitas mengasumsikan bahwa *region* dipisahkan oleh batasan yang jelas sehingga memanfaatkan perubahan intensitas lokal yang drastis untuk mengekstraksi batasan tersebut, misalnya segmentasi *edge-based*. Algoritma berbasis similaritas mengkategorikan gambar ke dalam *region* berdasarkan kemiripan intensitas satu *pixel* dengan *pixel* yang lain, misalnya *thresholding*, *region growing*, *region splitting,* dan *merging* [11].

2.4 Thresholding

Thresholding atau ambang batas merupakan sebuah pendekatan segmentasi yang melibatkan operator transformasi intensitas, yaitu level keabuan *(gray-level)*. *Thresholding* dapat digunakan dalam gambar yang terang pada latar belakang/ *background* yang gelap, yang mana nilai intensitas *(gray-level) pixel* dalam gambar tersebut dapat dikelompokkan menjadi dua nilai dominan. Nilai *threshold* T digunakan untuk memisahkan dua nilai dominan tersebut. Seluruh *pixel* dalam gambar input yang lebih besar dari T, disebut sebagai *object point*. Sebaliknya, seluruh *pixel* dalam gambar input yang lebih kecil sama dengan T, diistilahkan sebagai *background point*. [11]

Hasil proses *thresholding* dari gambar input f(x,y) menjadi gambar output g(x,y) dengan suatu nilai ambang batas T dapat ditulis sebagai berikut.

$$g(x,y) = \begin{cases} 1, jika\ f(x,y) > T \\ 0, jika\ f(x,y) \leq T \end{cases} \quad (1)$$

Jika suatu nilai T yang diaplikasikan pada seluruh *pixel* pada gambar ialah konstan, maka proses *thresholding* tersebut disebut *global thresholding*. Jika nilai T yang diaplikasikan pada suatu gambar berubah-ubah, *proses thresholding* tersebut disebut *variable thresholding*. Dalam *library* pemrosesan gambar OpenCV, *global thresholding* juga disebut sebagai *simple thresholding* dan *variable thresholding* disebut *adaptive thresholding*.

2.5 Edge Detection

Edge detection (deteksi tepi) merupakan salah satu aktivitas kunci dalam *computer vision* dan pemrosesan citra. Proses ini memberikan informasi krusial untuk berbagai tugas visual selanjutnya,

seperti *image recognition*, *image segmentation*, *image retrieval*, *face recognition*, *corner detection*, *road detection*, *medical imaging*, dan *target tracking*. Setiap aktivitas visual tersebut memerlukan pengenalan batas objek atau identifikasi tepi yang jelas dari gambar asli. Karena itu, penting untuk memastikan *Edge detection* yang stabil agar setiap tugas dapat dilakukan dengan efisien [12].

2.6 Deteksi Kontur

Kontur, atau batasan dari sebuah *region* R merupakan kumpulan *pixel* di dalam R yang bersebelahan dengan *pixel-pixel* yang berlawanan dengan R. Dengan kata lain, kontur dari sebuah objek adalah *pixel-pixel* dalam *region* objek yang memiliki paling tidak satu tetangga yang merupakan *background* [11]. Kontur juga dapat diartikan sebagai *edge* yang tertutup yang bersifat global.

Dalam pemrosesan citra, deteksi kontur terdiri dari mengidentifikasi batasan dari objek dalam suatu gambar. Oleh karena itu, citra yang telah melalui proses segmentasi untuk memisahkan objek menjadi citra biner terlebih dahulu akan menghasilkan deteksi kontur yang lebih akurat. Kontur dapat digunakan untuk analisa bentuk citra, deteksi objek dan pengenalan objek [13].

2.7 LSTM (Long-Short Term Memory)

Long-Short Term Memory (LSTM) adalah sebuah pengembangan lanjutan dari arsitektur *Recurrent Neural Network* (RNN), yang dirancang untuk meningkatkan kemampuan jaringan dalam menangani data sekuensial. Dengan menambahkan *memory cells* khusus, LSTM memungkinkan jaringan untuk menyimpan dan mengelola informasi dalam jangka waktu yang lebih lama, yang sebelumnya sulit dilakukan oleh RNN standar. *Memory cell* ini dilengkapi dengan mekanisme *gates* (pintu gerbang) yang mengatur aliran informasi secara cermat, sehingga data yang relevan dapat dipertahankan lebih lama, dan data yang tidak relevan dapat diabaikan. Selain itu, penambahan *memory cell* ini efektif dalam mengatasi permasalahan *vanishing gradient*, yang sering muncul dalam RNN ketika menangani data sekuensial yang panjang, sehingga membuat LSTM lebih stabil dan akurat dalam proses pembelajaran pada tugas-tugas yang membutuhkan pemahaman konteks jangka panjang [14].
Struktur *gates* pada arsitektur LSTM [15]:

1. *Forget Gate* (*ft*): mengambil output pada waktu $t-1$ dan input pada waktu t kemudian digabungkan serta menerapkan fungsi aktivasi sigmoid sehingga menghasilkan nilai 0 atau 1.
$$ft = \sigma(W_f \times S_{t-1} + W_f \times X_t) \quad (2)$$
Jika *ft = 0* (semakin mendekati 0), maka *state* sebelumnya akan di-*forget* (dilupakan), sementara jika *ft = 1*, maka *state* sebelumnya akan dipertahankan atau tidak berubah.
2. *Input Gate* (*it*): berperan mengambil input baru maupun output dari proses sebelumnya kemudian diteruskan ke lapisan sigmoid dan mengembalikan nilai 0 atau 1.
$$it = \sigma(W_i \times S_{t-1} + W_i \times X_t) \quad (3)$$
$$\tilde{C}_t = \tanh(W_c \times S_{t-1} + W_c \times X_t) \quad (4)$$
Lakukan *update state* dengan mengkombinasikan *state* sebelumnya dengan *state candidate*.
$$c_t = (it \times \tilde{C}_t + f_t \times c_{t-1}) \quad (5)$$
3. *Output Gate* (*ot*): mengontrol seberapa banyak *state* yang lewat dan menghasilkan *state* baru (*ht*).
$$ot = \sigma(W_o \times S_{t-1} + W_o \times X_t) \quad (6)$$
$$ht = ot * tanh(ct) \quad (7)$$

2.8 CNN (Convolutional Neural Networks)

Convolutional Neural Networks (CNN) adalah jenis model pembelajaran mesin yang sangat efektif dalam mengenali pola-pola dalam data visual, terutama berkat inspirasi dari struktur dan fungsi dasar sistem visual biologis. CNN mengandalkan konektivitas lokal dan bobot yang dibagikan dalam lapisan konvolusi, berbeda dengan jaringan saraf konvensional di mana setiap neuron terhubung ke semua neuron pada lapisan sebelumnya. Pada dasarnya, CNN terdiri dari unit-unit pemrosesan seperti neuron yang terhubung dalam lapisan-lapisan berurutan, mirip dengan area-area di otak. Melalui operasi

konvolusi, CNN dapat menangkap pola-pola spasial dalam data melalui filter yang memiliki bobot dapat dilatih. Proses ini memungkinkan jaringan untuk mempelajari hirarki abstraksi lapisan awal mendeteksi fitur dasar, sedangkan lapisan yang lebih dalam menangkap pola yang lebih kompleks [16].

2.9 Pengujian Algoritma

Pengujian algoritma dilakukan untuk mengukur kinerja model yang telah dibangun. Hal ini dapat dilakukan dengan *performance metrics*. Beberapa metrik yang umum digunakan, yakni [17], [18]:

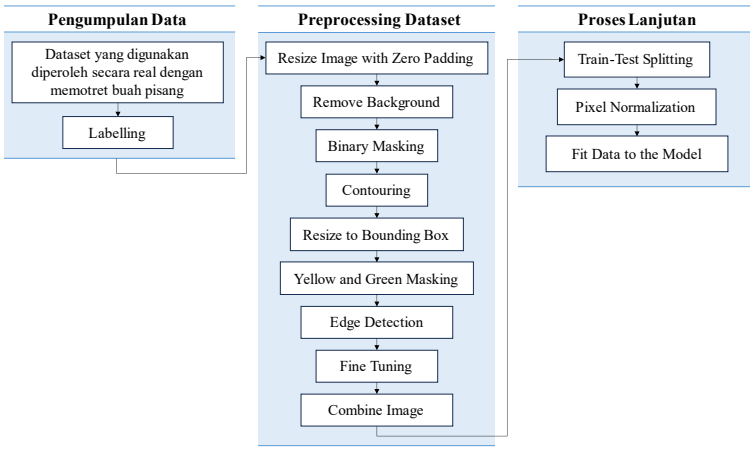
1. *Accuracy*, yaitu mengukur seberapa tepat/ akurat model dapat mengklasifikasikan input dengan benar. Untuk menghitung *accuracy* digunakan rumus:
$$accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (8)$$
2. *Precision*, yaitu mengukur seberapa reliabilitas model ketika model mengklasifikasikan sesuatu sebagai "Positif". Untuk menghitung *precision* digunakan rumus:
$$precision = \frac{TP}{TP+FP} \quad (9)$$
3. *Recall*, yaitu mengukur keberhasilan model dalam menemukan kembali sebuah informasi. Untuk menghitung *recall* digunakan rumus:
$$recall = \frac{TP}{TP+FN} \quad (10)$$
4. *F1-score,* yaitu perbandingan rata-rata *precision* dan *recall*. Untuk menghitung *f1-score* digunakan rumus:
$$F1\ score = \frac{2\times recall\times precision}{recall+precision} \quad (11)$$

