JURNAL ILMU KOMPUTER
KHARISMA .TECH

Jurnal KHARISMA Tech | ISSN:1907-2317 | e-ISSN: 2810-0344
https://jurnal.kharisma.ac.id/kharismatech
published by: Pusat Penelitian STMIK KHARISMA Makassar

Volume: 21 no. 02 – September 2026- hlm. 54-68

PENGEMBANGAN APLIKASI PENDETEKSI AKSARA LONTARA BERBASIS *CONVOLUTIONAL NEURAL NETWORKS* MENGGUNAKAN *FRAMEWORK FLUTTER*

Oleh:

Linda Yanti Yo[1], Mohammad Fajar[2*], Baizul Zaman[3]
[1,2,3]Teknik Informatika, STMIK Kharisma Makassar
e-mail: [1]lindayanti_23@kharisma.ac.id, [2]fajar@kharisma.ac.id, [3]baizul@kharisma.ac.id

Abstrak: *Aksara Lontara merupakan warisan budaya Bugis-Makassar yang kian terancam punah akibat terbatasnya media pembelajaran interaktif. Penelitian ini bertujuan mengembangkan aplikasi mobile pendeteksi aksara Lontara berbasis Flutter yang terintegrasi dengan model Convolutional Neural Network (CNN). Berbeda dengan penelitian sebelumnya, aplikasi ini mengintegrasikan model CNN dengan framework Flutter melalui REST API, sehingga dapat berjalan secara real-time di perangkat mobile. Dataset terdiri dari 5.017 citra aksara yang terbagi dalam 23 kelas, diproses melalui tahap grayscale dan normalisasi. Model CNN dirancang dengan tiga lapisan konvolusi dan pooling, lalu dievaluasi menggunakan metrik akurasi, presisi, recall, dan F1-score. Hasil pengujian menunjukkan aplikasi mampu mendeteksi aksara Lontara secara real-time, dengan tingkat akurasi model mencapai 97%. Pengujian pada kondisi gambar dengan gangguan visual noise dan blur memperlihatkan model tetap mampu melakukan deteksi hingga level tertentu, namun performa menurun signifikan pada gangguan simultan tinggi. Model juga terbukti robust terhadap variasi warna dan ketebalan stroke. Namun, model belum mampu mendeteksi aksara dalam bentuk multi-karakter. Temuan ini menegaskan bahwa aplikasi dapat mendukung upaya pelestarian aksara Lontara untuk karakter tunggal meski tetap membutuhkan kualitas citra yang memadai agar deteksi berjalan optimal.*

Kata kunci: *Aksara lontara, Convolutional Neural Networks, Flutter, pengembangan aplikasi, pengenalan pola*

Abstract: *The Lontara script is a Bugis-Makassar cultural heritage threatened by extinction due to a lack of interactive learning media. This study developed a Flutter-based mobile application for detecting the Lontara script. Unlike previous research, this application integrates a CNN model with the Flutter framework via a REST API, enabling real-time operation on mobile devices. A dataset of 5,017 images across 23 character classes was processed through grayscaling and normalization. The designed CNN model, with three convolution and pooling layers, was evaluated using accuracy, precision, recall, and F1-score metrics. Testing showed the application successfully detects Lontara script in real-time, with a model accuracy of 97%. Evaluations under visual disturbances (noise and blur) demonstrated the model's robustness up to a certain degradation level, though performance declined significantly under high simultaneous noise. The model also proved resilient to variations in ink color and stroke thickness due to grayscale preprocessing. However, it could not detect multi-character script sequences. These findings confirm that the application can support efforts to preserve Lontara script for single characters, although adequate image quality is still required for optimal detection.*

Keywords: *Lontara script, Convolutional Neural Networks, Flutter, application development, pattern recognition*

[*] Corresponding author : Mohammad Fajar (fajar@kharisma.ac.id)

Diterima : *Agustus, 2025* ***Disetujui :*** *Oktober, 2025* ***Dipublikasikan :*** *September, 2026*

1. PENDAHULUAN

Aksara Lontara merupakan warisan budaya masyarakat Bugis-Makassar dan Mandar sejak abad ke-14 hingga awal abad ke-20 [1]. Namun, di era globalisasi keberadaan aksara lontara terancam punah akibat berkurangnya penggunaannya dalam kehidupan sehari-hari. Permasalahan ini terlihat jelas dalam penelitian A. Muh. Ayyub HT dan Indriani (2024) [2] yang menunjukkan bahwa 39 dari 40 siswa Bugis perantauan yang berusia 12-15 tahun di Kabupaten Pasangkayu Sulawesi Selatan belum pernah mempelajari aksara lontara sama sekali. Penelitian Kilawati dkk (2023) [3] menunjukkan permasalahan yang sama pada siswa kelas V di Kabupaten Barru Barru gagal menguasai dasar-dasar aksara lontara, mengindikasikan adanya permasalahan sistemik dalam kurikulum muatan lokal. Penurunan minat ini sebagian besar disebabkan oleh minimnya media pembelajaran interaktif dan metode pengajaran yang masih konvensional membuatnya kurang efektif di kalangan siswa sekolah dasar [4]. Selain itu, jarangnya penggunaan aksara lontara dalam kegiatan tulis-menulis dan pengaruh dominan penggunaan huruf Latin dalam kehidupan sehari-hari secara cepat menggantikan aksara lontara [1]. Berbagai upaya pelestarian telah dilakukan untuk mengatasi masalah ini, mulai dari inisiatif kebijakan seperti lomba digitalisasi berkonten aksara lontara oleh PANDI [5] hingga inovasi di bidang pendidikan. Seperti pengembangan aplikasi *mobile* dengan fitur kuis dan permainan interaktif [4], pengembangan aplikasi berbasis *Augmented Reality* (AR) [6], dan pengembangan bahan ajar seperti buku 'Cinnong Ati: Buku Baca Tulis Aksara Lontara' untuk Siswa SD Fase C' [3]. Namun, sebagian besar upaya tersebut hanya berfokus pada pengajaran aksara bagi siswa sekolah dasar.

