



TUGAS AKHIR - IF184802

DETEKSI AKSARA JAWA MENGGUNAKAN YOLO UNTUK TRANSLITERASI BERBASIS LSTM PADA MANUSKRIP JAWA KUNO

MUHAMMAD ARIF FAIZIN

NRP 05111940000060

Dosen Pembimbing

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

NIP 19710428 199412 2 001

Dini Adni Navastara, S.Kom, M.Sc.

NIP 19851017 201504 2 001

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023

Halaman ini sengaja dikosongkan



TUGAS AKHIR - IF184802

DETEKSI AKSARA JAWA MENGGUNAKAN YOLO UNTUK TRANSLITERASI BERBASIS LSTM PADA MANUSKRIPT JAWA KUNO

Muhammad Arif Faizin

NRP 05111940000060

Dosen Pembimbing

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

NIP 19710428 199412 2 001

Dini Adni Navastara, S.Kom, M.Sc.

NIP 19851017 201504 2 001

Program Studi S1 Teknik Informatika

Departemen Teknik Informatika

Fakultas Teknologi Elektro dan Informatika Cerdas

Institut Teknologi Sepuluh Nopember

Surabaya

2023

Halaman ini sengaja dikosongkan



FINAL PROJECT - IF184802

JAVANESE LETTER DETECTION USING YOLO FOR TRANSLITERATION BASED ON LSTM IN JAVANESE ANCIENT MANUSCRIPTS

Muhammad Arif Faizin

NRP 05111940000060

Advisors

Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.

NIP 19710428 199412 2 001

Dini Adni Navastara, S.Kom, M.Sc.

NIP 19851017 201504 2 001

Study Program Bachelor of Informatics

Department of Informatics

Faculty of Intelligent Electrical and Informatics Technology

Institut Teknologi Sepuluh Nopember

Surabaya

2023

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN

DETEKSI AKSARA JAWA MENGGUNAKAN YOLO UNTUK TRANSLITERASI BERBASIS LSTM PADA MANUSKRIP JAWA KUNO

TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Memperoleh gelar Sarjana Komputer pada
Program Studi S-1 Teknik Informatika
Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh : **Muhammad Arif Faizin**

NRP. 05111940000060

Disetujui oleh Tim Penguji Tugas Akhir:

1. Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.
2. Dini Adni Navastara, S.Kom, M.Sc.
3. Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.
4. Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

Pembimbing

Ko-pembimbing

Penguji

Penguji

SURABAYA
Juli, 2023

Halaman ini sengaja dikosongkan

APPROVAL SHEET

JAVANESE LETTER DETECTION USING YOLO FOR TRANSLITERATION BASED ON LSTM IN JAVANESE ANCIENT MANUSCRIPTS

FINAL PROJECT

Submitted to fulfill one of the requirements
for obtaining a degree Computer Master at
Undergraduate Study Program of S-1 Informatics Engineering
Department of Informatics Engineering
Faculty of Intelligent Electrical and Information Technology
Institut Teknologi Sepuluh Nopember

By: **Muhammad Arif Faizin**

NRP. 05111940000060

Approved by Final Project Examiner Team:

1. Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.
2. Dini Adni Navastara, S.Kom, M.Sc.
3. Dr. Anny Yuniarti, S.Kom., M.Comp.Sc.
4. Shintami Chusnul Hidayati, S.Kom., M.Sc., Ph.D

Advisor 
Co-Advisor 
Examiner 
Examiner 

SURABAYA
July, 2023

Halaman ini sengaja dikosongkan

PERNYATAAN ORISINALITAS

Yang bertanda tangan di bawah ini:

Nama mahasiswa / NRP : Muhammad Arif Faizin / 05111940000060

Departemen : Teknik Informatika

Dosen Pembimbing 1 / NIP : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. / 19710428
199412 2 001

Dosen Pembimbing 2 / NIP : Dini Adni Navastara, S.Kom, M.Sc. / 19851017 201504 2 001

Dengan ini menyatakan bahwa Tugas Akhir dengan judul “Deteksi Aksara Jawa Menggunakan YOLO untuk Transliterasi Berbasis LSTM pada Manuskip Jawa Kuno” adalah hasil karya sendiri, bersifat orisinal, dan ditulis dengan mengikuti kaidah penulisan ilmiah.

Bilamana di kemudian hari ditemukan ketidaksesuaian dengan pernyataan ini, maka saya bersedia menerima sanksi sesuai dengan ketentuan yang berlaku di Institut Teknologi Sepuluh Nopember.

Surabaya, 27 Juli 2023

Mahasiswa



Muhammad Arif Faizin

NRP. 05111940000060

Mengetahui,

Dosen Pembimbing 1

Dosen Pembimbing 2



Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.
NIP. 19710428 199412 2 001



Dini Adni Navastara, S.Kom, M.Sc.
NIP. 19851017 201504 2 001

Halaman ini sengaja dikosongkan

STATEMENT OF ORIGINALITY

The undersigned below:

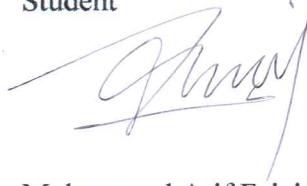
Name of student / NRP : Muhammad Arif Faizin / 05111940000060
Department : Teknik Informatika
Advisor 1 / NIP : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom. / 19710428
199412 2 001
Advisor 2 / NIP : Dini Adni Navastara, S.Kom, M.Sc. / 19851017 201504 2 001

Hereby declare that Final Project with the title of "Javanese Letter Detection using YOLO for Transliteration Based on LSTM in Javanese Ancient Manuscripts" is the result of my own work, is original, and is written by following the rules of scientific writing.

If in the future there is a discrepancy with this statement, then I am willing to accept sanctions in accordance with the provisions that apply at Institut Teknologi Sepuluh Nopember.

Surabaya, 27 July 2023

Student

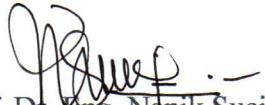


Muhammad Arif Faizin
NRP.

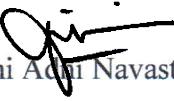
Acknowledge,

Advisor 1

Advisor 2



Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.
NIP. 19710428 199412 2 001



Dini Adni Navastara, S.Kom, M.Sc.
NIP. 19851017 201504 2 001

Halaman ini sengaja dikosongkan

DETEKSI AKSARA JAWA MENGGUNAKAN YOLO UNTUK TRANSLITERASI BERBASIS LSTM PADA MANUSKRIP JAWA KUNO

Nama Mahasiswa / NRP : Muhammad Arif Faizin / 05111940000060
Departemen : Teknik Informatika FTEIC - ITS
Dosen Pembimbing : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.
Dini Adni Navastara, S.Kom, M.Sc.

Abstrak

Keberadaan manuskrip di Indonesia harus dijaga dan dipertahankan, karena manuskrip mampu mengungkap pola pikir masyarakat pada masa itu. Manuskrip tersebar di seluruh dunia, namun dari 121.668 judul manuskrip, hanya sekitar 19 ribu yang sudah didigitalkan. Sebagian besar manuskrip dalam kondisi rusak dan membutuhkan ahli untuk memprosesnya, dengan memakan waktu yang lama dan biaya yang besar. Sementara itu, beberapa penelitian tentang pengolahan citra digital sedang berkembang. Penelitian tentang metode deteksi YOLO pada aksara Jawa menunjukkan performa model yang bagus dan cepat. Sementara itu, penelitian tentang transliterasi aksara Jawa masih minim, metode LSTM dinilai cukup relevan oleh peneliti untuk meningkatkan proses transliterasi aksara Jawa.

Pada penelitian ini, *dataset* yang dibangun adalah *dataset* manuskrip aksara Jawa berdasarkan manuskrip Serat Sewaka sebanyak 60 citra beserta anotasinya yang dinamakan *Handwritten Javanese Character on Sewaka Manuscripts Detection* (HJCS_DETC). Sedangkan untuk data latih, dataset perlu diproses menggunakan augmentasi data dan *split* yang dinamakan dengan HJCS_DETC_SPLIT. Data latih kemudian dilatih menggunakan model YOLOv5 dan menghasilkan *bounding box* untuk proses transliterasi. Proses transliterasi dilakukan dengan mendeteksi *midline*, mengurutkan karakter, pengenalan suku kata, memecahkan aksara *pasangan*, serta penggabungan dengan *dataset* terjemah transliterasi. Kemudian data diproses menggunakan model LSTM sebagai model transliterasi.

Hasil evaluasi terhadap model deteksi menunjukkan *F1 score* maksimal dengan nilai 83,2% dan nilai *mAP* 81,4% pada skenario 9 yang menggunakan *dataset* augmentasi dan citra *split* 5 terhadap 67 kelas aksara Jawa. Hasil evaluasi terhadap model transliterasi mendapatkan model dengan nilai CER 16,7% dan WER 23,3% pada model BiLSTM menggunakan *optimizer* Adam dan *learning rate* 0,001.

Kata kunci: Aksara Jawa, Deteksi Karakter, LSTM, Transliterasi, YOLO

Halaman ini sengaja dikosongkan

JAVANESE LETTER DETECTION USING YOLO FOR TRANSLITERATION BASED ON LSTM IN JAVANESE ANCIENT MANUSCRIPTS

Student Name / NRP: Muhammad Arif Faizin / 05111940000060
Department : Informatics Engineering ELECTICS - ITS
Advisor : Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom.
 Dini Adni Navastara, S.Kom, M.Sc.

Abstract

The existence of manuscripts in Indonesia must be maintained because manuscripts are able to reveal the mindset of the people at that time. Of the 121,668 manuscript titles spread across the world, only about 19,000 have been digitized. Most manuscripts are in damaged condition and require experts to process them within a long time and large costs. Meanwhile, some research on digital image processing is developed. Research on the YOLO detection method in Javanese script shows good and fast model performance. Meanwhile, research on the transliteration of Javanese script is still minimal. LSTM method is considered quite relevant by researchers to improve the process of Javanese script transliteration.