3. METODE PENELITIAN

3.1 Dataset dan Preprocessing

Dataset yang digunakan diperoleh melalui proses pemotretan langsung terhadap buah pisang dalam kondisi nyata. Pengambilan gambar dilakukan pada satuan individu pisang (bukan dalam bentuk sisir atau tandan). Jumlah dataset awal yang terkumpul terdiri dari 27 citra buah pisang matang dan 27 citra buah pisang mentah. Namun, untuk meningkatkan jumlah data guna mendukung pelatihan model yang lebih optimal, dilakukan penambahan dataset dengan memperoleh citra pisang dari sumber daring, seperti Google dan Kaggle. Dari hasil pencarian ini, diperoleh tambahan sebanyak 27 citra pisang matang dan 9 citra pisang mentah. Dengan demikian, total dataset yang digunakan dalam penelitian ini terdiri dari 54 citra pisang matang dan 36 citra pisang mentah.

Citra yang terkumpul akan melalui tahapan *preprocessing* sebelum menjadi input pada model. Alur *preprocessing* citra secara garis besar terlihat pada Gambar 1.



Gambar 1. Alur *Preprocessing* Dataset

3.1.1 Menyeragamkan Ukuran Citra dengan *Zero Padding*

Tahapan pertama dalam *preprocessing* adalah menyamakan ukuran citra input. Metode *zero padding* dipilih untuk memastikan bahwa citra yang diinput tetap mempertahankan ukuran dan bentuk aslinya, serta menghindari distorsi bentuk objek dan menjaga proporsi objek sehingga nantinya objek akan lebih mudah dianalisis. Dengan cara ini, citra utama tetap berada di tengah, sementara area yang tidak mencukupi ukuran yang ditetapkan akan diisi dengan *pixel* bernilai nol, sehingga *background* tidak mengganggu objek utama. Di tahap ini, semua ukuran citra akan distandarisasi menjadi 500x500 *pixel*.

CITRA ASLI (5x4)				
119	2	5	19	28
87	96	47	36	254
13	59	50	42	9
10	34	35	0	199

CITRA TARGET (7x7)						
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Gambar 2. Ukuran Citra Asli vs Citra Target

Sebagai contoh, misalkan citra awal berukuran 5x4 dengan ukuran citra target 7x7, seperti yang terlihat pada Gambar 3. Ukuran citra yang diinput masih belum sesuai target, sehingga akan dilakukan penambahan *zero padding* dengan menggunakan rumus sebagai berikut:

$$TopPad = \left\lfloor \frac{TargetHeight - InputHeight}{2} \right\rfloor (Round\ to\ bottom) \quad (12)$$

$$BottomPad = \left\lceil \frac{TargetHeight - InputHeight}{2} \right\rceil (Round\ to\ top) \quad (13)$$

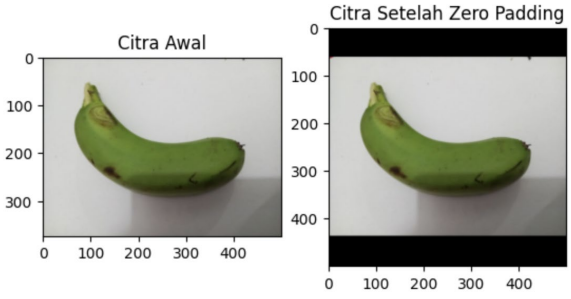
$$LeftPad = \left\lfloor \frac{TargetWidth - InputWidth}{2} \right\rfloor (Round\ to\ bottom) \quad (14)$$

$$RightPad = \left\lceil \frac{TargetWidth - InputWidth}{2} \right\rceil (Round\ to\ top) \quad (15)$$

Padding akan ditambahkan pada baris/ kolom terluar dengan jumlah/ ukuran sesuai hasil perhitungan yang didapatkan dari rumus diatas sehingga tampak seperti Gambar 3.

CITRA ASLI (5x4)				
119	2	5	19	28
87	96	47	36	254
13	59	50	42	9
10	34	35	0	199

ZERO PADDING (7x7)						
0	0	0	0	0	0	0
0	119	2	5	19	28	0
0	87	96	47	36	254	0
0	13	59	50	42	9	0
0	10	34	35	0	199	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0



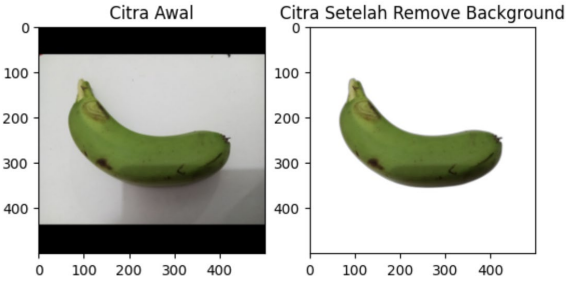
Gambar 3. Penambahan *Zero Padding* pada Citra

3.1.2 Menghilangkan *Background* Citra

Setelah ukuran citra diseragamkan, *background* citra dihilangkan untuk menonjolkan objek utama. Proses ini bertujuan agar objek dapat diekstrak dengan lebih akurat tanpa adanya gangguan dari elemen-elemen *background* yang tidak diperlukan. Teknik *background removal* ini menggunakan metode segmentasi yang secara otomatis mendeteksi dan menghapus latar belakang, sehingga objek utama akan tetap menjadi satu-satunya fokus dalam citra.

CITRA DENGAN ZERO PADDING (7x7)						
0	0	0	0	0	0	0
0	119	2	5	19	28	0
0	87	96	47	36	254	0
0	13	59	50	42	9	0
0	10	34	35	0	199	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

REMOVE BACKGROUND (7x7)						
0	0	0	0	0	0	0
0	119	0	0	0	28	0
0	87	96	47	36	0	0
0	0	59	50	42	0	0
0	0	34	35	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0



Gambar 4. Penghapusan *Background* Citra

3.1.3 Menerapkan *Binary Masking/ Thresholding*

Setelah *background* dihapus, gambar diubah menjadi *grayscale* untuk memudahkan pemrosesan *threshold*. *Grayscale* digunakan agar hanya informasi luminans (tingkat kecerahan) yang dipertahankan. Pada citra *grayscale*, *Gaussian Blur* kemudian diaplikasikan untuk mereduksi *noise* yang ada, sehingga detail yang tidak relevan tidak akan mengganggu proses selanjutnya.

Sebagai contoh, perhitungan nilai *gray* akan menggunakan rumus sebagai berikut:

$$Gray = 0.299 \times Red + 0.587 \times Green + 0.114 \times Blue \quad (16)$$

Namun dalam contoh perhitungan sederhana menggunakan 1 *channel* warna, maka kita anggap hanya menggunakan *channel Green*. Tiap nilai *pixel* akan dikalikan dengan 0.587 yang kemudian hasilnya dapat dilihat pada Gambar 5.

CITRA HASIL BACKGROUND REMOVAL (7x7)						
0	0	0	0	0	0	0
0	119	0	0	0	28	0
0	87	96	47	36	0	0
0	0	59	50	42	0	0
0	0	34	35	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

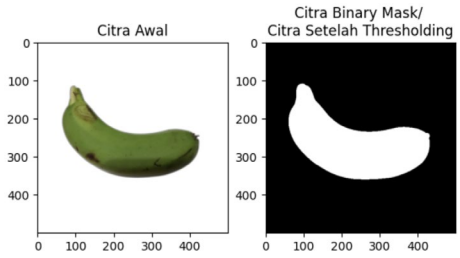
GRAYSCALE (7x7)						
0	0	0	0	0	0	0
0	69,85	0	0	0	16,44	0
0	51,07	56,35	27,59	21,13	0	0
0	0	34,63	29,35	24,65	0	0
0	0	19,96	20,55	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Gambar 5. Membuat Citra *Grayscale*

Setelah itu, *threshold* diterapkan untuk membuat citra biner, agar objek terpisah dari *background* secara tegas. *Threshold* dilakukan dengan fungsi opencv *threshold* dengan dengan tipe *cv::THRESH_BINARY*, parameter nilai *thresh* = 0 dan parameter nilai *maxval* = 255. Hasil *threshold* yaitu *pixel* objek yang intensitasnya > *thresh* menjadi *maxval* (255) yang merepresentasikan objek, sedangkan *pixel* yang intensitasnya <= *thresh* menjadi 0 yang merepresentasikan latar belakang.