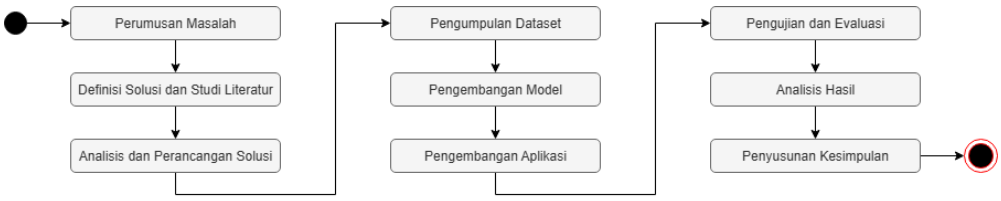
Di sisi lain, kemajuan teknologi *machine learning*, khususnya pengenalan pola, menawarkan peluang untuk memperluas jangkauan pelestarian aksara Lontara. Sejumlah penelitian terdahulu telah menunjukkan keberhasilan dalam pengenalan aksara ini. Mustafa dkk. (2024) [7] berhasil mengimplementasikan arsitektur CNN dengan akurasi 91,6%. Arthuriana dkk. (2024) [8] juga mencapai akurasi 97,65% menggunakan CNN. Sementara itu, Arhinza dkk. (2024) [9] menggunakan algoritma K-NN dengan ekstraksi fitur HOG dan memperoleh akurasi 92,35%. Meskipun akurasi model dalam penelitian-penelitian tersebut tinggi, solusinya masih berfokus pada pengembangan model dan belum terintegrasi penuh ke dalam aplikasi *mobile* yang modular. Oleh karena itu, penelitian ini mengusulkan integrasi model deteksi ke dalam aplikasi *mobile* menggunakan *framework Flutter*.

Berdasarkan tinjauan penelitian terdahulu, terdapat kesenjangan utama yang melatarbelakangi penelitian ini, yaitu minimnya studi yang mengintegrasikan model CNN untuk pengenalan aksara dengan *framework Flutter* melalui *REST API* pada perangkat *mobile*. Oleh karena itu, penelitian ini bertujuan merancang aplikasi pendeteksi aksara Lontara berbasis CNN dengan *Flutter*. Hasil penelitian diharapkan dapat memberikan kontribusi penting dalam pengembangan aplikasi *mobile* untuk pelestarian dan pembelajaran aksara Lontara.

2. METODE PENELITIAN

Penelitian diawali dengan perumusan masalah terkait aspek pengembangan aplikasi pendeteksi aksara Lontara berbasis metode *Convolutional Neural Network* dengan *framework*

Flutter. Tahap ini kemudian dilanjutnya dengan studi literatur untuk mengumpulkan data dan informasi dari penelitian-penelitian terdahulu terkait teknologi pengenalan pola aksara menggunakan CNN dan pengembangan aplikasi mobile dengan *Flutter*, yang kemudian mendefinisikan arsitektur solusi yang akan dibangun. Selanjutnya, pada tahap pengumpulan dataset, dikumpulkan citra aksara Lontara yang kemudian digunakan dalam tahap pengembangan model CNN sebagai bagian utama dari sistem deteksi. Kemudian pada tahap pengembangan aplikasi, model yang dihasilkan kemudian diintegrasikan ke dalam aplikasi berbasis *Flutter* sebagai media pembelajaran interaktif. Setelah itu dilakukan tahap pengujian dan evaluasi terhadap kinerja model dan aplikasi. Hasil pengujian dianalisis pada tahap analisis hasil, kemudian dirangkum dalam tahap penyusunan kesimpulan yang memuat temuan penelitian serta rekomendasi untuk penelitian selanjutnya. Gambar 1 menyajikan tahapan-tahapan dalam penelitian ini.



Gambar 1 Tahapan Penelitian

2.1. *Framework Flutter*

Flutter adalah *framework open source* yang dirancang oleh *Google* untuk pengembangan aplikasi *mobile multiplatform* berperforma tinggi dari satu *codebase* [10]. Pengembangan aplikasi menggunakan *Flutter* dilakukan dengan bahasa *Dart*.

Keuntungan menggunakan *Flutter* adalah kemampuan untuk beroperasi pada berbagai sistem operasi seperti iOS dan *Android* dalam satu *codebase*, siklus pengembangan yang cepat dengan fitur *hot reload* dan *hot restart*, kurva pembelajaran yang relatif mudah, dan kemudahan membuat UI yang lebih intuitif dan ekspresif [10].

2.2. Pengenalan Pola

Pengenalan pola adalah proses mengindentifikasi struktur data dengan membandingkannya terhadap pola referensi yang telah dikenal (*known pattern*), yang umumnya mengadopsi mengadopsi metode *machine learning* seperti klasifikasi atau *clustering* untuk mengelompokkan data berdasarkan kesamaan [7].

2.3. *Convolutional Neural Network* (CNN)

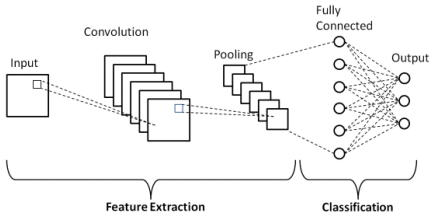
2.3.1. Konsep Dasar CNN

Convolutional Neural Network (CNN) merupakan salah satu bentuk jaringan saraf tiruan yang dibangun untuk memproses dan menganalisis data citra [11]. Secara struktural, CNN terdiri dari tiga lapisan utama:

a. Lapisan Konvolusi berfungsi mengaplikasikan filter (*kernel*) yang bergerak melintasi citra *input* untuk mendeteksi pola lokal (seperti tepi, tekstur, atau bentuk dasar). Setiap filter menghasilkan peta aktivasi yang menonjolkan keberadaan fitur tertentu [12].
b. Lapisan *Pooling*, berperan dalam mereduksi dimensi spasial peta fitur melalui operasi downsampling (*max/average pooling*) untuk meningkatkan efisiensi komputasi dan

memberikan ketahanan terhadap variasi posisi objek [12].

c. Lapisan *Fully-Connected*, berperan dalam mengintegrasikan fitur yang diekstrak untuk klasifikasi akhir menggunakan perceptron tradisional, seringkali diikuti fungsi aktivasi softmax untuk prediksi kelas [12].