The dataset built in this study is a Javanese script dataset based on 60 images and annotations of the Serat Sewaka manuscript called *Handwritten Javanese Character on Sewaka Manuscripts Detection* (HJCS_DETC). As for training data, the dataset needed to be processed using data augmentation and splitting called HJCS_DETC_SPLIT. The training data was trained using YOLOv5 and generated bounding box for the transliteration process. The transliteration process was carried out by detecting midlines, sequencing characters, recognizing syllables, breaking *pasangan* characters, and combining with transliterated translation datasets. Next, the data was processed using the LSTM as the transliteration model.

The evaluation results of the detection model showed a maximum F1 score of 83.2% and 81.4% mAP values in scenario 9 using augmentation datasets and 5 split images for 67 Javanese script classes. The evaluation results of the transliteration model showed with a CER of 16.7% and WER of 23.3% in the BiLSTM model with optimizer Adam and learning rate 0.001.

Keywords: Javanese Letter, Character Detection, LSTM, Transliteration, YOLO

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur kepada Allah Yang Maha Esa atas segala karunia dan rahmat-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul “Deteksi Aksara Jawa Menggunakan YOLO untuk Transliterasi Berbasis LSTM pada Manuskrip Jawa Kuno”. Harapan dari penulis, semoga apa yang tertulis di dalam buku tugas akhir ini dapat bermanfaat bagi pengembangan ilmu pengetahuan saat ini dan ke depannya, serta dapat memberikan kontribusi yang nyata.

Dalam pelaksanaan dan pembuatan tugas akhir ini tentunya sangat banyak bantuan yang penulis terima dari berbagai pihak, tanpa mengurangi rasa hormat penulis ingin mengucapkan terima kasih sebesar-besarnya kepada:

1. Allah SWT. Dan Nabi Muhammad SAW. yang telah membimbing penulis selama hidup.
2. Keluarga penulis (Ayah, Ibu, dan keluarga penulis yang lain) yang selalu memberikan dukungan baik berupa doa, moral, dan material yang tak terhingga kepada penulis, sehingga penulis dapat menyelesaikan Tugas Akhir ini.
3. Ibu Prof. Dr. Eng. Nanik Suciati, S.Kom, M.Kom dan Ibu Dini Adni Navastara, S.Kom., M.Sc. selaku pembimbing I dan II yang telah membimbing, memberikan motivasi, dan masukan dalam menyelesaikan Tugas Akhir ini.
4. Seluruh mahasiswa Teknik Informatika ITS angkatan 2019 yang telah menemani penulis dan berjuang bersama dalam menyelesaikan Tugas Akhir ini.
5. Semua teman dekat penulis yang memberikan semangat dan menghibur penulis selamaengerjaan Tugas Akhir.
6. Serta semua pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini yang tidak dapat disebutkan penulis satu persatu.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini. Namun, penulis memohon maaf apabila terdapat kekurangan, kesalahan maupun kelalaian yang telah penulis lakukan. Kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan selanjutnya. Selain itu, penulis berharap laporan Tugas Akhir ini dapat berguna bagi pembaca secara umum. Tetaplah berjuang karena akan selalu ada jalan untuk orang yang tidak pernah menyerah. Semoga kita semua selalu diberi kebahagiaan lahir dan batin dan kesuksesan dunia akhirat. Aamiin.

Surabaya, 20 Juli 2023

Penulis
Muhammad Arif Faizin

Halaman ini sengaja dikosongkan

DAFTAR ISI

HALAMAN JUDUL	i
LEMBAR PENGESAHAN	v
PERNYATAAN ORISINALITAS	ix
ABSTRAK	xiii
KATA PENGANTAR	xvii
DAFTAR ISI	xix
DAFTAR GAMBAR	xxi
DAFTAR TABEL	xxiii
DAFTAR SINGKATAN	xxv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Permasalahan	2
1.3 Batasan Masalah	2
1.4 Tujuan	2
1.5 Manfaat	2
BAB II TINJAUAN PUSTAKA	3
2.1 Penelitian Terkait	3
2.2 Dasar Teori	4
2.2.1 Aksara Jawa	4
2.2.2 <i>Image Processing</i>	6
2.2.2 <i>Transfer Learning</i>	8
2.2.3 <i>Convolutional Neural Network</i>	9
2.2.4 <i>You Only Look Once (YOLO)</i>	11
2.2.5 Model <i>Neural Machine Translation</i>	12
2.2.6 Evaluasi Model Deteksi	14
2.2.7 Evaluasi Model Transliterasi	16
BAB III METODOLOGI DAN IMPLEMENTASI	17
3.1 Perancangan Sistem	17
3.1.1 Membangun <i>Dataset</i> Aksara Jawa	17
3.1.2 Proses Latih Model Deteksi	22
3.1.3 Evaluasi Model Deteksi	24
3.1.4 Praproses <i>Dataset</i> Transliterasi	24
3.1.5 Proses Latih Model Transliterasi	30

3.1.6	Evaluasi Model Transliterasi	31
3.2	Peralatan Pendukung.....	31
3.3	Implementasi dan Uji Coba.....	31
3.3.1	Membangun <i>Dataset</i> Aksara Jawa.....	32
3.3.2	Proses Latih Model Deteksi	34
3.3.3	Evaluasi Model Deteksi	35
3.3.4	Praproses <i>Dataset</i> Transliterasi.....	35
3.3.5	Proses Latih Model Transliterasi	37
3.3.6	Evaluasi Model Transliterasi	40
	BAB IV HASIL DAN PEMBAHASAN	41
4.1	<i>Dataset</i> Aksara Jawa.....	41
4.1.1	<i>Dataset</i> HJCS_DETC	41
4.1.2	<i>Dataset</i> HJCS_DETC_SPLIT.....	43
4.2	Hasil dan Pembahasan Evaluasi Model Deteksi	44
4.2.1	Evaluasi Model dan Pengaruh <i>Dataset</i>	45
4.2.2	Analisis Kelas Karakter	47
4.2.3	Analisis <i>Overfitting</i> Model.....	48
4.3	Hasil dan Pembahasan Evaluasi Model Transliterasi	49
4.3.1	Perbandingan Model berdasarkan Variasi Data.....	49
4.3.2	Perbandingan Model berdasarkan Jenis Model	50
4.3.3	Perbandingan Model berdasarkan <i>Hyperparameter</i>	52
	BAB V KESIMPULAN DAN SARAN.....	55
5.1	Kesimpulan	55
5.2	Saran.....	55
	DAFTAR PUSTAKA	57
	LAMPIRAN-LAMPIRAN.....	61
	BIODATA PENULIS	85

DAFTAR GAMBAR

Gambar 2.1 Aksara Jawa <i>Wianjana</i>	4
Gambar 2.2 (a) Aksara Jawa <i>Pasangan</i> dan (b) Aksara Jawa <i>Sandhangan</i>	5
Gambar 2.3 Aksara Jawa <i>Murda</i>	5
Gambar 2.4 Beberapa posisi karakter berdasarkan <i>Medial Textline / Midline</i> (Kesiman & Dermawan, 2021)	5
Gambar 2.5 (a) Citra original (b) Citra hasil proses <i>grayscale</i> (Basak dkk., 2022).....	6
Gambar 2.6 (a) Citra original berukuran 1024 x 1024 (b) Hasil <i>gaussian blur</i> dengan <i>kernel</i> 21 x 21 (c) Hasil <i>gaussian blur</i> dengan <i>kernel</i> 43 x 43 (Gonzalez & Woods, 2018).....	6
Gambar 2.7 (a) Citra original (b) Histogram dari citra (c) Hasil <i>otsu thresholding</i> pada citra (Gonzalez & Woods, 2018)	7
Gambar 2.8 Ilustrasi dilasi pada dokumen citra (Gonzalez & Woods, 2018)	8
Gambar 2.9 Ilustrasi pencarian <i>connected component</i> (Gonzalez & Woods, 2018).....	8
Gambar 2.10 <i>Inductive Learning</i> dan <i>Inductive Transfer</i> (Zain & Santoso, 2021).....	9
Gambar 2.11 <i>Flowchart pre-trained</i> model pada <i>Transfer Learning</i> (Wang dkk., 2021)	9
Gambar 2.12 Contoh proses konvolusi pada citra (Yamashita dkk., 2018)	10
Gambar 2.13 Contoh <i>fully-connected layer</i> (Paoletti dkk., 2018).....	10
Gambar 2.14 Arsitektur YOLOv5 (Li dkk., 2022).....	11
Gambar 2.15 Jaringan RNN (Dias dkk., 2022)	12
Gambar 2.16 Jaringan LSTM (Dias dkk., 2022)	12
Gambar 2.17 Jaringan GRU (Abdulwahab dkk., 2018)	13
Gambar 2.18 Arsitektur <i>Bidirectional</i> Model (Pasaribu dkk., 2023)	14
Gambar 2.19 Ilustrasi probabilitas pada <i>confusion matrix</i> (Maxwell dkk., 2021)	15
Gambar 2.20 Ilustrasi <i>Intersection over Union</i> (Maxwell dkk., 2021)	15
Gambar 3.1 Rancangan sistem deteksi transliterasi	17
Gambar 3.2 Alur membangun <i>dataset</i> aksara Jawa	17
Gambar 3.3 (a) Citra asli manuskrip Serat Sewaka (b) Manuskrip yang sudah dipotong	18
Gambar 3.4 Pemrosesan pada dataset HJCS_DETC_SPLIT	19
Gambar 3.5 Proses deteksi <i>bounding box</i>	19
Gambar 3.6 Ilustrasi hasil deteksi <i>bounding box</i> karakter.....	20
Gambar 3.7 Proses pelabelan pada situs <i>online</i> roboflow.com	20
Gambar 3.8 Hasil ilustrasi anotasi <i>bounding box</i> pada citra	20
Gambar 3.9 (a) <i>Grayscale</i> pada citra (b) <i>Otsu thresholding</i> pada citra	21
Gambar 3.10 Rata-rata ukuran citra pada <i>dataset</i>	22
Gambar 3.11 Ilustrasi (a) citra dengan garis <i>split</i> (b) posisi garis <i>split</i> dan <i>midline</i> sebelum di geser (c) posisi garis <i>split</i> setelah di geser ke <i>midline</i> terdekat	22
Gambar 3.12 Rancangan proses latih model deteksi.....	23
Gambar 3.13 Arsitektur Model Deteksi	23
Gambar 3.14 Contoh citra yang akan dideteksi (a) kondisi baik (b) kondisi sedikit rusak	24
Gambar 3.15 Alur praproses <i>dataset</i> transliterasi	24
Gambar 3.16 (a) citra (b) <i>horizontal projection</i> akar suku kata beserta <i>midline</i>	25
Gambar 3.17 Ilustrasi pengurutan karakter akar suku kata	27
Gambar 3.18 Posisi <i>x</i> dan <i>y</i> yang digunakan untuk pengenalan sambungan	27
Gambar 3.19 Contoh hasil pengenalan suku kata.....	28
Gambar 3.20 Ilustrasi pemecahan <i>pasangan</i>	29
Gambar 3.21 Rancangan proses latih model transliterasi.....	30
Gambar 4.1 Beberapa karakteristik dataset HJCS_DETC (a) contoh karakter yang memotong (b) penulisan karakter yang tegak dan cenderung lurus (c) karakter yang buram.....	43