CITRA GRAYSCALE (7x7)						
0	0	0	0	0	0	0
0	69,85	0	0	0	16,44	0
0	51,07	56,35	27,59	21,13	0	0
0	0	34,63	29,35	24,65	0	0
0	0	19,96	20,55	0,00	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

BINARY MASKING (7x7)						
0	0	0	0	0	0	0
0	255	0	0	0	255	0
0	255	255	255	255	0	0
0	0	255	255	255	0	0
0	0	255	255	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0



Gambar 6. Hasil *Mask Binary*

3.1.4 Ekstraksi Kontur Terbesar

Tahap berikutnya adalah mencari kontur terbesar pada citra. Kontur terbesar ini diasumsikan sebagai objek utama dalam gambar, mengingat objek utama biasanya memiliki area paling luas. Dengan hanya memproses kontur terbesar, analisis akan difokuskan pada objek utama tanpa gangguan dari kontur-kontur lain yang lebih kecil.

Contoh ekstraksi kontur pada citra persegi panjang tampak seperti Gambar 7, dengan *pixel* yang diberi warna kuning merepresentasikan kontur citra.

CITRA AWAL (7x7)						
0	0	0	0	0	0	0
0	255	255	255	255	255	0
0	255	255	255	255	255	0
0	255	255	255	255	255	0
0	255	255	255	255	255	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

TITIK KONTUR (SIMPLE)						
0	0	0	0	0	0	0
0	255	255	255	255	255	0
0	255	255	255	255	255	0
0	255	255	255	255	255	0
0	255	255	255	255	255	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Gambar 7. Hasil Mencari Kontur Terbesar pada Citra Segiempat

3.1.5 Menerapkan *Bounding Box* pada Area Objek

Setelah kontur terbesar ditemukan, dicari *bounding box* dengan fungsi *opencv boundingRect*, berupa segiempat yang mengelilingi kontur terbesar. Fungsi ini mengembalikan koordinat titik x, y pada gambar asal, panjang *bounding box* dari titik tersebut, serta lebar *bounding box* dari titik tersebut. Citra kemudian dipotong (*crop*) berdasarkan *bounding box* persegi dari kontur tersebut, sehingga area citra berpusat pada objek utama. Langkah ini memastikan mengisolasi *Region of Interest* (ROI) atau objek utama dalam satu *frame* dengan ukuran yang sesuai untuk analisis. ROI juga akan digunakan dalam menghitung kepadatan tepi yang berperan dalam mendeteksi obyek "bukan pisang" dalam citra agar tidak diproses lebih lanjut.

REMOVE BACKGROUND (7x7)						
0	0	0	0	0	0	0
0	119	0	0	0	28	0
0	87	96	47	36	0	0
0	0	59	50	42	0	0
0	0	34	35	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

BINARY MASKING (7x7)						
0	0	0	0	0	0	0
0	255	0	0	0	255	0
0	255	255	255	255	0	0
0	0	255	255	255	0	0
0	0	255	255	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

BOUNDING BOX (5x4)				
255	0	0	0	255
255	255	255	255	0
0	255	255	255	0
0	255	255	0	0

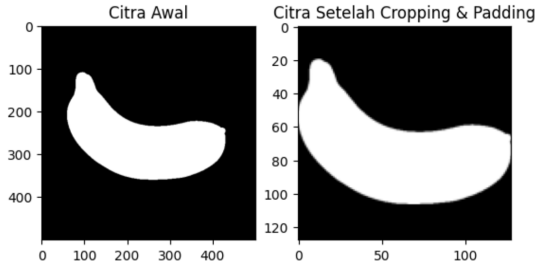
CROP ON LARGEST CONTOUR (5x4)				
119	0	0	0	28
87	96	47	36	0
0	59	50	42	0
0	34	35	0	0

Gambar 8. Memotong Citra pada Area Kontur Terbesar

Setelah itu, ukuran citra hasil *cropping* ini distandarisasi kembali menjadi 128x128 *pixel* dengan memanfaatkan *zero padding*, memastikan objek utama tetap berada dalam ukuran dan posisi yang konsisten untuk pemrosesan lebih lanjut. Contoh pengaplikasian *zero padding* pada citra hasil *crop* berukuran 5x4 *pixel* menjadi citra berukuran 5x5 *pixel* tampak seperti Gambar 9.

CITRA DENGAN LARGEST CONTOUR (5x4)				
119	0	0	0	28
87	96	47	36	0
0	59	50	42	0
0	34	35	0	0

ZERO PADDING (5X5)				
119	0	0	0	28
87	96	47	36	0
0	59	50	42	0
0	34	35	0	0
0	0	0	0	0



Gambar 9. Menambahkan *Zero Padding* pada Citra

3.1.6 Membuat *Yellow* dan *Green Mask* untuk Citra

Dalam tahap ini, *mask* berwarna kuning dan hijau dibuat untuk memisahkan tingkat kematangan objek buah pisang berdasarkan spektrum warnanya. *Yellow mask* digunakan untuk mendeteksi area yang sudah matang, sedangkan *green mask* untuk mendeteksi area yang masih mentah. Dengan adanya dua jenis *masking* ini, model dapat mempelajari perbedaan tingkat kematangan objek pisang berdasarkan distribusi warna, yang dapat membantu menentukan klasifikasi kematangan buah pisang tersebut.

Sebagai contoh, *threshold* yang digunakan untuk menandai *yellow masking* berada pada *range* 28-50 dan *green masking* pada *range* 51-120.

Yellow Mask:

$$if\ PIXEL < 28\ or\ PIXEL > 50\ then\ PIXEL = 0 \quad (17)$$

Green Mask:

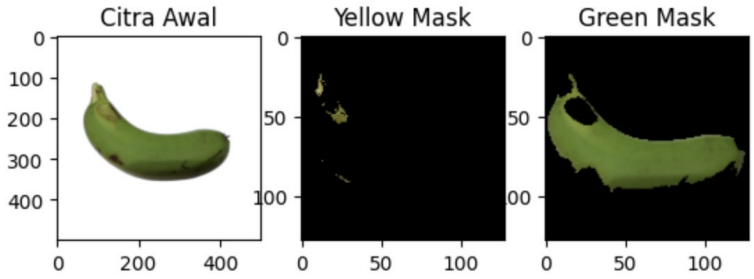
$$if\ PIXEL < 51\ or\ PIXEL > 120\ then\ PIXEL = 0 \quad (18)$$

Semua *pixel* yang bernilai antara 28 hingga 50 akan dipertahankan dan nilai *pixel* lainnya akan dijadikan *yellow mask* dengan nilai 0. Untuk *green mask*, semua *pixel* yang bernilai antara 51 hingga 120 akan dipertahankan dan nilai *pixel* lainnya akan diubah menjadi 0.

CITRA ROI + ZERO PADDING				
119	0	0	0	28
87	96	47	36	0
0	59	50	42	0
0	34	35	0	0
0	0	0	0	0

YELLOW MASK (5X5)				
0	0	0	0	28
0	0	47	36	0
0	0	50	42	0
0	34	35	0	0
0	0	0	0	0

GREEN MASK (5X5)				
119	0	0	0	0
87	96	0	0	0
0	59	0	0	0
0	0	0	0	0
0	0	0	0	0



Gambar 10. Membuat *Yellow* dan *Green Masking*

3.1.7 Deteksi Tepi pada Objek dalam Citra

Setelah proses *masking* warna, deteksi tepi (*edge detection*) diterapkan pada citra. Metode ini bertujuan untuk mengidentifikasi batas-batas atau tepi pada objek utama dan menghasilkan citra tepi yang menunjukkan pola tekstur pada objek. Teknik ini memberikan informasi tambahan tentang tekstur permukaan objek, yang nantinya bisa digunakan untuk mengidentifikasi karakteristik tertentu yang mungkin relevan dalam analisis kematangan atau pengenalan objek.

Sebagai contoh, menggunakan citra yang sudah bersih dari tahapan sebelumnya, konvolusi dilakukan dengan kernel Sobel.