Gambar 2 Alur pemrosesan CNN [13]

2.3.2. Implementasi Arsitektur CNN

Arsitektur model CNN yang dikembangkan pada penelitian ini dirancang dengan tiga blok konvolusi-*pooling*, di mana setiap blok konvolusi terdiri dari:

a. Lapisan Konvolusi 2D dengan 32, 64, dan 128 filter secara berurutan untuk setiap blok, menggunakan ukuran *kernel* 3x3 dan fungsi aktivasi *Rectified Linear Unit* (ReLU).
b. Lapisan *Max-Pooling* 2D dengan ukuran *pool* 2x2 yang bertujuan untuk mengurangi dimensi spasial dan menekankan fitur yang paling dominan.

Hasil dari blok konvolusi ketiga kemudian diratakan (*flattened*) menjadi sebuah vektor satu dimensi. Vektor ini kemudian dihubungkan ke lapisan *fully-connected* (*dense*) yang terdiri dari 512 *neuron* dengan aktivasi ReLU. Pada lapisan akhir yang merupakan lapisan *fully-connected* dengan 23 *neuron* menggunakan fungsi aktivasi *softmax* untuk melakukan klasifikasi multi-kelas ke dalam 23 aksara Lontara. Model dicompile menggunakan *optimizer Adam* dengan *learning rate default* (0,001), fungsi *loss Categorical Crossentropy*, dan metrik evaluasi akurasi. Pelatihan model dilakukan sebanyak 250 *epoch* dengan *batch size* 32. Dataset utama dengan 5,017 citra dibagi dengan rasio 70:30 untuk data latih dan validasi.

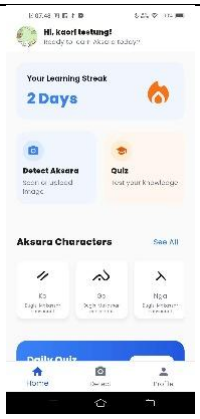
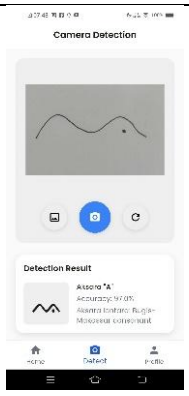
3. HASIL DAN PEMBAHASAN

3.1. Hasil Implementasi Sistem

Hasil implementasi sistem merupakan tampilan aplikasi yang diimplementasikan dari antarmuka pengguna (UI) yang telah didesain terdiri dari tampilan *splash screen, onboarding,* autentikasi, *home,* deteksi aksara, list aksara Lontara, *quiz*, hasil *quiz*, profil *user*, dan *edit* profil *user*. Integrasi antarmuka ini dengan model CNN melalui REST API menghadapi beberapa tantangan teknis yang berhasil diidentifikasi dan diatasi. Pertama, tantangan *latency* dan beban komputasi diatasi dengan mengoptimasi *pipeline* inferensi pada sisi server. Gambar yang dikirim dari aplikasi Flutter diproses secara efisien di server dengan melakukan *resizing* ke resolusi 64x64 piksel dan konversi *grayscale* sebelum dilakukan inferensi oleh model. Kedua, untuk *deployment* yang stabil dan efisien, model dikonversi ke format *TensorFlow Lite* (TFLite) yang lebih ringan dan cepat. Ketiga, integrasi antara aplikasi Flutter dan server menerapkan *error handling* dan *timeout management* pada setiap request API untuk meminimalisir dampak dari kegagalan jaringan atau server. Solusi-solusi ini memastikan bahwa aplikasi tidak hanya

berfungsi secara visual tetapi juga robust dan responsif dalam penggunaan nyata. Tabel 1 menyajikan tampilan antarmuka utama yang diimplementasikan dalam aplikasi berbasis *Flutter*.

Tabel 1 Tampilan Antarmuka Aplikasi

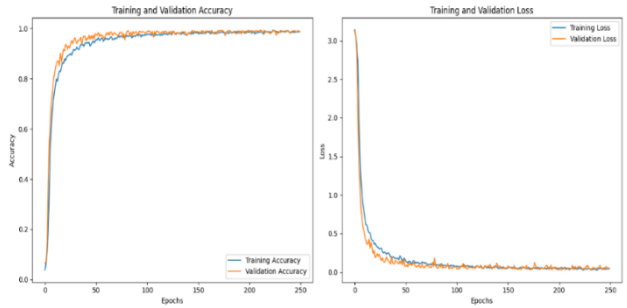
Tampilan	Hasil
Home	
Deteksi aksara	
List aksara lontara	
Profil User	

3.2. Hasil Pengujian Kinerja Model

Pengujian kinerja model dilakukan untuk mengetahui sejauh mana model yang telah dilatih dapat mengenali aksara Lontara dengan baik. Pengujian ini dibagi menjadi tiga bagian, yaitu hasil pelatihan model, hasil pengujian berdasarkan *dataset* uji, dan hasil pengujian aksara.

3.2.1. Hasil Pelatihan Model

Pada tahap pelatihan, model mulai mempelajari pola aksara Lontara yang tersedia di *dataset* latih. Selama proses ini, kinerja model dipantau melalui dua indikator utama, yaitu akurasi dan *loss*, baik pada data latih maupun data validasi. Perkembangan kinerja tersebut dapat dilihat pada Gambar 3.