Gambar 4.2 Contoh hasil deteksi pada citra tes	45
Gambar 4.3 Perbandingan <i>mAP validation</i> selama proses latih masing-masing skenario	48
Gambar 4.4 Contoh hasil transliterasi.....	49
Gambar 4.5 Perbandingan nilai <i>loss</i> dan <i>val_loss</i> pada proses latih model BiLSTM	50

DAFTAR TABEL

Tabel 3.1 Contoh <i>dataset</i> transliterasi	25
Tabel 3.2 Klasifikasi kelas karakter berdasarkan posisi.....	26
Tabel 3.3 Klasifikasi batas pengenalan karakter	27
Tabel 3.4 <i>Dataset</i> transliterasi setelah digabungkan	29
Tabel 3.5 Pengacakan urutan suku kata pada <i>dataset</i>	30
Tabel 3.6 Contoh perbedaan nilai metrik CER dan WER	31
Tabel 3.7 Tabel Spesifikasi Perangkat Keras	31
Tabel 3.8 Parameter <i>PyTorch</i> pada proses latih YOLOv5	34
Tabel 4.1 Jumlah karakter Aksara Jawa pada setiap kelas	41
Tabel 4.2 Persebaran klasifikasi kluster kelas	43
Tabel 4.3 Variasi <i>dataset</i> HBCL_DETC_SPLIT	44
Tabel 4.4 Persebaran data <i>training</i> , <i>validation</i> , dan <i>testing</i>	44
Tabel 4.5 Evaluasi Model Deteksi.....	45
Tabel 4.6 Perbedaan jumlah anotasi pada skenario	46
Tabel 4.7 Rata-rata tinggi citra pada skenario	47
Tabel 4.8 Kelas dengan <i>mAP</i> cukup buruk.....	47
Tabel 4.9 Kelas dengan kesamaan bentuk citra.....	48
Tabel 4.10 Perbandingan akurasi model berdasarkan jenis variasi data	49
Tabel 4.11 Perbandingan akurasi model berdasarkan jenis model dan <i>batch size</i>	50
Tabel 4.12 Contoh perbandingan hasil deteksi BiLSTM	51
Tabel 4.13 Beberapa kemunculan objek suku kata yang tidak konsisten	51
Tabel 4.14 Perbandingan akurasi model berdasarkan parameter <i>optimizer</i>	52
Tabel 4.15 Perbandingan akurasi model berdasarkan parameter <i>learning rate</i>	53

Halaman ini sengaja dikosongkan

DAFTAR SINGKATAN

CER	<i>Character Error Rate</i>
CNN	<i>Convolutional Neural Network</i>
BiGRU	<i>Bidirectional Gated recurrent Unit</i>
BiLSTM	<i>Bidirectional Long Short-Term Memory</i>
BiRNN	<i>Bidirectional Recurrent Neural Network</i>
GRU	<i>Gated Recurrent Unit</i>
IoU	<i>Intersection Over Union</i>
LSTM	<i>Long Short-Term Memory</i>
mAP	<i>Mean Average Precision</i>
NLP	<i>Natural Language Processing</i>
R-CNN	<i>Region-based Convolutional Neural Network</i>
RNN	<i>Recurrent Neural Network</i>
YOLO	<i>You Only Look Once</i>
WER	<i>Word Error Rate</i>

Halaman ini sengaja dikosongkan

BAB I PENDAHULUAN

1.1 Latar Belakang

Keberadaan manuskrip di Indonesia menjadi salah satu peninggalan khazanah budaya masa lampau yang perlu untuk dijaga dan dipertahankan. Manuskrip mampu mengungkap pola pikir dan aktivitas kehidupan masyarakat Nusantara di masa itu. Oleh karena itu informasi yang di dalamnya sangat penting untuk dapat diungkap dan disampaikan kepada masyarakat (Hendrawati, 2018).

Manuskrip ini tersebar di berbagai tempat di pulau Jawa bahkan di luar Jawa, dengan kondisi yang sangat bervariasi. Data *Grand Design Pengolahan Naskah Nusantara* di Perpustakaan Nasional mencatat, setidaknya ada 121.668 judul manuskrip Nusantara yang tersebar di seluruh dunia. Indonesia memegang sekurangnya 82.821 manuskrip, dengan baru sekitar 19 ribu manuskrip yang dapat diselamatkan (Sucayyo, 2023). Banyak lembaga yang mencoba melestarikan manuskrip-manuskrip tersebut melalui proses alih media, alih aksara, dan alih bahasa. Di antaranya adalah proses alih media berupa digitalisasi manuskrip yang dilakukan oleh Perpustakaan Nasional Republik Indonesia (Perpusnas RI). Sekurangnya 12.572 manuskrip tersedia secara *online* dan gratis di situs KHASTARA (Khasanah Pustaka Nusantara) Perpusnas RI dan dapat digunakan masyarakat secara umum. Sayangnya, kurangnya informasi akan pengolahan citra menyebabkan belum adanya *dataset* aksara Jawa yang dapat digunakan untuk proses deteksi maupun transliterasi secara otomatis.

Pada proses selanjutnya, proses alih aksara dan bahasa mengalami kendala yang cukup sulit. Menurut data pada situs sastra.org, hanya sekitar 2.495 manuskrip yang sudah dialih aksarakan (Paterson, 2023). Manuskrip harus dengan kondisi yang baik untuk dapat dikenali, minimal masih memiliki informasi visual yang cukup pada setiap karakter sehingga bentuk setiap karakter tetap utuh, jelas dan benar. Namun, banyak di antaranya yang rusak, sehingga sulit untuk memahami makna yang tertulis dalam manuskrip tersebut. Dibutuhkan seorang ahli sastra Jawa untuk memahaminya, meski membutuhkan waktu yang cukup lama (Himamunanto & Setyowati, 2017).

Di sisi lain, perkembangan bidang teknologi pengolahan citra digital sekarang sudah cukup cepat untuk melakukan proses ekstraksi fitur citra. Beberapa penelitian pengolahan citra digital berpeluang membantu mempercepat berbagai pekerjaan untuk identifikasi dan penggalian informasi dari manuskrip aksara Jawa. Beberapa penelitian awal sudah pernah dilakukan, seperti proses perbaikan citra (Widiarti dkk., 2014), ekstraksi fitur berupa segmentasi karakter (Arifianto, 2016; Damayanti dkk., 2020) ataupun baris (Widiarti & Hartati, 2013).

Penelitian mengenai deteksi aksara Jawa sedang mengalami perkembangan. Ada beberapa metode yang pernah digunakan, seperti metode *Convolutional Neural Network* (CNN) dan metode *You Only Look Once* (YOLO) (Christian Adi Pradhana dkk., 2020; Suciati dkk., 2022). Pada penelitian dengan metode YOLO, model yang dihasilkan cenderung memiliki performa yang bagus dan cepat untuk mendeteksi aksara Jawa.

Sementara itu, penelitian mengenai transliterasi aksara Jawa masih cukup minim. Pada domain yang serupa, penelitian yang dilakukan oleh (Sutramiani, 2022) terhadap transliterasi aksara Bali melakukan proses transliterasi menggunakan aturan pasang aksara. Pada penelitian lainnya, model LSTM memiliki performa yang bagus untuk proses penerjemahan antar bahasa (Fauziyah dkk., 2022). Penerjemahan berbasis *sequence-to-sequence* pada penelitian tersebut dinilai cukup relevan oleh peneliti untuk meningkatkan proses transliterasi pada aksara Jawa.

Sehingga, dibutuhkan *dataset* aksara Jawa berdasarkan manuskrip Jawa kuno untuk membuat model deteksi menggunakan YOLO. Selain itu, pada penelitian ini juga diusulkan model transliterasi menggunakan LSTM untuk meningkatkan efektivitas dan mempercepat proses penggalian informasi pada manuskrip Jawa kuno.

1.2 Rumusan Permasalahan

Dari uraian latar belakang tersebut, dapat dibuat rumusan masalah sebagai berikut :

1. Bagaimana membangun *dataset* aksara Jawa dari manuskrip Jawa kuno?
2. Bagaimana membangun model deteksi aksara Jawa menggunakan YOLO?
3. Bagaimana membangun model transliterasi aksara Jawa dalam aksara Latin menggunakan LSTM?
4. Bagaimana mengukur performa model dalam mendekripsi aksara Jawa?
5. Bagaimana mengukur performa model dalam menransliterasi aksara Jawa dalam aksara Latin?

1.3 Batasan Masalah

Batasan masalah untuk penelitian ini adalah :

1. Sumber *dataset* menggunakan manuskrip Serat Sewaka sebanyak 60 halaman yang diambil dari Website KHASTARA Perpusnas RI (khaustara.perpusnas.go.id).
2. Model deteksi hanya melakukan deteksi terhadap 67 kategori aksara Jawa yaitu 20 aksara *wianjana*, 20 aksara *pasangan wianjana*, 14 aksara *sandhangan*, 3 aksara *tandha*, 7 aksara *murda*, 2 aksara *pasangan murda*, dan 1 aksara *mahaprana*.
3. Transliterasi yang dilakukan pada penelitian ini mencakup pengenalan suku kata aksara Jawa.