CLEANED IMAGE (5X5)				
119	0	0	0	28
87	96	47	36	0
0	59	50	42	0
0	34	35	0	0
0	0	0	0	0

KERNEL SOBEL HORIZONTAL (Gx) (3x3)		
-1	0	1
-2	0	2
-1	0	1

KERNEL SOBEL VERTIKAL (Gy) (3x3)		
-1	-2	-1
0	0	0
1	2	1

Gambar 11. Citra 5x5 dan Kernel Sobel 3x3

POTONGAN CITRA				KERNEL SOBEL HORIZONTAL (Gx) (3x3)				HASIL KONVOLUSI				
119	0	0		-1	0	1		-1 x 119	0 x 0	1 x 0		
87	96	47	x	-2	0	2	=	-2 x 87	0 x 96	2 x 47	=	−149
0	59	50		-1	0	1		-1 x 0	0 x 59	1 x 50		

POTONGAN CITRA				KERNEL SOBEL VERTIKAL (Gy) (3x3)				HASIL KONVOLUSI				
119	0	0		-1	-2	-1		-1 x 119	-2 x 0	-1 x 0		
87	96	47	x	0	0	0	=	0 x 87	0 x 96	0 x 47	=	49
0	59	50		1	2	1		1 x 0	2 x 59	1 x 50		

Gambar 12. Konvolusi Potongan Citra 3x3 dengan Kernel Horizontal dan Vertikal

Kemudian hitung besaran *gradient*:

$$G = \sqrt{Gx^2 + Gy^2} \tag{19}$$

$$G = \sqrt{(-149)^2 + (49)^2} \approx 156{,}88$$

Hitung juga arah dari *gradient* menggunakan fungsi *Arctangen (Tan inverse)*:

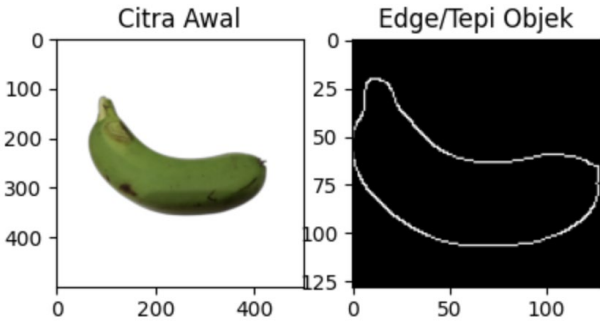
$$\theta = \tan^{-1}\left(\frac{Gy}{Gx}\right) \tag{20}$$

$$\theta = \tan^{-1}\left(\frac{49}{-149}\right) \approx -18.26°$$

Thresholding akan diterapkan pada *range* 150 – 200 untuk mengambil *pixel* yang akan dijadikan *edge* untuk diubah ke nilai 255 dan *pixel* yang tidak memenuhi *threshold* akan diubah menjadi 0.

CITRA ROI + ZERO PADDING				
119	0	0	0	28
87	96	47	36	0
0	59	50	42	0
0	34	35	0	0
0	0	0	0	0

EDGE DETECTION (5x5)				
255	0	0	0	255
255	255	255	255	0
0	255	0	255	0
0	255	255	0	0
0	0	0	0	0



Gambar 13. Mendeteksi Tepi Objek dalam Citra

3.1.8 Menghitung Kepadatan Tepi pada Objek

Setelah deteksi tepi, kepadatan tepi pada area objek juga perlu dihitung. Kepadatan ini diukur untuk memahami sejauh mana objek memiliki karakteristik tekstur yang sesuai dengan kriteria yang diharapkan. Kepadatan tepi yang terlalu rendah atau terlalu tinggi dapat menunjukkan bahwa objek tersebut bukanlah pisang atau tidak memenuhi syarat untuk dilanjutkan ke tahap analisis lebih lanjut. Hanya objek yang memenuhi rentang kepadatan tepi yang diinginkan yang akan diproses ke tahap berikutnya, yaitu klasifikasi kematangan oleh model.

Kepadatan tepi dapat dihitung dengan:

$$Edge\ Density = \frac{Total\ Edge\ Pixels}{Total\ Pixels\ in\ ROI} \tag{21}$$

Sebagai contoh, hitung jumlah *pixel* yang ditandai sebagai *edge* (*pixel* bernilai 255 pada citra) dan jumlah *pixel* objek pada citra *cropped* hasil perhitungan kontur terbesar (bernilai lebih besar dari 0). Gunakan *thresholding* untuk menentukan apakah kekuatan tepi berada dalam *range* 15% – 20% yang akan disebut sebagai pisang.

$$Edge\ Density = \frac{10}{11} \approx 0{,}91 = 91\%$$

Dari contoh diatas, karena kepadatan tepi diluar dari *threshold* yang sudah ditentukan, maka objek dalam citra tersebut akan dikatakan sebagai "bukan pisang".

3.1.9 Penggabungan Citra

Pada tahap akhir, tiga lapisan gambar yang berisi *yellow mask*, *green mask*, dan *edge* ditumpuk menjadi satu *array* 3 dimensi (128x128x3) yang siap dianalisis oleh model dengan fitur warna dan tepi yang kaya, sehingga dapat membantu proses klasifikasi atau deteksi lebih lanjut.

YELLOW MASK (5X5)				
0	0	0	0	28
0	0	47	36	0
0	0	50	42	0
0	34	35	0	0
0	0	0	0	0

+

GREEN MASK (5X5)				
119	0	0	0	0
87	96	0	0	0
0	59	0	0	0
0	0	0	0	0
0	0	0	0	0

+

EDGE DETECTION (5x5)				
255	0	0	0	255
255	255	255	255	0
0	255	0	255	0
0	255	255	0	0
0	0	0	0	0

2622-8130 Jurnal Sifo Mikroskil (JSM) - Volume 26, No 1, April 2025
DOI: https://doi.org/10.55601/jsm.v26i1.1500

STACKED IMAGE (5x5x3)				
(0, 119, 255)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(28, 0, 255)
(0, 87, 255)	(0, 96, 255)	(47, 0, 255)	(36, 0, 255)	(0, 0, 0)
(0, 0, 0)	(0, 59, 255)	(50, 0, 0)	(42, 0, 255)	(0, 0, 0)
(0, 0, 0)	(34, 0, 255)	(35, 0, 255)	(0, 0, 0)	(0, 0, 0)
(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)

Gambar 14. Menggabungkan *Yellow* dan *Green Mask* dengan *Edge* dari Citra

3.1.10 Normalisasi *Pixel*

Pixel Normalization merupakan langkah penting dalam pemrosesan citra untuk memastikan bahwa input yang diberikan ke model berada dalam rentang nilai yang konsisten dan seragam. Proses ini mengubah nilai *pixel* citra agar berada dalam rentang tertentu, seperti 0 hingga 1, yang merupakan standar umum dalam pelatihan model *deep learning*.

NORMALIZED YELLOW MASK (5X5)				
0,000	0,000	0,000	0,000	0,110
0,000	0,000	0,184	0,141	0,000
0,000	0,000	0,196	0,165	0,000
0,000	0,133	0,137	0,000	0,000
0,000	0,000	0,000	0,000	0,000

+

NORMALIZED GREEN MASK (5X5)				
0,467	0,000	0,000	0,000	0,000
0,341	0,376	0,000	0,000	0,000
0,000	0,231	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000
0,000	0,000	0,000	0,000	0,000

+

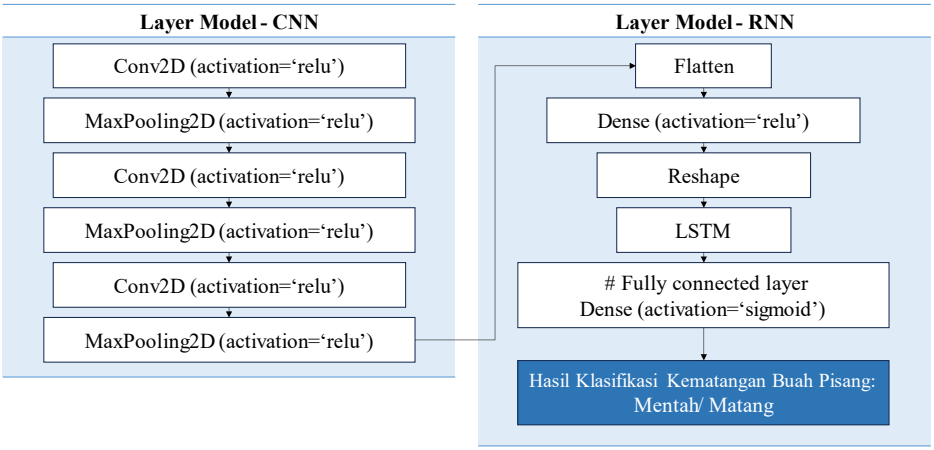
NORMALIZED EDGE (5x5)				
1,0	0,0	0,0	0,0	1,0
1,0	1,0	1,0	1,0	0,0
0,0	1,0	0,0	1,0	0,0
0,0	1,0	1,0	0,0	0,0
0,0	0,0	0,0	0,0	0,0