Gambar 3 Kurva Performa Model

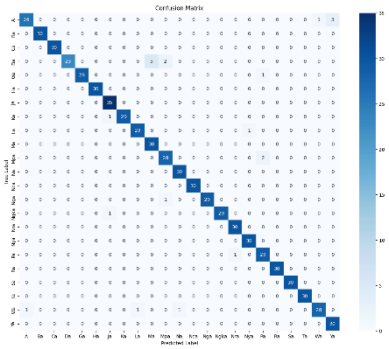
Berdasarkan Gambar 3, grafik akurasi di sisi kiri menunjukkan bahwa akurasi model pada data latih (kurva biru) dan data validasi (kurva jingga) sama-sama meningkat seiring bertambahnya epoch. Kurva validasi yang mengikuti pola kurva latih menandakan bahwa model tidak hanya menghafal data latih, tetapi juga mampu mengenali data baru dengan baik. Sementara, pada grafik *loss* di sisi kanan menggambarkan tingkat kesalahan model selama proses pelatihan, di mana seiring meningkatnya akurasi, nilai *loss* pada data latih dan validasi terus menurun, yang menunjukkan bahwa model belajar secara bertahap dan semakin baik pada tiap *epoch*. Konvergensi kedua kurva loss pada nilai yang rendah menjadi tanda bahwa model telah mencapai performa yang stabil dan optimal pada akhir. Pada epoch terakhir, model mencapai akurasi training sebesar 98.75% dan akurasi validasi sebesar 98.88%, dengan loss training sebesar 3.74% dan loss validasi sebesar 4.92%.

3.2.2. Hasil Pengujian *Dataset* Uji

Setelah proses pelatihan selesai, model dievaluasi menggunakan *dataset* uji untuk melihat seberapa baik kinerja model terhadap data yang belum pernah dilihat sebelumnya. Proses evaluasi ini menghasilkan *confusion matrix* serta laporan klasifikasi yang mencakup metrik presisi, *recall*, *f1-score*, dan akurasi keseluruhan.

3.2.2.1. Confusion Matrix

Hasil *confusion matrix* yang diperoleh dari pengujian di *Google Colab* dapat dilihat pada Gambar 4.



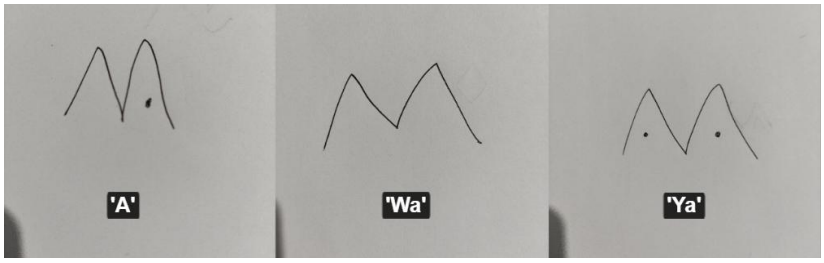
Gambar 4 Confusion Matrix

Pada *confusion matrix*, nilai yang berada di garis diagonal utama menunjukkan jumlah data yang berhasil diprediksi dengan benar oleh model (*True Positive*), sedangkan nilai yang berada di luar diagonal utama menunjukkan kesalahan prediksi, di mana data dari suatu kelas salah diklasifikasikan sebagai kelas lainnya (*False Positive* atau *False Negative*).

Berdasarkan *confusion matrix* yang disajikan pada gambar 4, terlihat bahwa sebagian besar nilai pada diagonal utama menunjukkan jumlah yang mendekati total data uji untuk masing-masing kelasnya yang berkisar 30 data per kelas. Hal ini menunjukkan kemampuan model yang baik dalam mengenali sebagian besar aksara lontara. Meski demikian, masih terdapat beberapa kesalahan klasifikasi kecil, seperti:

1. Pada kelas A, model mengklasifikasikan 26 data dengan benar, tetapi terdapat 1 data yang salah diprediksi sebagai ‘Wa’ dan 3 data sebagai ‘Ya’.
2. Pada kelas Da, terdapat 23 prediksi benar, tetapi 5 data salah diklasifikasikan sebagai ‘Ja’ dan 2 data sebagai ‘Ka’.
3. Pada kelas Wa, dari 31 data, 28 data diklasifikasikan dengan benar, dan 3 data salah diklasifikasikan ke kelas ‘A’, ‘La’, dan ‘Na’.
4. Kesalahan kecil juga terlihat pada beberapa kelas lain seperti 'Ga', 'Ja', 'Ka', 'Ma', 'Na', dan 'Pa' yang mengalami satu kali kesalahan prediksi.

Kesalahan klasifikasi yang terjadi diduga kuat terjadi karena kemiripan visual pada bentuk lengkung dasar dan proporsi aksara. Sebagai contoh, aksara 'A', 'Wa', dan ‘Ya’ memiliki struktur vertikal yang mirip, hanya berbeda pada keberadaan dan posisi titik (nikka), yang mungkin hilang atau tidak jelas pada citra yang terdegradasi, di mana penulisan antara ketiga aksara tersebut dapat dilihat pada Gambar 5. Namun, secara keseluruhan, hasil *confusion matrix* ini memperlihatkan kemampuan model cukup baik dalam mengenali aksara Lontara, dengan tingkat kesalahan yang relatif rendah dan akurasi yang mendekati sempurna di sebagian besar kelas.



Gambar 5 Tulisan Aksara A, Wa, Ya

3.2.2.2. Laporan Klasifikasi

Selain confusion matrix, kinerja model juga dapat dilihat melalui laporan klasifikasi yang dapat diperoleh secara otomatis melalui Google *Colab* maupun secara manual dengan perhitungan. Gambar 6 mensajikan laporan klasifikasi yang dihasilkan dari Google *Colab*.