1.4 Tujuan

Tujuan dari penelitian ini adalah :

1. Membangun *dataset* aksara Jawa dari manuskrip Jawa kuno
2. Membangun model deteksi aksara Jawa pada manuskrip Jawa menggunakan YOLO.
3. Membangun model transliterasi aksara Jawa dalam aksara Latin menggunakan LSTM.

1.5 Manfaat

Manfaat dari penelitian ini adalah :

1. Memudahkan deteksi aksara Jawa dalam penelitian maupun kehidupan sehari-hari.
2. Melestarikan manuskrip yang menggunakan aksara Jawa.
3. *Dataset* dan model yang dihasilkan dapat digunakan untuk penelitian selanjutnya

BAB II TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Dalam melakukan penelitian ini, peneliti menggunakan serta mengkaji beberapa penelitian terkait *dataset* manuskrip aksara Jawa yang pernah dilakukan sebelumnya. Pemrosesan aksara Jawa dari manuskrip Jawa sudah pernah dilakukan, seperti dalam penelitian (Widiarti dkk., 2014) yang melakukan segmentasi pada manuskrip Jawa PB.A57 menggunakan metode kombinasi *otsu thresholding*, *image projection*, dan *connected component* menghasilkan akurasi 95% untuk segmentasi karakter. Sayangnya pada penelitian tersebut pengujian hanya dilakukan pada sebuah citra saja. Sementara itu (Arifianto, 2016) menggunakan metode *adaptive thresholding* untuk melakukan segmentasi karakter pada citra tulisan tangan manuskrip Jawa milik R. Agus Sudjatmoko. Metode tersebut menggunakan nilai *threshold* berdasarkan variasi intensitas tiap lokal *window* dan menghasilkan akurasi sebesar 88,6% dari 30 citra manuskrip Jawa. Kemudian pada penelitian lain, (Damayanti dkk., 2020) menggunakan *connected component labeling* untuk melakukan segmentasi pada 30 citra dari manuskrip berbeda. Hasilnya, metode tersebut menghasilkan akurasi 80,9% untuk rata-rata data uji.

Metode lain dalam ekstraksi fitur juga pernah dilakukan pada penelitian (Sugianela & Suciati, 2019) menggunakan *Histogram of Oriented Gradient* (HOG) terhadap 10 citra dari dokumen aksara Jawa. Dari hasil pengujian didapatkan nilai akurasi optimal ROI pada sebuah citra mencapai 93,3%. Sementara itu, pada penelitian (Indrawijaya & Adipranata, 2015) ekstraksi fitur yang digunakan adalah menggunakan serangkaian pemrosesan yang dinamakan *skeletonizing* pada karakter aksara Jawa yang telah tersegmentasi. Dari hasil penelitian didapatkan bahwa pemrosesan berhasil mengekstrak fitur-fitur pada aksara Jawa dengan cukup baik, namun sayangnya tidak ada nilai evaluasi yang dilakukan pada hasil uji. Metode lain juga pernah digunakan pada penelitian (Nugroho dkk., 2021) menggunakan Zernike Moment terhadap citra aksara Jawa Kontemporer dan aksara Jawa Kawi. Hasil dari penelitian tersebut didapatkan kemiripan optimal dengan nilai selisih *Zernike Moment Descriptor* (ZMD) 20,31% dan kemiripan 79,69% pada aksara Jawa.

Selain itu, penelitian terkait pengenalan karakter aksara Jawa pernah dilakukan oleh (Christian Adi Pradhana dkk., 2020) menggunakan CNN untuk mengenali 2800 citra aksara Jawa tulisan tangan dari siswa SMPN 1 Magelang ke dalam 20 aksara Jawa dasar (*nglegena / wianjana*). Dari hasil latih, pengenalan tersebut mendapatkan akurasi 95,35% untuk data latih dan 73% untuk data uji.

Penelitian pengenalan karakter aksara Jawa dalam manuskrip juga pernah dilakukan, seperti (Novaliandy & Widiarti, 2022) menggunakan jaringan syaraf tiruan *backpropagation* untuk mendeteksi 1463 data citra dari manuskrip Hamong Tani ke dalam 20 aksara Jawa dasar. Hasilnya, model tersebut mampu menghasilkan akurasi 92,1% pada data latih dan 76,1% pada data uji. Selain itu, ada penelitian dari (Putra, 2020) menggunakan 4 citra manuskrip berjudul “Kocap wulangira Sultan Hamengkubuwana I” ke dalam 20 aksara *wianjana* dan 4 aksara *sandhangan*. Penelitian tersebut menggunakan *pre-trained* model VGG-16 dan menghasilkan akurasi 95% pada data latih dan 87% pada data validasi, serta nilai *Kappa* 0,86.

Penelitian pada domain aksara lainnya juga sudah pernah dilakukan, pada aksara Bali misalkan oleh (Suciati dkk., 2022) yang mendeteksi karakter aksara Bali pada Manuskrip Lontar sebanyak 200 citra menggunakan metode LONTAR_DETC berbasis YOLOv4 ke dalam 55 kelas aksara. Dari penelitian tersebut, diperoleh akurasi tertinggi *mAP* sebesar 99,55% menggunakan data dengan augmentasi *grayscale* dan *adaptive gaussian thresholding*. Penelitian pada aksara Sunda dilakukan oleh (Paulus dkk., 2018) menggunakan ekstraksi fitur *Histogram of Oriented Gradient* (HOG) dan model *K-Nearest Neighbor* untuk mengklasifikasikan 60 kelas karakter aksara Sunda. Penelitian tersebut menghasilkan model

optimal pada *dataset* original dengan tambahan augmentasi dengan akurasi 93,2% serta akurasi optimal menggunakan *cross validation* pada KF10 sebesar 77%. Pada aksara matematis Bengali, (Islam & Khan, 2019) menggunakan YOLOv3 untuk mengenali 140 citra berisikan ekspresi matematika, dan menghasilkan akurasi 98,6% pada data latih dan 99,62% pada data uji NumtaDB dan 99.08% pada data uji CMATERdb.

Selain itu penelitian tentang transliterasi aksara juga telah pernah dilakukan pada domain dan target yang berbeda. Penelitian menggunakan *deep learning* (Kesiman & Dermawan, 2021) pada domain aksara Bali, menggunakan *dataset* AMADI_LontarSet sebanyak 230 citra untuk mengenali kata yang telah ditransliterasi. Pengenalan kata dilakukan dengan menggunakan model LSTM dengan BGT (*Backpropagation Through Time*) untuk mendeteksi kata. Hasil dari evaluasi 10,475 citra uji menghasilkan nilai CER 19,78%. Selain itu pada domain aksara Jawa, (Widiarti dkk., 2018) juga melakukan transliterasi menggunakan *database of features* untuk menentukan *output* berbentuk kata. Menggunakan *probability calculation* subproses, ekstraksi fitur dari citra di proses untuk menentukan kata yang paling dekat berdasarkan database dan menghasilkan akurasi antara 69,20% - 87,29% terhadap sebuah citra.

Sedangkan penelitian tentang transliterasi aksara menggunakan aturan (*rule-based*), pernah dilakukan oleh (Sutramiani, 2022) pada aksara Bali. Transliterasi dilakukan dengan menggunakan aturan pasang aksara pada yang relevan dengan aksara Bali. Dari hasil pengenalan suku kata, didapatkan akurasi sebesar 71,18% dari aturan tersebut.

2.2 Dasar Teori

Dalam penelitian ini, terdapat beberapa dasar teori yang digunakan dalam rancangan maupun implementasi. Subbab ini menjelaskan teori yang berkaitan dengan penelitian. Subbab ini terbagi menjadi beberapa bagian, yaitu Aksara Jawa, *Image Processing*, *Transfer Learning*, *Convolutional Neural Network* (CNN), *You Only Look Once* (YOLO), Model *Neural Machine Translation*, Evaluasi Model Deteksi, dan Evaluasi Model Transliterasi.

2.2.1 Aksara Jawa

Aksara Jawa merupakan karakter Jawa kuno yang sudah digunakan sejak abad ke-17 selama masa kerajaan Mataram. Sejarah Indonesia kebanyakan ditulis menggunakan karakter ini yang tertulis pada batu-batuan. Beberapa orang lokal masih menggunakan karakter ini sebagai contoh untuk nama tempat, tempat turis, pernikahan, batu nisan, dan lain-lain.

Dibandingkan dengan karakter Latin, aksara Jawa mempunyai struktur dan bentuk yang berbeda, serta memiliki karakteristik tersendiri. Di antaranya seperti (1) tidak memiliki spasi untuk memisahkan kata (2) bentuk karakter berbeda jika terdapat sambungan (*sandhangan*, *pasangan*, atau *tandha*) (3) memiliki aksara tingkat tinggi, namun memiliki penyebutan yang sama yang digunakan untuk menyebutkan istilah ataupun karakter khusus (*murdha*, *rekan*, *mahaprana*). Basis dari aksara Jawa disebut dengan *Wianjana*, yang mana terdiri dari 20 aksara yang dinamakan dengan *Dentawianjana* yang dapat dilihat pada Gambar 2.1.

Aksara Wianjana				
හ	න	ڽ	ڒ	ߞ
ha	na	ca	ra	ka
ڽ	ڽා	ڽ	ڽ	ڽප
da	ta	sa	wa	la
ڽ	ڽ	ڽ	ڽ	ڽ
pa	dha	ja	ya	nya
ڽ	ڽ	ڽ	ڽ	ڽ
ma	ga	ba	tha	nga

Gambar 2.1 Aksara Jawa *Wianjana*

Aksara *Pasangan* merupakan karakter yang digunakan untuk *pasangan* dari aksara *Wianjana*, di mana setiap karakter memiliki pasangannya, aksara ini digunakan untuk menunjukkan huruf mati dalam kalimat. Aksara *Pasangan* bisa dilihat pada Gambar 2.2. Selain itu terdapat aksara *Sandhangan*. Aksara *Sandhangan* adalah karakter spesial yang biasanya digunakan sebagai karakter pelengkap, vokal dan konsonan yang biasanya digunakan dalam bahasa sehari-hari. *Sandhangan* dapat dilihat pada Gambar 2.3. Terdapat bentuk lain dari Aksara Jawa pada umumnya, yaitu Aksara *Murda*. Aksara *Murda* yaitu aksara yang digunakan untuk menuliskan nama seseorang, gelar kehormatan, nama instansi, atau nama tempat. Aksara *Murda* bisa dilihat pada Gambar 2.4.