STACKED IMAGE (5x5x3)				
(0,00, 0,47, 1,00)	(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)	(0,11, 0,00, 1,00)
(0,00, 0,34, 1,00)	(0,00, 0,38, 1,00)	(0,18, 0,00, 1,00)	(0,14, 0,00, 1,00)	(0,00, 0,00, 0,00)
(0,00, 0,00, 0,00)	(0,00, 0,23, 1,00)	(0,20, 0,00, 0,00)	(0,16, 0,00, 1,00)	(0,00, 0,00, 0,00)
(0,00, 0,00, 0,00)	(0,13, 0,00, 1,00)	(0,14, 0,00, 1,00)	(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)
(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)	(0,00, 0,00, 0,00)

Gambar 15. Normalisasi Nilai *Pixel* Citra dalam *Range* 0-1

Tanpa normalisasi, perbedaan skala antara *pixel* yang memiliki nilai sangat besar (0–255) dan nilai yang lebih kecil dapat menyebabkan masalah dalam proses pelatihan, seperti konvergensi yang lambat atau terjadinya masalah pada gradien (misalnya *vanishing gradient* ataupun *exploding gradients*). Dengan memastikan semua input citra memiliki rentang yang seragam, normalisasi membantu mempercepat konvergensi model, meningkatkan stabilitas pelatihan, serta memungkinkan model untuk belajar dengan lebih efektif dari data yang diberikan.

3.2 Arsitektur Model



Gambar 16. Bagan Lapisan Model C-LSTM

1. Conv2D

Input : *batch* dataset berisi gambar 2 dimensi berukuran 128x128 *pixel* dan 9 *channel* (3 *yellow mask*, 3 *green mask*, 3 *edge mask*).

Output : 126x126 *pixel* hasil konvolusi dengan 64 buah kernel 3x3 untuk mengekstrak fitur *spatial* dari gambar yang diinput.

Berikut contoh perhitungan konvolusi pada 1 *channel* input:

NORMALIZED YELLOW MASK (5X5)						KERNEL 3X3				PERHITUNGAN KONVOLUSI				HASIL		
0,467	0,000	0,000	0,000	0,000												
0,341	0,376	0,000	0,000	0,000	x	0,2	0,1	0,2	=	0,2 x 0,47	0,1 x 0,00	0,2 x 0,00	=	0,1396	0,0231	0
0,000	0,231	0,000	0,000	0,000		0	0	0		0 x 0,34	0 x 0,38	0 x 0,00		0,1058	0,0752	0
0,000	0,000	0,000	0,000	0,000		0,1	0,2	0,1		0,1 x 0,00	0,2 x 0,23	0,1 x 0,00		0,0231	0,0462	0
0,000	0,000	0,000	0,000	0,000												

Gambar 17. *Layer* 1 - *Convolution*

2. MaxPooling2D

Output : mengurangi sebagian ukuran gambar menjadi 63x63 *pixel* dan memilih nilai *pixel* tertinggi *(max-pooling)* dengan ukuran kernel 2x2.

INPUT		
0,1396	0,0231	0
0,1058	0,0752	0
0,0231	0,0462	0

MAX POOLING (1)		
0,1396	0,0231	0
0,1058	0,0752	0
0,0231	0,0462	0

MAX POOLING (2)		
0,1396	0,0231	0
0,1058	0,0752	0
0,0231	0,0462	0

HASIL	
0,1396	0
0,0462	0

MAX POOLING (3)		
0,1396	0,0231	0
0,1058	0,0752	0
0,0231	0,0462	0

MAX POOLING (4)		
0,1396	0,0231	0
0,1058	0,0752	0
0,0231	0,0462	0

Gambar 18. *Layer* 2 - *Max Pooling*

3. Conv2D

Output : 61x61 *pixel* hasil konvolusi dengan 256 buah kernel 3x3 untuk mengekstrak fitur kompleks dari gambar.

4. MaxPooling2D

Output : mengurangi sebagian ukuran gambar menjadi 30x30 *pixel* dan memilih nilai *pixel* tertinggi dengan ukuran kernel 2x2.

5. Conv2D

Output : 28x28 *pixel* hasil konvolusi dengan 256 buah kernel 3x3 untuk membuat model mempelajari lebih banyak fitur dan pola yang kompleks.

6. MaxPooling2D

Output : mengurangi sebagian ukuran gambar menjadi 14x14 *pixel* dan memilih nilai *pixel* tertinggi dengan ukuran kernel 2x2.

7. Flatten

Output : mengubah ukuran gambar dari 3 dimensi menjadi 1 dimensi berukuran 50.176 untuk diteruskan ke *fully connected layer*.

YELLOW			GREEN			EDGE			FLATTENED												
0,1396	0	+	0	0,0928	+	0	1	=	0,1396	0	0	0	0,0928	1	0,0462	0,0057	1	0	0	1
0,0462	0		0,0057	0		1	1													

Gambar 19. *Array* 1 Dimensi Hasil *Flatten*

8. Dense

Output : mengurangi dimensi gambar dari 50.176 menjadi sebesar nilai parameter *units* yang ditentukan pada saat inisialisasi *layer* dense dan mempersiapkan data untuk proses sekuensial di *layer* LSTM.

9. Reshape

Output : mengubah ukuran gambar 1 dimensi menjadi 2 dimensi untuk dimasukkan sekaligus ke *layer* LSTM dengan 128 fitur.

10. LSTM
Output : *vector* berukuran 64 untuk menangani informasi sekuensial fitur.
11. Dense (Output; *Fully Connected Layer*)
Output : menggunakan fungsi aktivasi sigmoid untuk menghasilkan prediksi/ klasifikasi *binary* bernilai 0 atau 1.

3.3 Pelatihan dan Validasi

Dataset yang sudah melalui tahapan *preprocessing* akan dibagi menjadi 2 bagian dengan rasio 7:3, dimana 70% dataset akan digunakan dalam proses *training* data dan sisanya 30% (`test_size=0.3`) akan digunakan untuk *validation* dan *testing* model yang sudah dilatih. Proses pengacakan data dilakukan dengan menggunakan parameter `random_state=42` untuk memastikan bahwa setiap percobaan menghasilkan distribusi data yang konsisten.

```
x_train, x_test, y_train, y_test = train_test_split(
  x, y,
  test_size=0.3,
  random_state=42
)
```

Gambar 20. *Train-Test Splitting*

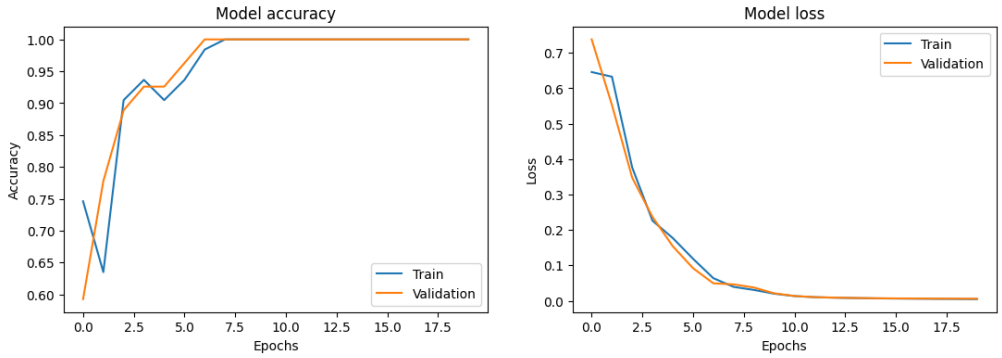
Proses *training* akan dilakukan selama 20 *epochs*, dimana masing-masing *epoch* akan memproses data dengan ukuran *batch* 2 data yang disesuaikan dengan jumlah dataset yang sudah dikumpulkan.

```
Epoch 1/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 177ms/step - accuracy: 0.7161 - loss: 0.6612 - val_accuracy: 0.5926 - val_loss: 0.7379
Epoch 2/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step - accuracy: 0.6108 - loss: 0.6664 - val_accuracy: 0.7778 - val_loss: 0.5527
Epoch 3/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 88ms/step - accuracy: 0.8948 - loss: 0.3993 - val_accuracy: 0.8889 - val_loss: 0.3473
Epoch 4/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - accuracy: 0.9473 - loss: 0.2209 - val_accuracy: 0.9259 - val_loss: 0.2366
                                    ...
Epoch 18/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 1.0000 - loss: 0.0054 - val_accuracy: 1.0000 - val_loss: 0.0060
Epoch 19/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - accuracy: 1.0000 - loss: 0.0050 - val_accuracy: 1.0000 - val_loss: 0.0059
Epoch 20/20
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step - accuracy: 1.0000 - loss: 0.0047 - val_accuracy: 1.0000 - val_loss: 0.0057
```

Gambar 21. Proses *Training* Dataset

Metrik pengukuran nilai *loss* akan menggunakan *Binary Cross Entropy* karena jumlah kelas yang akan diprediksi bersifat *binary* yang terdiri dari label matang dan mentah. Metrik pengukuran lain yang digunakan adalah akurasi, untuk menentukan apakah model sudah cukup tepat dalam menebak/ mengklasifikasi.