No.	Aksara	precision	recall	f1-score	support
1.	A	0.96	0.87	0.91	30
2.	Ba	1.00	1.00	1.00	30
3.	Ca	1.00	1.00	1.00	30
4.	Da	1.00	0.77	0.87	30
5.	Ga	1.00	0.97	0.98	30
6.	Ha	1.00	1.00	1.00	30
7.	Ja	0.95	1.00	0.97	30
8.	Ka	1.00	0.97	0.98	30
9.	La	0.97	0.97	0.97	30
10.	Ma	0.86	1.00	0.98	30
11.	Mpa	0.90	0.93	0.92	30
12.	Na	0.97	1.00	0.98	30
13.	Nca	1.00	1.00	1.00	30
14.	Nga	1.00	0.97	0.98	30
15.	Ngka	1.00	0.97	0.98	30
16.	Nra	0.97	1.00	0.98	30
17.	Nya	0.97	1.00	0.98	30
18.	Pa	0.91	0.97	0.94	30
19.	Ra	1.00	1.00	1.00	30
20.	Sa	1.00	1.00	1.00	30
21.	Ta	1.00	1.00	1.00	30
22.	Wa	0.97	0.90	0.93	31
23.	Ya	0.91	1.00	0.95	30
	Accuracy			0.97	696
	macro avg	0.97	0.97	0.97	696
	weighted avg	0.97	0.97	0.97	696

Gambar 6 Hasil laporan klasifikasi

Berdasarkan Gambar 6, dapat dilihat bahwa performa model konsisten dalam mengenali masing-masing aksara dengan tingkat kesalahan yang rendah, di mana rata-rata nilai presisi, *recall*, dan *f1-score* pada semua kelasnya mayoritas berada di atas 0.95. Untuk memastikan kesesuaian hasil laporan klasifikasi yang diperoleh dari *Google Colab*, peneliti melakukan verifikasi dengan perhitungan manual pada salah satu kelas aksara, yaitu kelas ‘Ya’. Berdasarkan data pada *confusion matrix* yang diperoleh, maka dapat diidentifikasi komponen-komponen berikut untuk kelas 'Ya'.

1. *True Positive* (TP) berisi jumlah prediksi yang benar, yaitu 30.
2. *False Positive* (FP) berisi jumlah prediksi keliru sebagai 'Ya' dari kelas lain, yaitu 3 (dari kelas 'A').
3. *False Negative* (FN) berisi jumlah 'Ya' yang keliru diprediksi sebagai kelas lain, yaitu 0.

Berdasarkan komponen tersebut, metrik untuk kelas 'Ya' dihitung sebagai berikut:

$$Akurasi = \frac{Jumlah\ prediksi\ benar}{Total\ data} = \frac{30}{30} = 1.00$$

$$Presisi = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)} = \frac{30}{30+3} = 0.909$$

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)} = \frac{30}{30+0} = 1.00$$

$$F1 - Score = 2\ x\ \frac{Presisi\ x\ Recall}{Presisi + Recall} = 2\ x\ \frac{0.909\ x\ 1.00}{0.909 + 1.00} = 0.9523$$

Berdasarkan hasil perhitungan yang telah dilakukan, menunjukkan bahwa hasil perhitungan manual untuk akurasi, presisi, *recall*, dan *f1-score* pada kelas 'Ya' menunjukkan hasil yang sama dengan tabel laporan klasifikasi, perbedaan kecil yang terlihat disebabkan oleh pembulatan angka atau metode perhitungan rata-rata (*macro* dan *weighted average*) yang digunakan oleh *Google Colab*.

3.3. Hasil Pengujian Aksara

Setelah dilakukan pengujian *dataset* uji dengan *confusion matrix* dan laporan klasifikasi, tahap pengujian berikutnya meliputi pengujian ketahanan model terhadap variasi kualitas gambar dengan gangguan *noise* dan *blur*, pengujian aksara terhadap variasi visual dengan pulpen berwarna dan stabilo dan pengujian *multi*-karakter aksara.

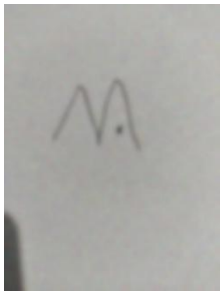
3.3.1. Pengujian Aksara dengan Gangguan Visual *Noise* dan *Blur*

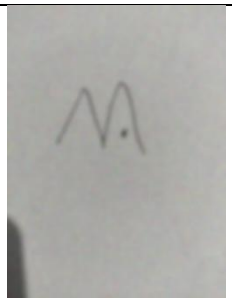
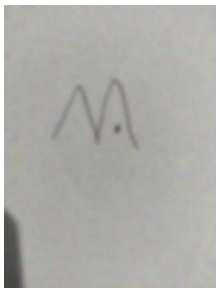
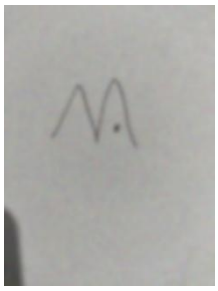
Pengujian ini mensimulasikan kondisi dunia nyata di mana kualitas *input* citra tidak selalu ideal untuk mengevaluasi ketahanan (*robustness*) model CNN dalam mengenali aksara Lontara yang terdegradasi. Degradasi kualitas citra diterapkan menggunakan *OpenCV* dan *NumPy* dengan prosedur sebagai berikut:

1. Penambahan *Gaussian Noise* pada citra asli menggunakan fungsi np.*random.normal*(*mean*,*sigma*,(*row, col, ch*)). Intensitas *noise* dikontrol oleh parameter simpangan baku (*standard deviation*) *sigma* (σ) dengan parameter 1 hingga 99.
2. Penambahan *Gaussian Blur* pada citra yang telah diberikan *noise* menggunakan fungsi *cv2.GaussianBlur*() dengan tingkat keburaman yang dikontrol melalui parameter ukuran *kernel* dan divariasikan dari 1 hingga 99.

Kombinasi dari variasi parameter *sigma* (*noise*) dan ukuran *kernel* (*blur*) menghasilkan berbagai tingkat degradasi. Hasil pengujian aksara dengan gangguan visual disajikan pada Tabel 2.