Pasangan Aksara Wianjana					Sandhangan				
ha	na	ca	ra	ka	wulu	pepet	suku	taling	taling tarung
da	ta	sa	wa	la	cека	лайар	пангкон	пенгкол	вигнан
pa	dha	ja	ya	nya	cакра	лакра керет	па cereк	нга леlet	па lингса
ma	ga	ba	tha	nga	па мадья	purwa па			

(a)

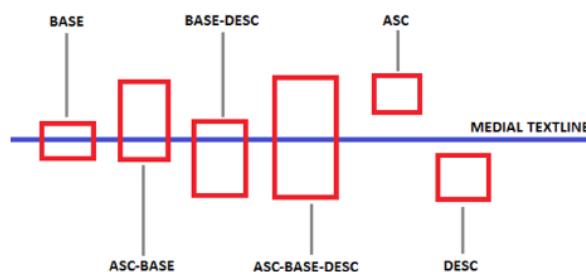
(b)

Gambar 2.2 (a) Aksara Jawa *Pasangan* dan (b) Aksara Jawa *Sandhangan*

Aksara Murda					Pasangan Murda			
na	ka	ta	sa	na	ka	ta	sa	
pa	nya	ga	ba	pa	nya	ga	ba	

Gambar 2.3 Aksara Jawa *Murda*

Beberapa bentuk aksara Jawa ada yang dituliskan di atas baris teks utama (sebagai *Ascender*) dan di bawah baris teks utama (sebagai *Descender*) seperti yang ditunjukkan pada Gambar 2.5. Sehingga pada umumnya, bunyi ucapan pada suku kata aksara Jawa dapat berubah tergantung dengan posisi karakternya. Pengucapan tersebut bergantung pada aturan fonologis tertentu pada aksara Jawa.

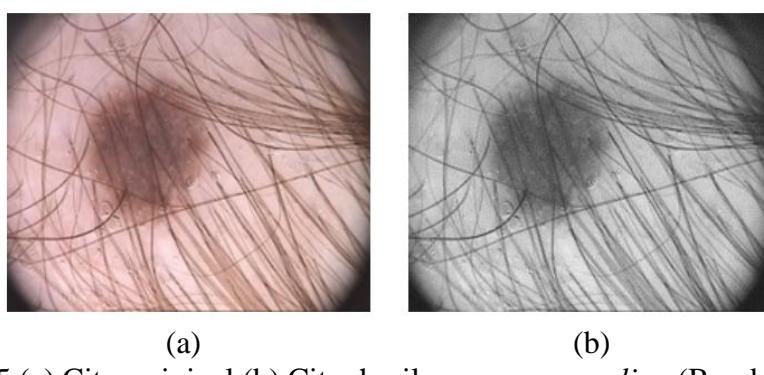
Gambar 2.4 Beberapa posisi karakter berdasarkan *Medial Textline / Midline* (Kesiman & Dermawan, 2021)

2.2.2 Image Processing

Pengolahan citra atau *image processing* merupakan salah satu teknik upaya untuk mendapatkan fitur dari citra. Citra dengan kondisi yang bermacam-macam perlu untuk diproses terlebih dahulu agar memiliki bentuk atau kondisi yang diinginkan. Ada beberapa macam jenis *image processing* yang dapat dilakukan terhadap sebuah citra, seperti peningkatan citra, segmentasi citra, deteksi dan filter tepi, ekstraksi fitur, pencocokan dan pembandingan citra, restorasi citra, pengenalan pola, serta rekonstruksi citra.

a) Grayscale

Grayscale merupakan salah satu peningkatan citra (*image enhancement*), yang mengonversi citra dengan komposisi warna RGB (*red green blue*) atau komposisi warna lainnya ke dalam *grayscale* (skala hitam putih). Biasanya proses ini digunakan untuk mengurangi fitur yang tidak digunakan pada kasus tertentu, seperti pengenalan tulisan tangan atau deteksi tepi objek. Contoh hasil dari *grayscale* pada citra kulit seperti pada Gambar 2.6.



Gambar 2.5 (a) Citra original (b) Citra hasil proses *grayscale* (Basak dkk., 2022)

b) Gaussian Blur

Salah satu peningkatan citra lainnya adalah *gaussian blur*, metode ini digunakan untuk menghilangkan *noise* pada citra serta mendapatkan informasi fitur utama dari sebuah objek. *Gaussian blur* mengubah citra dengan menggunakan persamaan *gaussian* seperti pada Persamaan (2.1) untuk menentukan matriks *kernel* filter. Matriks *kernel* filter dapat menghasilkan citra yang lebih buram karena membuat nilai piksel pada citra relatif terhadap piksel di sekitarnya. Hasil dari proses *gaussian blur* dapat dilihat pada Gambar 2.7.

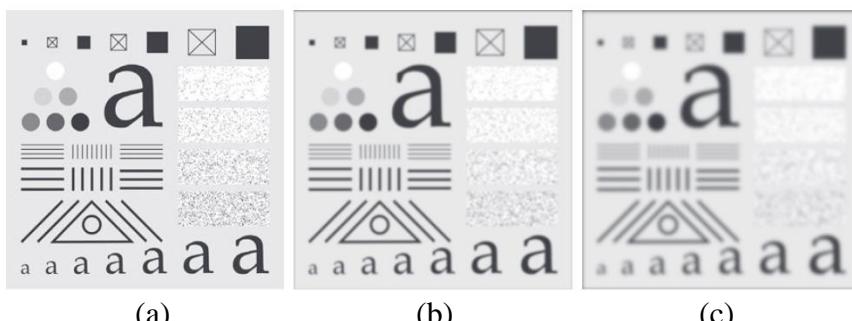
$$G_{\sigma}(s, t) = K e^{-\frac{s^2+t^2}{2\sigma^2}} \quad (2.1)$$

K = matriks *kernel*

s = lebar matriks *kernel*

t = tinggi matriks *kernel*

σ = standar deviasi



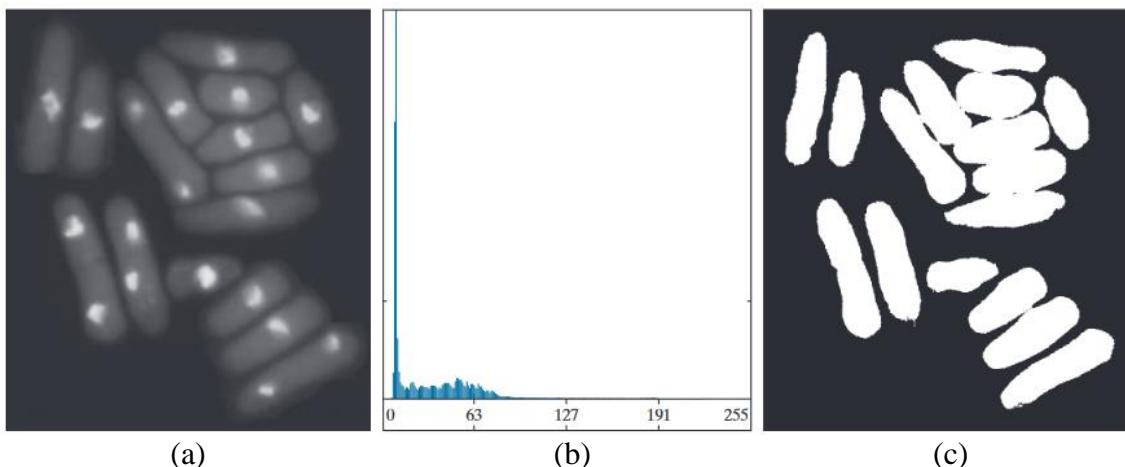
Gambar 2.6 (a) Citra original berukuran 1024 x 1024 (b) Hasil *gaussian blur* dengan *kernel* 21 x 21 (c) Hasil *gaussian blur* dengan *kernel* 43 x 43 (Gonzalez & Woods, 2018)

c) *Otsu Thresholding*

Otsu thresholding merupakan salah satu jenis pemrosesan citra untuk mendapatkan citra biner. Proses ini menganggap sebuah citra sebagai *bimodal* histogram atau citra dengan dua bagian yang membagi piksel pada citra menjadi histogram dengan dua jenis modal, yakni *intra-class* dan *inter-class* (Boudraa dkk., 2019). Pemrosesan ini membagi citra dengan meminimalkan varian *intra-class* dan memaksimalkan varian *inter-class* seperti pada Persamaan (2.2). Hasil dari pembagian tersebut kemudian menjadi *foreground* (teks atau objek) dan *background* (latar belakang) seperti pada Gambar 2.8.

$$T = \underset{t}{\operatorname{argmax}} \left[v_b = |v_0 - v_1| = \omega_0(t)\omega_1(t)(\mu_0(t) - \mu_1(t))^2 \right] \quad (2.2)$$

- t = nilai *threshold*
- $v_{0,1}$ = varian *intra-class*
- $\omega_{0,1}(t)$ = probabilitas *class*
- $\mu_{0,1}(t)$ = nilai rata-rata *class*



Gambar 2.7 (a) Citra original (b) Histogram dari citra (c) Hasil *otsu thresholding* pada citra (Gonzalez & Woods, 2018)