Detail nilai *loss* dan akurasi *training* dapat dilihat pada grafik dibawah ini:



Gambar 22. Perbandingan Nilai *Loss* dan Akurasi pada Mode *Training* dan Validasi

Nilai *loss* semakin menurun seiring *epoch* dan akurasi model juga meningkat secara signifikan mulai dari *epoch* ke 5 dan 6. Melihat garis naik turunnya akurasi dan *loss* model pada saat *training* dan validasi tidak terlalu jauh berbeda menandakan bahwa model tidak *underfitting* maupun *overfitting*.

4. HASIL DAN PEMBAHASAN

4.1 Hasil Pengujian

Hasil pengujian model yang dikembangkan akan disajikan untuk mengevaluasi performanya pada data *testing*. Model diuji menggunakan data *testing* yang terpisah dari data *training* untuk memastikan hasil pengujian yang objektif dan tidak bias.

4.1.1 Hasil akurasi dan *loss* model

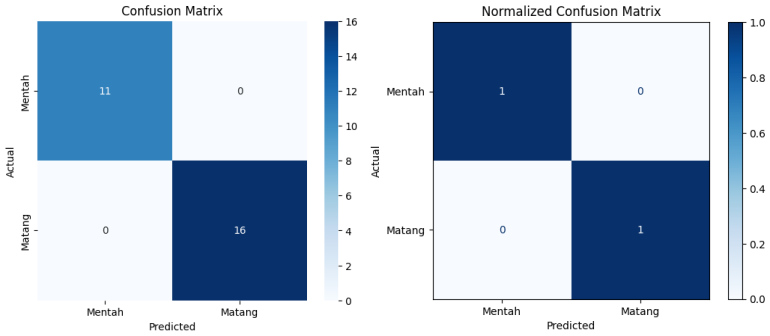
Model yang diimplementasikan menunjukkan akurasi sebesar 100% dengan nilai *loss* 0.0061 pada data *testing*. Hal ini mengindikasikan bahwa model berhasil mengenali pola dan mengklasifikasikan tingkat kematangan pisang dengan sangat baik tanpa adanya kesalahan prediksi.

```
14/14 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 1.0000 - loss: 0.0061
```

Gambar 23. Evaluasi *Loss* dan Akurasi pada Mode *Testing*

4.1.2 *Confusion Matrix*

Hasil pengujian *ditampilkan* dalam *confusion matrix* berikut:



Gambar 24. *Confusion Matrix* pada Data *Testing*

Dari 27 data *testing*, di mana 11 data berlabel mentah dan 16 data berlabel matang, berhasil diklasifikasikan oleh model dengan tepat untuk semua data yang ada.

4.1.3 Analisis Kinerja Model

	Mentah	Matang	accuracy	macro avg	weighted avg
precision	1.0	1.0	1.0	1.0	1.0
recall	1.0	1.0	1.0	1.0	1.0
f1-score	1.0	1.0	1.0	1.0	1.0
support	11.0	16.0	1.0	27.0	27.0

Gambar 25. Metrik Pengujian Analisis Kinerja Model

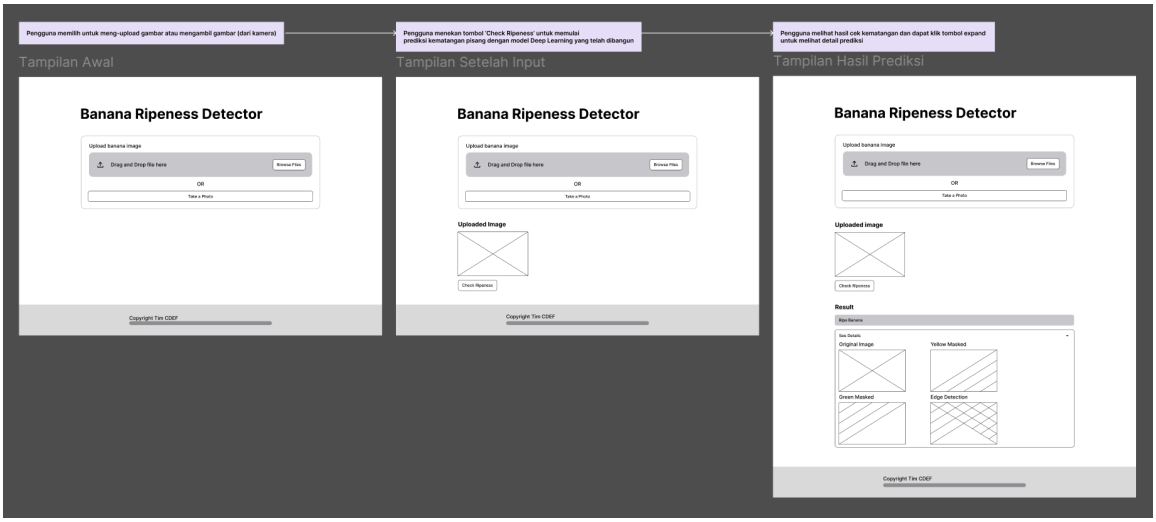
Selain dievaluasi dengan *confusion matrix*, model juga dievaluasi menggunakan beberapa metrik kinerja utama:

- *Precision*: mengukur sejauh mana prediksi benar untuk setiap kelas. Pada kasus ini, nilai *precision* untuk kedua kelas adalah 1.00 (100%).

- *Recall*: mengukur kemampuan model dalam menangkap semua kasus dalam dataset. Nilai *recall* adalah 1.00 (100%).
- *F1-Score*: memberikan keseimbangan antara *precision* dan *recall*. Nilai *F1-score* yang diperoleh adalah 1.00 (100%).

4.2 Desain Sistem

Perancangan aplikasi dilakukan dengan *tool* Figma. Aplikasi yang akan dikembangkan merupakan aplikasi *web* yang memungkinkan pengguna untuk meng-*upload* foto dan melakukan deteksi apakah gambar tergolong ke dalam pisang matang, pisang mentah, maupun bukan pisang. Rancangan *mockup* aplikasi dapat dilihat pada Gambar 26.



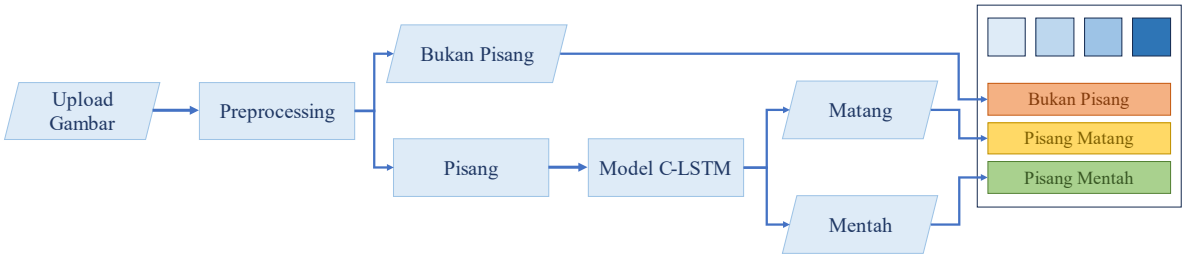
Gambar 26. Desain Sistem Aplikasi

4.3 Implementasi Sistem

Sistem ini dirancang untuk mengklasifikasikan gambar input dengan alur kerja sebagai berikut:

1. Pengguna mengunggah gambar ataupun memotret gambar secara langsung melalui *interface web*.
2. Gambar diproses melalui *pipeline preprocessing* untuk memastikan data berada dalam format yang sesuai dan mengekstrak fitur penting dari gambar.
3. Sistem memeriksa apakah gambar yang diunggah merupakan buah pisang atau bukan berdasarkan kepadatan tepi (*edge density*).
4. Model *deep learning* yang telah dilatih sebelumnya, melakukan prediksi untuk mengidentifikasi kelas gambar berupa tingkat kematangan buah.
5. Hasil prediksi ditampilkan pada *interface* pengguna, dan UI diperbarui secara *real-time* agar pengguna dapat langsung melihat tahapan proses klasifikasi.