Tabel 2 Hasil Pengujian Aksara dengan Gangguan Visual

No.	Aksara	Kondisi	Hasil
1.	A	Kondisi gambar *noise* 63 dan *blur* 63 	Gagal mengenali aksara 
2.	A	Kondisi gambar *noise* 61 dan *blur* 61	Berhasil mengenali aksara

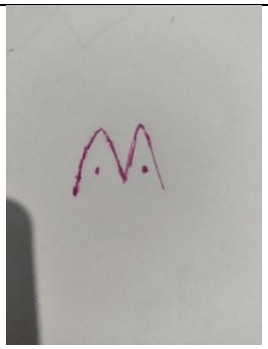
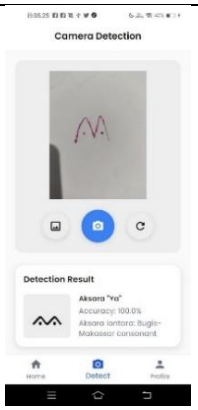
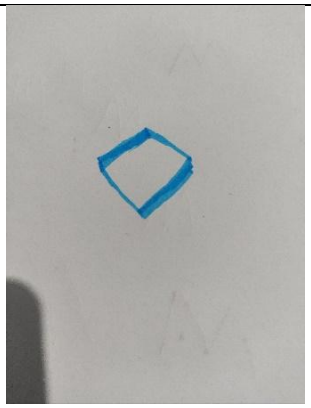
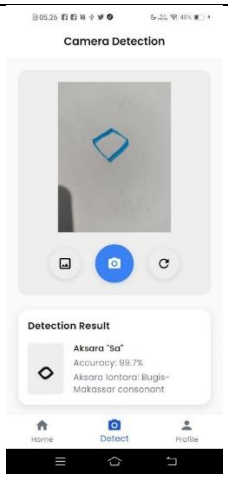
No.	Aksara	Kondisi	Hasil
			
3.	A	Kondisi gambar *noise* 63 dan *blur* 61 	Berhasil mengenali aksara 
4.	A	Kondisi gambar *noise* 67 dan *blur* 61 	Gagal mengenali aksara 

Berdasarkan hasil pengujian pada Tabel 2, model menunjukkan ketangguhan (robustness) yang baik dalam mengenali aksara Lontara 'A' dalam kondisi citra terdegradasi. Model mampu berhasil mengenali aksara pada tingkat *noise* dan *blur* menengah, seperti pada kondisi *noise* σ=61 dan *blur kernel*=61. Selain itu, model menunjukkan kemampuan untuk mentolerir satu jenis degradasi dengan intensitas lebih tinggi selama nilai parameter lainnya tidak terlalu tinggi, sebagaimana terbukti ketika *noise* dinaikkan menjadi σ=63 dengan *blur kernel*=61 model masih berhasil mengenali aksara dari *input* citra. Namun, model mengalami kegagalan ketika kedua parameter degradasi dinaikkan melebihi ambang toleransi, seperti pada *noise* σ=63 dengan *blur kernel*=63 atau *noise* σ=67 dengan *blur kernel*=61. Pada kondisi tersebut, fitur-fitur visual dari aksara seperti tepi, sudut, dan tekstur telah mengalami degradasi parah, sehingga fitur visualnya tidak dapat diekstraksi dan dikenali oleh model. Hasil ini membuktikan bahwa kualitas *input* citra yang tidak ideal, seperti citra yang memiliki *noise* atau keburaman yang berlebihan, dapat secara signifikan menurunkan kinerja model dalam mengenali aksara.

3.3.2. Pengujian Aksara Terhadap Variasi Visual: Pulpen Berwarna dan Stabilo

Pengujian aksara terhadap variasi visual dilakukan untuk mengevaluasi ketahanan model dalam mengenali aksara ketika diberikan citra dengan karakteristik visual yang berbeda. Pada pengujian ini, digunakan dua jenis alat tulis, yaitu pulpen berwarna *pink* untuk penulisan biasa, dan *stabilo* biru untuk penulisan dengan kombinasi goresan tebal dan tipis, menyerupai gaya tulisan menggunakan pena kaligrafi.

Tabel 3 Pengujian Aksara Terhadap Variasi Visual

No.	Aksara	Alat Tulis	Gambar	Hasil
1.	A	Pulpen *pink*		
2.	Sa	*Stabilo* biru		

Berdasarkan hasil pengujian yang disajikan pada Tabel 3, model terbukti mampu mengenali aksara dengan akurasi cukup tinggi meskipun dengan variasi visual pada *input* citra, dalam hal ini warna tinta (pulpen pink) dan variasi ketebalan stroke (stabilo biru). Hal ini disebabkan oleh tahap pra-pemrosesan yang mengonversi citra menjadi format *grayscale*, sehingga model tidak terpengaruh oleh perubahan warna, karena hanya informasi intensitas cahaya (*luminance*) yang diproses oleh model. Dengan demikian, model tetap dapat mengenali aksara yang ditulis menggunakan pulpen berwarna maupun *stabilo* dengan tingkat akurasi yang cukup memuaskan selama bentuk dasar (shape) dan tekstur aksara tetap terjaga.

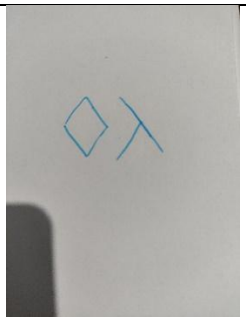
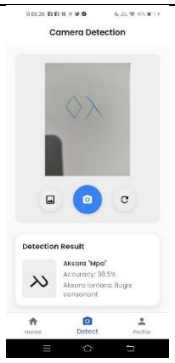
Namun, apabila variasi ketebalan stroke pada tulisan lebih tebal atau bentuknya belum tercakup dalam dataset pelatihan, model berpotensi mengalami penurunan performa dan tidak dapat mengenali aksara. Oleh karena itu, pengembangan selanjutnya perlu menambahkan

variasi *dataset* yang lebih beragam, terutama pada penulisan aksara dengan beragam ketebalan stroke. Hal ini bertujuan untuk meminimalisir kesalahan deteksi dan meningkatkan ketahanan model dalam menghadapi variasi visual yang lebih beragam.

3.3.3. Pengujian Aksara untuk Multi-Karakter

Pengujian Aksara untuk *multi*-karakter per gambar dapat terlihat pada Tabel 4.