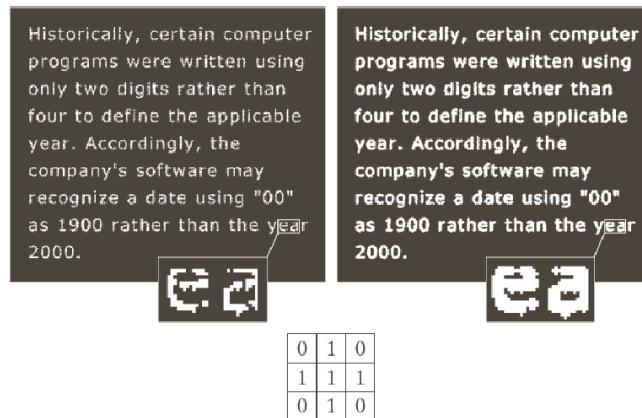
d) Dilasi

Dilasi adalah salah satu proses restorasi citra yang bertujuan untuk mengembalikan atau meminimalkan fitur yang hilang dari sebuah citra dengan cara melapisi struktur aslinya. Dilasi biasanya digunakan untuk memperbaiki citra yang rusak atau terputus, sehingga struktur asli elemen citra terlapisi dengan elemen dilasi. Secara matematis, proses dilasi dapat dirumuskan dalam Persamaan (2.3) dan (2.4).

$$A \oplus B = \left\{ z \mid (\hat{B})_z \cap A \neq \emptyset \right\} \quad (2.3)$$

$$A \oplus B = \left\{ z \mid [(\hat{B})_z \cap A] \subseteq A \right\} \quad (2.4)$$

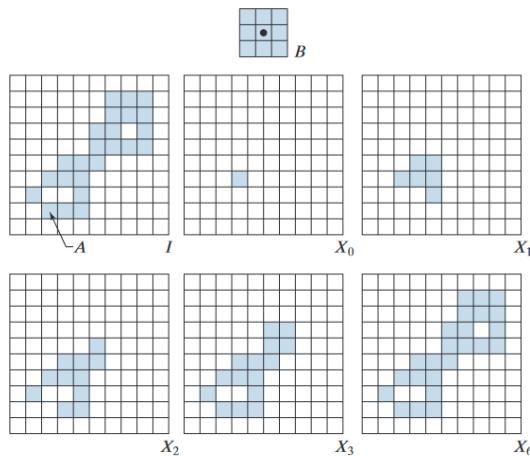
Dengan A adalah citra dan B adalah elemen penstruktur dilasi, maka proses dilasi A terhadap B adalah serangkaian penggantian z . Sehingga setiap elemen dari B berpotongan minimal satu elemen dari citra A . Proses dilasi dilakukan dengan membandingkan piksel-piksel citra *input* dengan elemen penyusunnya, untuk menghasilkan citra yang lebih tebal mengikuti objeknya seperti pada Gambar 2.9.



Gambar 2.8 Ilustrasi dilasi pada dokumen citra (Gonzalez & Woods, 2018)

e) *Connected Component*

Connected component merupakan salah satu proses ekstraksi fitur pada citra. Proses ini digunakan untuk mengelompokkan elemen yang terhubung untuk mengetahui sebuah objek. Secara berulang proses ini dapat diilustrasikan pada Gambar 2.10. Masing-masing iterasi dapat dirumuskan seperti pada Persamaan (2.5).



Gambar 2.9 Ilustrasi pencarian *connected component* (Gonzalez & Woods, 2018)

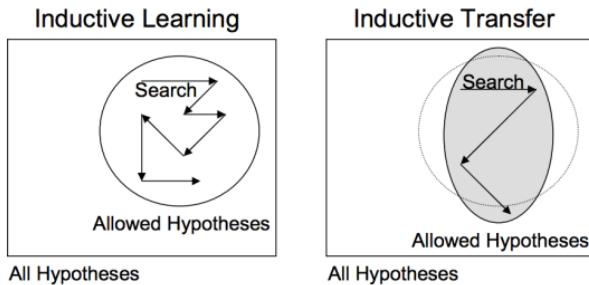
$$X_k = (X_{k-1} \oplus B) \cap I \quad k = 1, 2, 3, \dots \quad (2.5)$$

Dengan B adalah elemen struktur dan I adalah elemen citra target yang akan di deteksi, serta k adalah indeks iterasi dalam proses pencarian komponen. Proses pencarian tersebut berhenti ketika $X_k = X_{k-1}$ atau sudah tidak ada lagi perubahan dari pencarian komponen. Hasil dari proses ini berupa *array* dari seluruh *connected component* yang ditemukan selama iterasi.

2.2.2 Transfer Learning

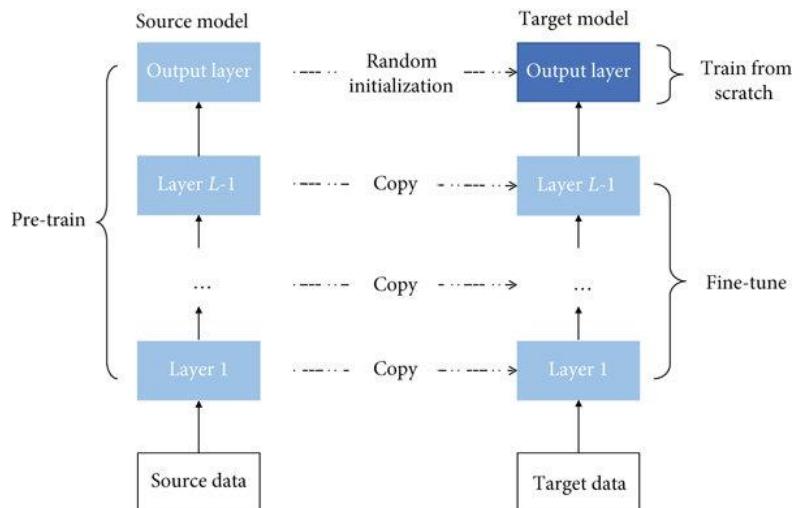
Transfer learning adalah metode *machine learning* di mana model yang dikembangkan untuk suatu tugas digunakan kembali sebagai titik awal untuk model pada tugas kedua. Ini adalah pendekatan populer dalam pembelajaran mendalam di mana model *pre-trained* digunakan sebagai titik awal pada *computer vision* dan tugas *Natural Language Processing* (NLP) mengingat sumber daya komputasi dan waktu yang luas yang diperlukan untuk mengembangkan model jaringan saraf pada masalah ini dan dari lompatan besar dalam keterampilan yang mereka berikan pada masalah terkait. Bentuk pembelajaran transfer yang digunakan dalam pembelajaran mendalam ini disebut transfer induktif (*inductive transfer*).

Seperti pada Gambar 2.11, transfer induktif membuat ruang lingkup model yang mungkin (bias model) dipersempit dengan cara yang menguntungkan dengan menggunakan model fit pada tugas yang berbeda tetapi terkait.



Gambar 2.10 *Inductive Learning* dan *Inductive Transfer* (Zain & Santoso, 2021)

Terdapat dua metode dalam *transfer learning* yaitu menggunakan model yang dikembangkan dan model yang telah dilatih. Model yang telah dilatih ini biasanya disebut *pre-trained model*, dengan struktur yang terlihat pada Gambar 2.12. Pada penulisan ini digunakan model yang telah dilatih atau *pre-trained model*, yaitu model YOLOv5.



Gambar 2.11 *Flowchart pre-trained model pada Transfer Learning* (Wang dkk., 2021)

2.2.3 Convolutional Neural Network

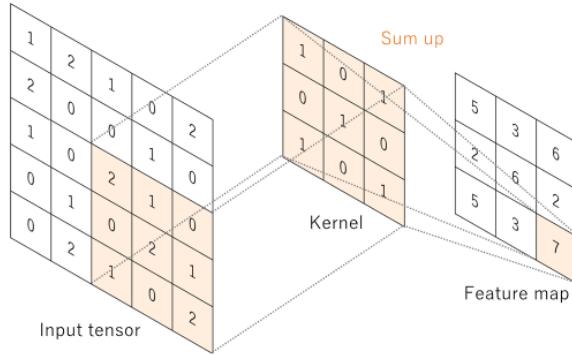
Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data citra. CNN bisa digunakan untuk mendekripsi dan mengenali objek pada sebuah citra. Secara garis besar CNN tidak jauh beda dengan *neural network* biasanya. CNN terdiri dari neuron yang memiliki *weight*, *bias*, dan *activation function*. Proses yang dilakukan oleh *Convolutional Neural Network*, yaitu: *Convolutional Layer*, *Non-Linearity Layer* (ReLU Layer), *Pooling Layer*, dan *Fully-Connected Layer* (Paoletti dkk., 2018).

a) Convolutional Layer

Convolutional Layer adalah *layer* yang bertujuan untuk mengambil informasi posisi dan nilai dari layer sebelumnya dengan konvolusi. Konvolusi dilakukan pada level *neuron* untuk mendapatkan *layer* selanjutnya. *Neuron* melakukan operasi *dot product* antara *weights* dan *region* kecil dari *layer* sebelumnya.

$$z_i^l = B^l + \sum_{j=1}^{k^{l-1}} W_{i,j}^l * z_j^{l-1} \quad i \in [1, k^l] \quad (2.6)$$

Convolution Layer terdiri dari k filter, dengan ukuran $l \times l \times q$ di mana neuronnya menggunakan *weight* dan *bias* yang sama dan mengoneksikan *feature maps input* dengan *feature maps output*. Setiap filter mendeteksi fitur-fitur yang ada di setiap lokasi *input*. *Output* dari *layer l* adalah *feature map* dengan ukuran $d^l \times d^l \times k^l$ yang menyimpan informasi di mana fitur tersebut muncul pada input original. Persamaan *Convolutional Layer* dihitung sebagai Persamaan (2.6), di mana B^l adalah matriks bias dari *layer l* dan $W_{i,j}^l$ adalah matriks *weight* atau filter yang mengoneksikan *feature maps* ke- j dari *layer l - 1* (z_j^{l-1}) dengan *feature maps* ke- i pada *layer l*. Lebih jelas ilustrasi dari *Convolutional Layer* terdapat pada Gambar 2.13.