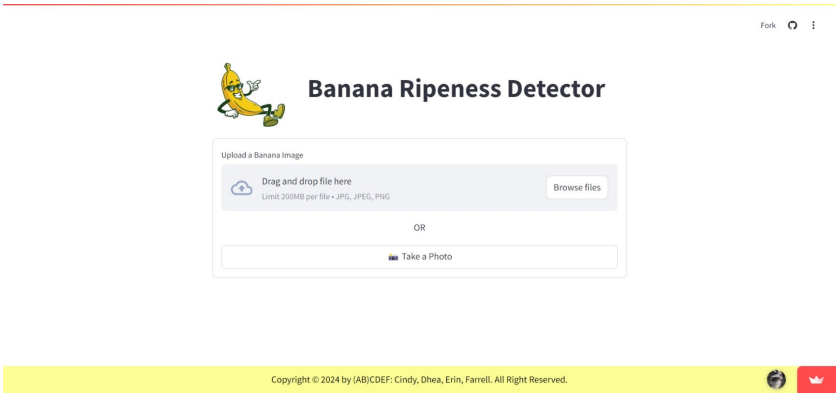
Diagram *workflow* sistem ditunjukkan pada gambar di bawah ini:



Gambar 27. *Workflow* Sistem

4.3.1 Hasil Implementasi dan Visualisasi

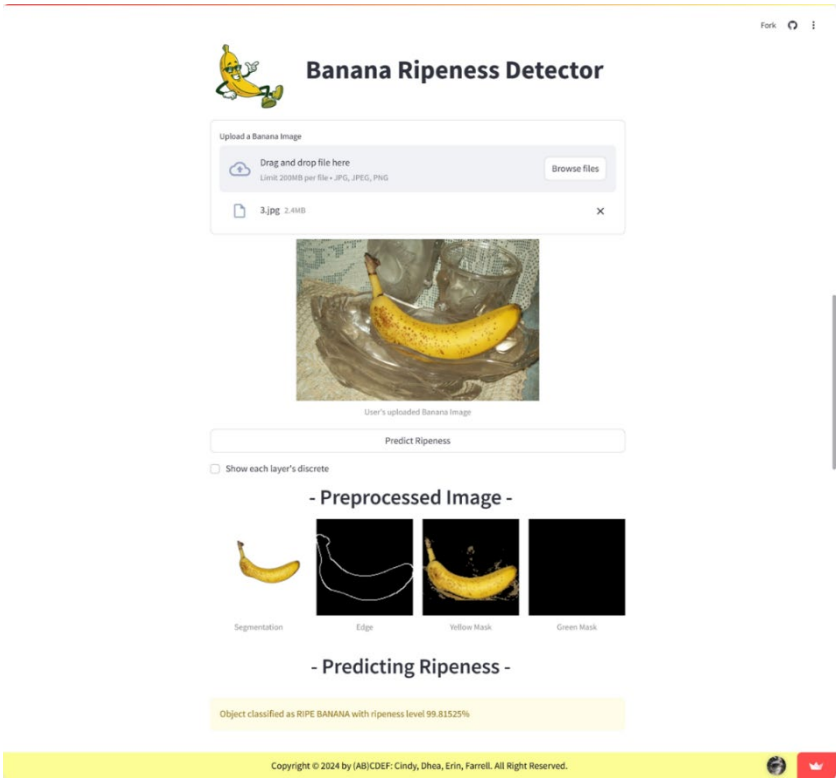
Sistem dapat langsung diakses melalui tautan berikut: banana-ripeness-detector-cdef.streamlit.app (diakses pada 22 Februari 2025). Gambar di bawah menunjukkan tampilan antar muka pengguna di mana pengguna dapat mengunggah gambar dan mendapatkan hasil prediksi apakah gambar tersebut adalah pisang atau bukan, kemudian dilanjutkan untuk proses prediksi tingkat kematangannya. Hasil prediksi diperbarui secara *real-time*, dan proses prediksi berjalan cepat dan efisien.



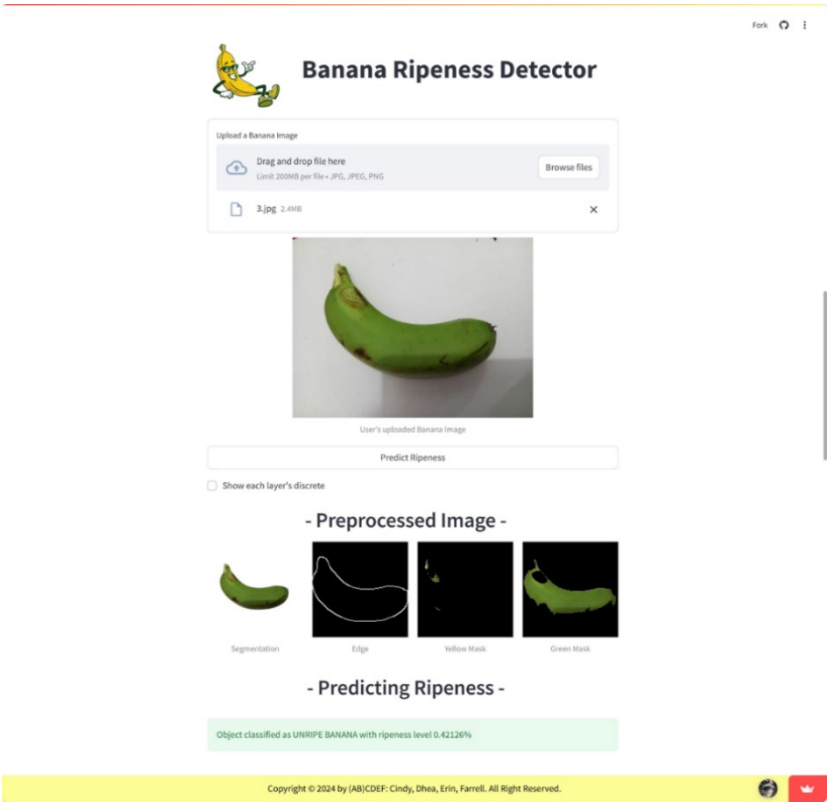
Gambar 28. Tampilan Halaman Utama Aplikasi

Gambar di bawah ini menunjukkan tampilan *interface* sistem saat pengguna mengunggah gambar pisang dan mendapatkan hasil prediksi.

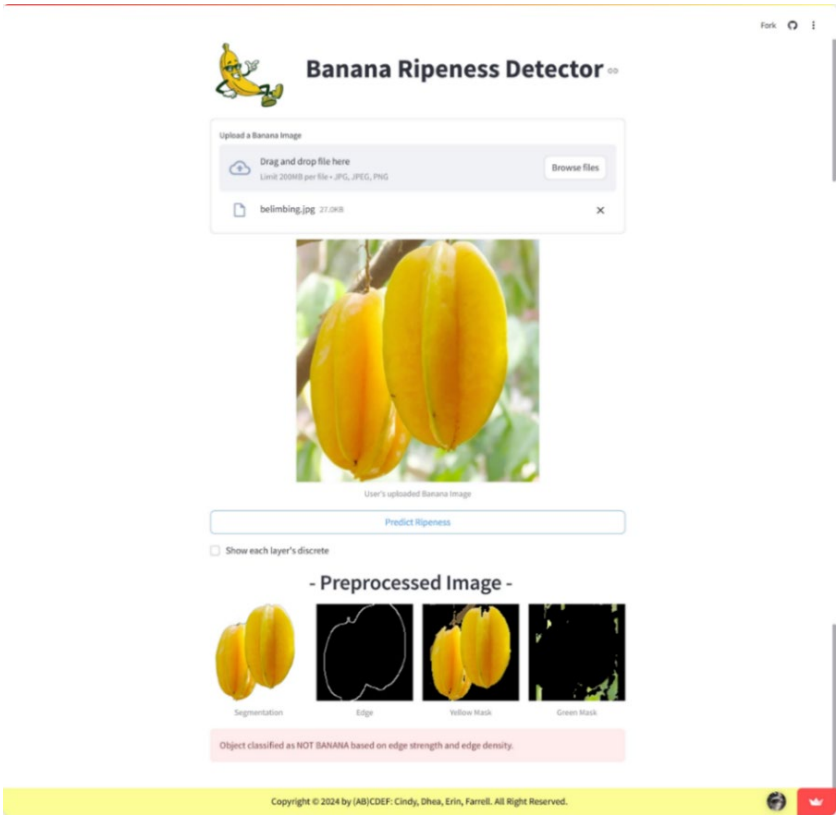
Jika gambar yang diunggah berisi obyek buah pisang, gambar akan diteruskan ke model untuk diprediksi tingkat kematangannya.