Tabel 4 Pengujian Multi-Karakter Aksara

No.	Aksara	Gambar	Hasil
1.	A		

Berdasarkan hasil pengujian yang ditampilkan pada Tabel 4, diketahui bahwa model belum mampu mengenali huruf aksara dalam citra yang mengandung lebih dari satu karakter (*multi*-karakter) yang menyebabkan model memberikan hasil deteksi yang tidak sesuai. Hal ini disebabkan oleh keterbatasan arsitektur model *Convolutional Neural Network* (CNN) yang digunakan dirancang khusus untuk klasifikasi citra tunggal per kelas dan *dataset* pelatihan yang hanya berisi gambar satu karakter per sampel sehingga model tidak pernah belajar pola *multi*-karakter. Dengan demikian, model tidak mampu mengakomodasi *input* berupa citra yang mengandung rangkaian aksara atau kata. Untuk dapat mengenali kata atau rangkaian karakter secara utuh, diperlukan pendekatan yang dirancang khusus untuk menangani data sekuensial, di mana model tidak hanya mengekstraksi fitur visual dari satu aksara, tetapi juga memahami hubungan spasial antar karakter yang berdekatan. Metode seperti *Convolutional Recurrent Neural Network* (CRNN) atau kombinasi CNN dengan *Long Short-Term Memory* (LSTM) dan *Connectionist Temporal Classification* (CTC) *Loss* umum digunakan dalam pengenalan tulisan tangan *multi*-karakter. Metode tersebut melakukan ekstraksi fitur visual melalui CNN, kemudian memproses fitur tersebut sebagai urutan menggunakan LSTM, sehingga mampu mengenali keseluruhan kata dalam sebuah gambar.

Meskipun pendekatan tersebut belum diimplementasikan dalam penelitian ini karena keterbatasan waktu dan ruang lingkup, integrasi metode sekuensial seperti CRNN menjadi salah satu arah pengembangan yang berpotensial untuk penelitian lebih lanjut. Implementasi tersebut akan memperluas kemampuan aplikasi dalam mendeteksi tidak hanya karakter tunggal, tetapi juga kata atau kalimat utuh dari citra *input*.

3.4. Hasil Pengujian Fungsional Aplikasi

Pengujian fungsional aplikasi dilakukan menggunakan metode *black box testing* yang berfokus pada kesesuaian keluaran terhadap setiap masukan yang diberikan, tanpa

mempertimbangkan struktur internal kode. Pengujian ini dilakukan dengan mengecek serangkaian skenario yang mencakup semua fungsionalitas utama aplikasi dan dibandingkan dengan hasil yang diharapkan. Gambar 7 menyajikan scenario utama dari 21 skenario yang telah dilakukan.

No.	Menu yang diuji	Skenario Pengujian	Tipe Pengujian	Hasil yang diharapkan	Keterangan
1.	*Home*		Positif	Menampilkan halaman *home*.	Berhasil
2.	*List* Aksara		Positif	Menampilkan 23 huruf aksara lontara beserta detailnya	Berhasil
3.	Deteksi Aksara	Menangkap tulis tangan aksara lontara menggunakan kamera atau meng*upload* melalui galeri *user*. Kemudian mengklik tombol "Check".	Positif	Menampilkan hasil prediksi aksara lontara	Berhasil
4.	*Quiz*	Mengklik jawaban yang benar	Positif	Menampilkan perubahan warna tombol "Check answer", opsi jawaban yang dipilih menjadi hijau, dan berpindah halaman ke soal selanjutnya.	Berhasil
5.	Nilai *quiz*		Positif	Menampilkan nilai kuis dari hasil jawaban *user*.	Berhasil
6.	Profil *user*		Positif	Menampilkan data *user* dan tombol *edit user* serta *logout*.	Berhasil

Gambar 7 Hasil Pengujian *Black Box*

Berdasarkan hasil pengujian *black box* pada Gambar 7, sebanyak 21 skenario pengujian telah dijalankan secara manual untuk mencakup seluruh alur kerja utama aplikasi yang meliputi autentikasi pengguna, deteksi aksara, kuis, dan manajemen profil. Hasil pengujian menunjukkan bahwa 20 skenario (95.2%) berhasil dengan hasil yang sesuai dengan spesifikasi yang diharapkan (*expected output*), sementara 1 skenario (4.8%) mengalami kegagalan, yaitu pada fitur *streak*. Kegagalan pada fitur streak disebabkan oleh logika pemrograman yang belum menambahkan 1 hari secara otomatis ketika pengguna menggunakan aplikasi pada hari berturut-turut. Secara keseluruhan, hasil ini mengindikasikan bahwa aplikasi telah berfungsi sesuai dengan fungsionalitas fitur yang dirancang. Namun, untuk memastikan reliabilitas dan kestabilan aplikasi, pengujian lebih lanjut dengan skenario yang lebih komprehensif serta melibatkan pengguna nyata (*user acceptance testing*) sangat diperlukan.

4. KESIMPULAN

Berdasarkan hasil penelitian yang telah dilakukan, disimpulkan bahwa:

1. Aplikasi deteksi aksara lontara yang dikembangkan menggunakan *framework flutter* melalui *REST API* mampu menjalankan alur deteksi dari *input* gambar oleh pengguna hingga menampilkan *output* prediksi secara *real-time*. Hal ini dapat dilakukan terhadap semua aksara lontara Bugis.
2. Model *Convolutional Neural Network* (CNN) yang dirancang terbukti efektif dan mampu mencapai akurasi sebesar 97% dalam mengenali aksara Lontara. Pengujian ketahanan