Gambar 2.12 Contoh proses konvolusi pada citra (Yamashita dkk., 2018)

b) Non-Linearity Layer

Non-Linearity layer bertujuan untuk membuat hasil dari *Convolutional Layer* menjadi tidak linear. Prosesnya dilakukan dengan cara memasukkan setiap elemen hasil z^l dari *Convolutional Layer* ke dalam suatu fungsi f seperti pada Persamaan (2.7) (Paoletti dkk., 2018). *Layer* ini memiliki ukuran yang sama dengan *inputnya*.

$$a^l = f(z^l) \quad (2.7)$$

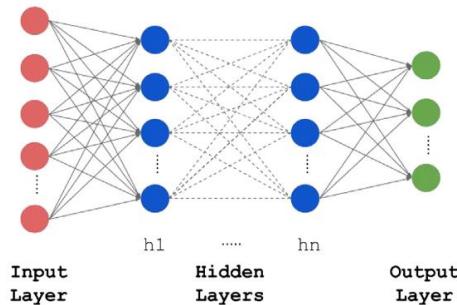
c) Pooling Layer

Pooling Layer ini digunakan untuk mengurangi fitur *invariant* berdasarkan lokasinya dengan cara menyimpulkan *output* dari beberapa neuron dari *Convolution Layer* melalui fungsi *pooling*. Salah satu jenis *layer pooling* adalah *Max Pooling*, di mana *layer* ini menyimpulkan *output* menggunakan nilai maksimum. Setiap elemen piksel a_i dari *layer l* dalam region R dipilih nilai maksimumnya seperti pada Persamaan (2.8) (Paoletti dkk., 2018).

$$p^l = \max_{i \in R} a_i^l \quad (2.8)$$

d) Fully-Connected Layer

Fully-Connected Layer adalah *layer* di mana setiap elemen yang ada pada *layer* terkoneksi dengan *layer* sebelumnya. Biasanya *layer* ini berada di *layer* terakhir pada model CNN. Pada *layer* terakhir, *layer* ini mengubah data berupa matriks n-dimensi menjadi matriks linear atau 1-dimensi, sehingga dapat dilakukan klasifikasi dengan lebih mudah seperti pada Gambar 2.14.



Gambar 2.13 Contoh *fully-connected layer* (Paoletti dkk., 2018)

2.2.4 You Only Look Once (YOLO)

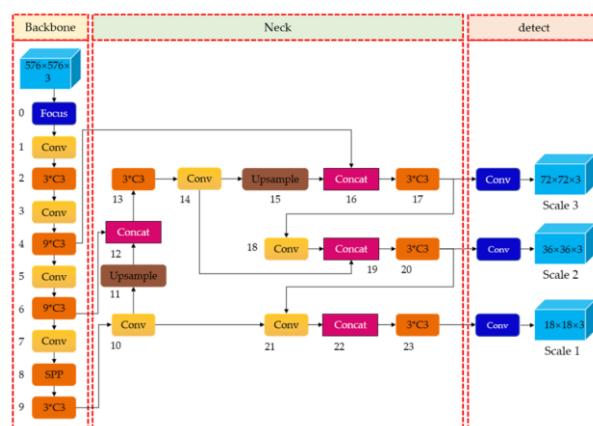
You Only Look Once (YOLO) merupakan salah satu sistem deteksi objek pada citra yang paling sesuai untuk pendekripsi objek *real-time* pada video. YOLO dipilih karena akurasi *real-time* dan kodennya yang bisa diakses publik. Teknik ini cocok digunakan untuk deteksi objek, salah satunya deteksi objek tulisan tangan (Islam & Khan, 2019).

Terdapat beberapa versi dari YOLO sejak ditemukan oleh Joseph Redmon, versi ke-1 menggunakan *framework* Darknet yang dilatih menggunakan *dataset* ImageNet-1000. Versi ini memiliki banyak batasan, seperti tidak dapat mendekripsi objek kecil dalam sebuah kluster objek, serta tidak dapat mendekripsi objek dengan dimensi yang berbeda dari data latih. Versi ke-2 dengan sebutan YOLO9000 kemudian lahir, dengan banyak perubahan, di antaranya penggunaan *batch normalization*, *anchor boxes*, resolusi *input* yang lebih tinggi, penggunaan *multi-scale training*, serta penggunaan Darknet-19 untuk mempercepat prediksi.

Kemudian versi ke-3 dari YOLO hadir dengan banyak improvisasi, seperti skor *bounding box*, prediksi multi-kelas, penggunaan *Feature Pyramid Networks* (FPN) untuk meningkatkan kemampuan ekstraksi fitur yang lebih baik (Jonnalagadda, 2019). Setelah itu, penemu YOLO Joseph Redmon untuk menghentikan penelitiannya karena dampak negatif yang muncul dari penelitiannya. Namun pengembangan YOLO tidak berhenti, Alexey Bochkovskiy melanjutkan penelitian tersebut dan muncullah YOLO versi ke-4. Versi ini menambahkan penggunaan *state of art* BoF (*bag of freebies*) dan beberapa BoS (*bag of specials*) untuk meningkatkan akurasi 10% dari versi sebelumnya.

Dalam penelitian ini, model yang digunakan adalah YOLO versi ke-5 (YOLOv5). Arsitektur yang digunakan kurang lebih sama dengan versi sebelumnya, hanya saja terdapat beberapa perubahan, seperti penggunaan PyTorch sebagai *framework* sehingga lebih mudah diakses melalui Python. Perubahan tambahan lainnya dalam YOLOv5 ini adalah terdapat *augmented data training* melalui sebuah *data loader*, spesifiknya pada *scaling*, *color space adjustments* dan *mosaic augmentation* (Schreurs, 2021).

Secara arsitektur, YOLOv5 terdiri dari empat bagian: *input*, *backbone*, *neck*, dan *output*. Bagian *input* berisi *preprocessing* dari data, termasuk *mosaic data augmentation* dan *adaptive image filling*. Untuk dapat beradaptasi dengan *dataset* yang berbeda, terdapat *adaptive anchor frame* pada *input*, sehingga dapat menyesuaikan ukuran *frame dataset*. Bagian *backbone* berisi *cross-stage partial network* (CSP) dan *spatial pyramid pooling* (SPP) untuk mengekstrak fitur dari citra *input*. Bagian *neck* berisi *BottleneckCSP* yang mengurangi jumlah kalkulasi dan meningkatkan akurasi deteksi. Terakhir, bagian *output* memprediksi target dalam ukuran yang berbeda dalam fitur map. YOLOv5 memiliki 4 macam arsitektur, YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x. Perbedaan arsitektur tersebut terdapat pada jumlah modul ekstraksi fitur dan *convolution kernel* model. Arsitektur dari YOLOv5 terdapat pada Gambar 2.15.



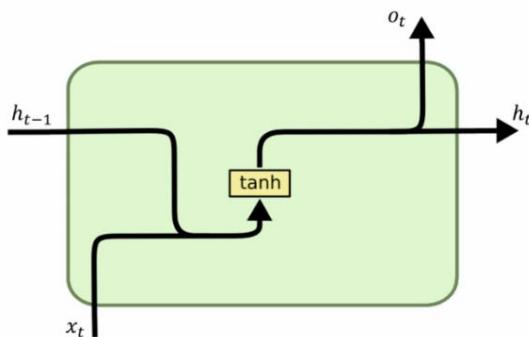
Gambar 2.14 Arsitektur YOLOv5 (Li dkk., 2022)

2.2.5 Model Neural Machine Translation

Neural Machine Translation (NMT) merupakan salah satu pendekatan dalam *machine translation* yang menggunakan *neural network* untuk memprediksi sebuah rangkaian kata, khususnya memprediksi rangkaian utuh dari sebuah kalimat dalam sebuah model. Model ini biasanya terdiri dari *encoder* dan *decoder* sebagai penerjemah dalam proses translasi, sehingga dapat digunakan untuk menerjemahkan antar bahasa ataupun transliterasi. Model ini biasanya digunakan untuk dalam masalah sekuens, seperti menerjemahkan teks ataupun memprediksi lanjutan kata.

a) Recurrent Neural Network (RNN)

Salah satu pendekatan *machine translation* yang digunakan untuk data sekuens adalah *recurrent neural network* (RNN). Setiap bagian dari rangkaian urutan diulangi dalam model untuk mendapatkan informasi pokoknya. Setiap *output* dihitung berdasarkan nilai pada bagian sebelumnya seperti pada Gambar 2.16, sehingga informasi pada rangkaian tetap terjaga.

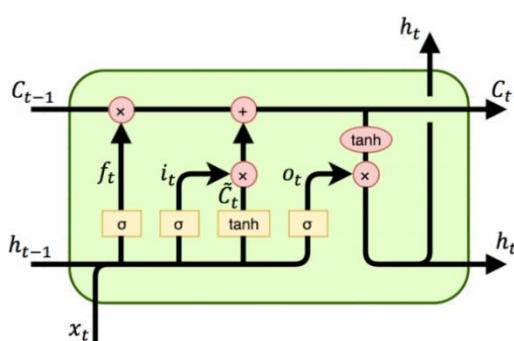


Gambar 2.15 Jaringan RNN (Dias dkk., 2022)

b) Long-Short Term Memory (LSTM)

LSTM memiliki dua properti utama, *context sensitive learning* dan *good generalization*. Kedua properti tersebut membuat LSTM berjalan berdasarkan konteks, dan tidak tergantung pada bahasa yang digunakan. Sehingga LSTM dianggap cocok untuk digunakan sebagai penerjemah bahasa maupun fungsi sekuens lainnya.

Jaringan LSTM menambahkan *multiplicative gates* dan *additive feedback* pada arsitektur RNN. Sebuah sel pada LSTM bisa menghapus informasi di dalam sel yang sudah tidak dibutuhkan, dan bisa menambahkan informasi baru untuk disimpan ke dalam sel. Sebuah sel LSTM terdiri dari beberapa *gate*. Pertama, *Forget Gate* dengan *Logistic Sigmoid Function*. *Forget Gate* akan menentukan masukan yang harus dipertahankan di setiap time step tertentu. *Forget Gate* menghasilkan nilai antara 0 (jika sel menghapus seluruh informasi di dalamnya) dan 1 (jika sel mempertahankan seluruh informasi di dalamnya). Kedua, *Input Gate* yang akan menentukan masukan yang akan diproses oleh sel. Ketiga, *Output Gate* yang akan menentukan nilai keluaran dari sel.