Gambar 29. Prediksi dengan Gambar Pisang Matang



Gambar 30. Prediksi dengan Gambar Pisang Mentah



Gambar 31. Prediksi dengan Gambar Bukan Pisang – Belimbing

Jika pengguna mengunggah gambar "bukan pisang" dan menekan tombol "*Predict Ripeness*", gambar akan melalui tahapan *preprocessing* untuk mengekstrak tepi dan menghitung kekuatan (*strength*) serta kepadatan (*density*) tepi. Jika kekuatan dan kepadatan tepi pisang melewati ambang batas yang ditentukan, gambar akan langsung dikembalikan bersama dengan hasil yang menyatakan bahwa obyek dalam gambar tersebut bukanlah pisang.

5. KESIMPULAN

Berdasarkan hasil penelitian dan pengembangan sistem klasifikasi kematangan buah pisang, berikut kesimpulan yang dapat diambil:

1. Model yang dikembangkan

Model *deep learning* berbasis arsitektur CNN dan LSTM yang dikembangkan mampu mengklasifikasikan tingkat kematangan pisang dengan akurasi 100% pada dataset *testing*. Hal ini menunjukkan bahwa pendekatan tersebut sangat efektif dalam menangkap pola fitur visual untuk membedakan kategori kematangan. Analisis kinerja model juga menunjukkan metrik evaluasi yang sempurna, pada *precision*, *recall*, dan *F1-score* sebesar 1.00 (100%) untuk semua kategori tingkat kematangan pisang. Hal ini menandakan model memiliki kemampuan yang sangat baik dalam mengenali pola visual dan menghasilkan prediksi yang konsisten.

2. *User Interface*

Antarmuka pengguna berbasis *Streamlit* memungkinkan sistem diakses dengan mudah oleh pengguna tanpa memerlukan instalasi tambahan. Sistem berhasil menampilkan prediksi tingkat kematangan pisang secara *real-time* dengan visualisasi yang jelas dan informatif.

Dengan hasil ini, sistem klasifikasi tingkat kematangan pisang berbasis *deep learning* dapat diandalkan untuk digunakan dalam aplikasi praktis, seperti *monitoring* kualitas buah agar dapat dipanen pada waktu yang tepat.

6. SARAN

Berdasarkan penelitian ini, saran yang dapat diberikan untuk pengembangan lebih lanjut adalah sebagai berikut:

1. Penambahan dataset

Menambah lebih banyak data *training* dengan variasi kondisi, seperti pencahayaan yang berbeda, *background* yang lebih kompleks, pemotretan dari berbagai sudut pandang buah pisang, atau hasil augmentasi data (rotasi, *flipping*, dll). Selain itu, juga dapat ditambahkan data berlabel bukan pisang dalam pelatihan model agar sistem tidak hanya mengandalkan pendeteksian tepi untuk mendeteksi citra bukan pisang. Hal dilakukan untuk meningkatkan kemampuan generalisasi model.

2. Eksplorasi model yang lebih kompleks

Penelitian lebih lanjut dapat dilakukan dengan mengeksplorasi arsitektur model yang lebih canggih, seperti menggunakan *pretrained models* atau penggabungan teknik *ensemble learning* untuk meningkatkan akurasi prediksi.

Dengan mengimplementasikan saran-saran ini, diharapkan sistem klasifikasi tingkat kematangan buah pisang dapat semakin optimal dan memberikan manfaat yang lebih besar dalam aplikasi praktis.

DAFTAR PUSTAKA

[1] J. Naranjo-Torres, M. Mora, R. Hernández-García, R. J. Barrientos, C. Fredes, and A. Valenzuela, "A Review of Convolutional Neural Network Applied to Fruit Image Processing," *Applied Sciences (Switzerland)*, vol. 10, no. 10, May 2020, doi: 10.3390/app10103443.

[2] W. Castro, J. Oblitas, M. De-La-Torre, C. Cotrina, K. Bazan, and H. Avila-George, "Classification of Cape Gooseberry Fruit According to its Level of Ripeness Using Machine

Learning Techniques and Different Color Spaces," *IEEE Access*, vol. 7, pp. 27389–27400, 2019, doi: 10.1109/ACCESS.2019.2898223.

[3] R. E. Saragih and A. W. R. Emanuel, "Banana Ripeness Classification Based on Deep Learning using Convolutional Neural Network," in *The 3rd 2021 East Indonesia Conference on Computer and Information Technology (EIConCIT)*, Surabaya: Institute of Electrical and Electronics Engineers (IEEE), Apr. 2021, pp. 85–89. doi: 10.1109/eiconcit50028.2021.9431929.

[4] R. Kosasih, "Klasifikasi Tingkat Kematangan Pisang Berdasarkan Ekstraksi Fitur Tekstur dan Algoritme KNN," *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, vol. 10, no. 4, Sep. 2021.

[5] Rini Andriani, Rizki Risdah Sitorus, Samuel Anaya Putra Zai, and Yesika Syalomi Pasaribu, "Penggunaan Algoritma CNN untuk Mengidentifikasi Jenis Anjing Menggunakan Metode Supervised Learning," *Mutiara : Jurnal Penelitian dan Karya Ilmiah*, vol. 1, no. 6, pp. 393–403, Dec. 2023, doi: 10.59059/mutiara.v1i6.741.

[6] N. M. Elshennawy and D. M. Ibrahim, "Deep-Pneumonia Framework Using Deep Learning Models Based on Chest X-Ray Images," *Diagnostics*, vol. 10, no. 9, p. 649, Aug. 2020, doi: 10.3390/diagnostics10090649.

[7] B. S. C. Putra and I. Tahyudin, "Performance Evaluation of CNN-LSTM and CNN-FNN Combinations for Pneumonia Classification Using Chest X-ray Images," *JITE (Journal of Informatics and Telecommunication Engineering)*, p. 12, Jan. 2025, doi: 10.31289/jite.v8i2.13503.

[8] N. Ketkar and J. Moolayil, *Deep learning with python: Learn Best Practices of Deep Learning Models with PyTorch*. Apress Media LLC, 2021. doi: 10.1007/978-1-4842-5364-9.

[9] A. Mathew, P. Amudha, and S. Sivakumari, "Deep Learning Techniques: An Overview," in *Advanced Machine Learning Technologies and Applications*, 2020, pp. 599–608.

[10] S. Ullah and S. H. Song, "Design of compensation algorithms for zero padding and its application to a patch based deep neural network," *PeerJ Comput Sci*, vol. 10, pp. 1–17, 2024, doi: 10.7717/PEERJ-CS.2287.

[11] R. C. . Gonzalez and R. E. . Woods, *Digital Image Processing, Fourth Edition*. Harlow: Pearson Education, 2018.

[12] J. Jing, S. Liu, G. Wang, W. Zhang, and C. Sun, "Recent advances on image edge detection: A comprehensive review," *Neurocomputing*, vol. 503, pp. 259–271, Sep. 2022, doi: 10.1016/j.neucom.2022.06.083.

[13] OpenCV, "Contours : Getting Started," https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html. Accessed: Nov. 11, 2024. [Online]. Available: https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html

[14] A. Hanafiah, Y. Arta, H. O. Nasution, and Y. D. Lestari, "Penerapan Metode Recurrent Neural Network dengan Pendekatan Long Short-Term Memory (LSTM) Untuk Prediksi Harga Saham," *Bulletin of Computer Science Research*, vol. 4, no. 1, pp. 27–33, Dec. 2023, doi: 10.47065/bulletincsr.v4i1.321.

[15] D. Setiawan, K. Stefani, Y. J. Shandy, and C. A. F. Patra, "Sistem Analisa Harga Saham Menggunakan Algoritma Long Short Term Memory (LSTM)," *Media Informatika*, vol. 21, no. 3, pp. 264–279, Mar. 2023, doi: 10.37595/mediainfo.v21i3.159.

[16] A. Celeghin *et al.*, "Convolutional Neural Networks for Vision Neuroscience: Significance, Developments, and Outstanding Issues," 2023, *Frontiers Media SA*. doi: 10.3389/fncom.2023.1153572.

[17] M. S. Anggreany, "Confusion Matrix." Accessed: Nov. 11, 2024. [Online]. Available: https://socs.binus.ac.id/2020/11/01/confusion-matrix/

[18] J. Xu, Y. Zhang, and D. Miao, "Three-way confusion matrix for classification: A measure driven view," *Inf Sci (N Y)*, vol. 507, pp. 772–794, Jan. 2020, doi: 10.1016/J.INS.2019.06.064.