(robustness) model terhadap degradasi gambar menunjukkan bahwa model mampu beradaptasi pada kondisi gangguan visual *noise* dan *blur* tingkat menengah (contoh: *noise* σ=61 dan *blur kernel*=61). Model terbukti toleran terhadap nilai salah satu parameter yang lebih tinggi (contoh: *noise* σ=63 dengan *blur kernel*=61). Namun, performa model menurun signifikan dan mengalami kegagalan ketika kedua parameter degradasi diberikan secara bersamaan pada tingkat tinggi (seperti pada kondisi *noise* σ=63 dengan *blur kernel*=63 atau *noise* σ=67 dengan *blur kernel*=61). Hal ini mengonfirmasi bahwa model memiliki batas toleransi terhadap kualitas citra, sehingga dalam penerapan nyata diperlukan input citra dengan kualitas baik untuk hasil deteksi yang optimal. Di sisi lain, pengujian terhadap variasi visual warna tinta (pulpen berwarna) dan ketebalan *stroke* (*stabilo*) membuktikan ketahanan model dalam mengenali aksara, karena tahap pra-pemrosesan grayscale membuat model hanya berfokus pada bentuk dan tekstur aksara, sehingga tidak terpengaruh oleh variasi warna.

3. Model yang dikembangkan masih terbatas pada deteksi 1 karakter aksara per citra dan belum mampu mengenali aksara dalam bentuk rangkaian atau kata. Hal ini disebabkan oleh arsitektur CNN yang digunakan hanya dirancang untuk klasifikasi satu kelas per gambar dan dataset latih yang tidak mencakup pola *multi*-karakter. Oleh karena itu, untuk pengembangan lebih lanjut diperlukan pendekatan lain, seperti *Convolutional Recurrent Neural Network* (CRNN) atau kombinasi CNN dengan LSTM, agar aplikasi dapat mengenali rangkaian aksara secara utuh dari *input* citra.

DAFTAR PUSTAKA

[1] Anton Setiawan, “Lontara, Aksara Mendunia dari Bugis,” indonesia.go.id. Accessed: Apr. 22, 2025. [Online]. Available: https://indonesia.go.id/kategori/budaya/2576/lontara-aksara-mendunia-dari-bugis

[2] A. M. Ayyub HT and Indriani, “Revitalisasi Aksara Lontara sebagai Identitas Budaya bagi Siswa Suku Bugis Perantauan,” Nov. 2024. doi: https://doi.org/10.58230/27454312.1344.

[3] A. Kilawati, M. Zulham, and Sunardin, “PENGEMBANGAN BUKU BACA TULIS AKSARA LONTARA’ BERBASIS BUDAYA BUGIS UNTUK SEKOLAH DASAR,” no. Vol. 14 No. 2 (2023): Jurnal Pendidikan Dasar, Dec. 2023, doi: 10.21009/jpd.v14i2.

[4] Rismayanti, Apriyanto, and R. Suppa, “APLIKASI PEMBELAJARAN AKSARA LONTARA BUGIS BERBASIS ANDROID PADA SDN 152 CENNING DI DESA SALOBONGKO KECAMATAN MALANGKE BARAT,” *Jurnal Informatika dan Teknik Elektro Terapan*, vol. 13, no. 2, Apr. 2025, doi: https://doi.org/10.23960/jitet.v13i2.6075.

[5] Agus Tri Haryanto, “Digitalisasi Aksara Lontara adalah Mimpi Lama yang Segera Terwujud,” Nov. 07, 2020. Accessed: Apr. 22, 2025. [Online]. Available: https://inet.detik.com/cyberlife/d-5245495/digitalisasi-aksara-lontara-adalah-mimpi-lama-yang-segera-terwujud

[6] M. Sabri, Nurhayati, and Syahrir, “Rancang Bangun Aplikasi Pembelajaran Aksara Lontara Bugis Makassar Berbasis Mobile,” Makassar, Oct. 2020.

[7] M. Mustafa, F. Wajidi, A. Amirul, A. Cirua, and N. Nur, "Pengenalan Huruf Aksara Lontara Menggunakan Metode *Convolutional Neural Network*," *Journal of Computer and Information System (J-CIS*, vol. 7, no. 1, pp. 34–42, 2024, doi: 10.31605/jcis.v7i1.

[8] W. Arthanugraha, Z. Zainuddin, and A. L. Arda, "Implementation of *Convolutional Neural Network* in Recognize Lontara Text," *Jurnal Informatika*, vol. 11, no. 2, pp. 65–72, Aug. 2024, doi: 10.31294/inf.v11i2.20328.

[9] R. Saneval Arhinza, A. Puspita Sari, and F. Ali Akbar, "Klasifikasi Citra Aksara Lontara menggunakan K-NN dan Ekstraksi Fitur HOG," Sep. 2024. doi: http://dx.doi.org/10.29407/gj.v8i2.22580.

[10] A. Naufal Indisa, R. Dwiyansaputra, and F. Bimantoro, "IMPLEMENTASI MODEL MACHINE LEARNING PENGENALAN AKSARA BIMA MENGGUNAKAN APLIKASI PEMBELAJARAN AKSARA BIMA BERBASIS ANDROID (MACHINE LEARNING MODEL," Jun. 2023.

[11] Abdul Rahim, Febryanti Sthevanie, and Kurniawan Nur Ramadhani, "Klasifikasi Aksara Lontara Dari Sulawesi Selatan Menggunakan CNN," Jan. 2025, doi: http://dx.doi.org/10.25124/logic.v2i2.8812.

[12] N. P. I. P. Nurahdika and J. Sutopo, "ANALISA PERFORMA CONVOLUTION NEURAL NETWORK DALAM PENGENALAN TULISAN TANGAN AKSARA LONTARA PERFORMANCE ANALYSIS OF CONVOLUTIONAL NEURAL NETWORK IN HANDWRITTEN LONTARA SCRIPT RECOGNITION," *Journal of Information Technology and Computer Science (INTECOMS)*, vol. 8, no. 1, 2025.

[13] G. Samara, E. Al-Daoud, N. Swerki, and D. Alzu'bi, "The Recognition of Holy Qur'an Reciters Using the MFCCs' Technique and Deep Learning," *Advances in Multimedia*, vol. 2023, pp. 1–14, Mar. 2023, doi: 10.1155/2023/2642558.