Gambar 2.16 Jaringan LSTM (Dias dkk., 2022)

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (2.9)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (2.10)$$

$$v_t = \tanh(W_{xv}x_t + W_{hv}h_{t-1} + b_v) \quad (2.11)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (2.12)$$

$$C_t = f_t C_{t-1} + i_t v_t \quad (2.13)$$

$$h_t = o_t \tanh(C_t) \quad (2.14)$$

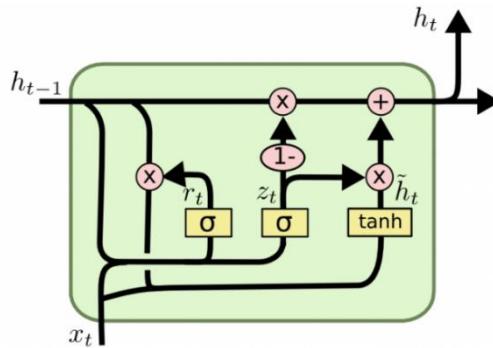
$$\sigma = \frac{1}{1 + e^{-x}} \quad (2.15)$$

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (2.16)$$

Proses *forward training* dari LSTM dinyatakan dalam (2.9)-(2.14). Parameter f_t adalah *forget gate layer*, i_t adalah *input gate layer*, o_t adalah *output gate layer*, C_t adalah *Cell State*, h_t adalah keluaran sel dan h_{t-1} adalah keluaran sel pada langkah sebelumnya. Parameter b_y dengan $y \in \{f, i, v, o\}$ adalah unit bias untuk *forget gate*, *input gate*, *input squashing* dan *output gate*, W_{xy} adalah bobot koneksi simpul eksternal ke- i dan ke- j . Misalnya, σ adalah *Logistic Sigmoid Function* (Persamaan 2.15), \tanh adalah fungsi tangen hiperbolik (Persamaan 2.16).

c) Gated Recurrent Unit (GRU)

Salah satu pendekatan *machine translation* lainnya adalah *gated recurrent unit* (GRU). Pendekatan ini merupakan variasi dari RNN yang bertujuan untuk mengatasi permasalahan kehilangan gradien pada RNN, dengan membagi *gates* ke dalam 2 bagian, *Update Gate* dan *Reset Gate*. Selain itu juga terdapat *current state* yang akan menyimpan informasi dari *input*. Pertama, *Update Gate* dihitung dengan mengalikan *input* x_t dengan bobotnya sendiri W_{xz} dan mengalikan informasi sebelumnya h_{t-1} dengan bobotnya sendiri W_{hz} beserta biasnya b_z seperti pada Persamaan (2.17). Begitu juga pada *Reset Gate*, hanya saja berbeda pada *weight* dan penggunaannya saja, seperti pada Persamaan (2.18). Begitu juga dengan *current state*, hanya saja *current state* dihitung dengan menambahkan *element-wise* pada hasil *Reset Gate* dengan informasi sebelumnya h_{t-1} beserta bobotnya W_{hh} seperti pada Persamaan (2.19). Kemudian, *output* h_t ditentukan dengan melakukan *element-wise* terhadap *current state* \bar{h}_t dengan *Update Gate* z_t dan menambahkannya dengan hasil *element-wise* informasi sebelumnya h_{t-1} dengan negatif dari *Update Gate* $1 - z_t$ seperti pada Persamaan (2.20).



Gambar 2.17 Jaringan GRU (Abdulwahab dkk., 2018)

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \quad (2.17)$$

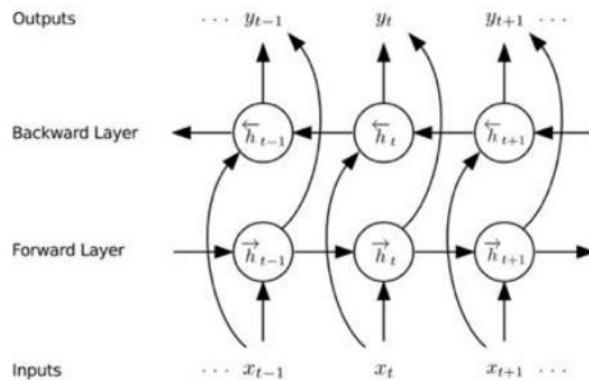
$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \quad (2.18)$$

$$\bar{h}_t = \tanh(W_{xh}x_t + r_t \odot W_{hh}h_{t-1} + b_h) \quad (2.19)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \bar{h}_t \quad (2.20)$$

d) Bidirectional Model

Mekanisme *bidirectional* merupakan pendekatan yang dilakukan pada model translasi untuk dapat mempelajari informasi masa lalu dan informasi masa mendatang untuk setiap urutan *input*. Pada mekanisme ini terdapat dua jaringan model, jaringan pertama berfungsi untuk memproses urutan masukan data ke arah depan (*forward*) dan jaringan kedua berfungsi untuk memproses urutan masukan data ke arah sebaliknya (*backward*) (Puteri, 2023). Kemudian *output* tersebut dari jaringan *forward* dan *backward* digabungkan pada setiap urutan waktu seperti pada Gambar 2.19.



Gambar 2.18 Arsitektur *Bidirectional Model* (Pasaribu dkk., 2023)

Dari Gambar 2.19, didapatkan bahwa setiap *hidden unit* keluaran pada lapisan *forward* dan *backward* digabungkan dan membentuk nilai fitur kata tersebut dengan ukuran lebih panjang daripada menggunakan model biasa. Karena menghasilkan nilai fitur yang lebih panjang, maka informasi yang diproses pada tahap selanjutnya akan mengklasifikasikan dengan lebih akurat.

2.2.6 Evaluasi Model Deteksi

Evaluasi pada model deteksi dengan banyak jenis kelas biasanya dilakukan dengan menggunakan metrik *mean average precision* (*mAP*) pada setiap kelasnya. Nilai *mAP* didapatkan dari metrik lainnya, yaitu *Confusion Matrix*, *Intersection over Union* (*IoU*), *Precision*, dan *Recall*. Selain itu juga terdapat metrik *F1 Score* sebagai validasi dari seluruh metrik dan menentukan model yang optimal pada kelas dengan persebaran yang tidak merata.

a) Mean Average Precision (*mAP*)

Mean Average Precision adalah ukuran atau metrik untuk mengevaluasi model deteksi objek seperti *Fast R-CNN*, *YOLO*, *Mask R-CNN*, dan lain-lainnya. Nilai dari *mAP* dihitung berdasarkan nilai *recall* dari 0 hingga 1. Formula *mAP* berdasarkan sub-metrik *Confusion Matrix*, *Intersection over Union* (*IoU*), *Recall*, dan *Precision*.

Nilai *mAP* merupakan rata-rata AP dari seluruh kategori, di mana AP (*Average Precision*) merupakan area di bawah kurva *Precision-Recall* (P-R). Sehingga, nilai AP biasanya dihitung menggunakan rata-rata dari nilai *recall* dengan jarak yang sama, misalkan $Recall_i = [0, 0.1, 0.2, \dots, 1.0]$, maka *mAP* dapat dihitung menggunakan Persamaan (2.21). Selain persamaan tersebut, nilai *mAP* juga bisa dinyatakan dalam persamaan umum pada Persamaan (2.22) yang dapat digunakan untuk kondisi rata-rata nilai AP yang tidak ditentukan intervalnya.

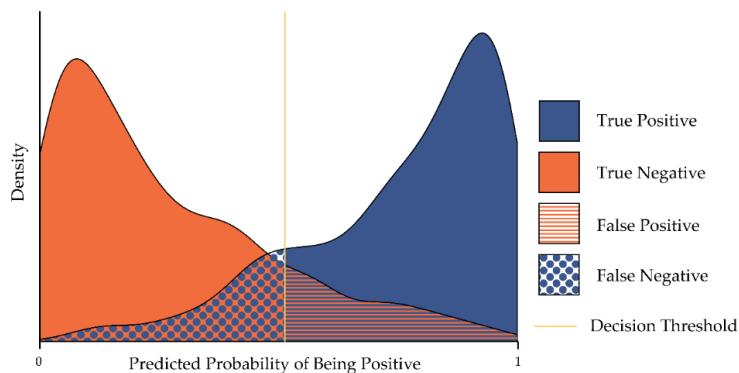
$$mAP = \frac{1}{11} \sum_{Recall_i} Precision at Recall_i \quad (2.21)$$

$$mAP = \frac{\sum_{x=1}^X AP(x)}{X} \quad (2.22)$$

b) *Confusion Matrix*

Confusion Matrix adalah pengukuran performa untuk klasifikasi *machine learning* di mana keluarannya dapat berupa dua kelas atau lebih. Dalam *confusion matrix* seperti pada Gambar 2.20 terdapat atribut-atribut berikut :

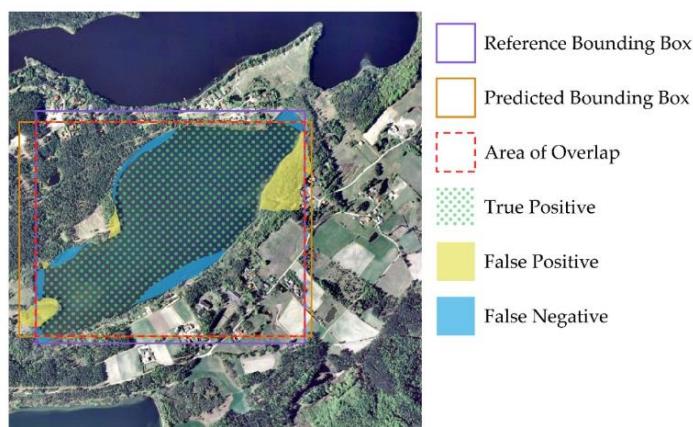
- *True Positives (TP)* : model memprediksi label dan sesuai dengan *ground truth*
- *True Negatives (TN)* : model tidak memprediksi label dan tidak menjadi bagian dari *ground truth*
- *False Positives (FP)* : model memprediksi label tetapi tidak menjadi bagian dari *ground truth (Type I Error)*
- *False Negatives (FN)* : model tidak memprediksi label tetapi menjadi bagian dari *ground truth (Type II Error)*



Gambar 2.19 Ilustrasi probabilitas pada *confusion matrix* (Maxwell dkk., 2021)

c) *Intersection over Union (IoU)*

Intersection over Union menunjukkan area irisan antara koordinat *prediction box* dengan *ground truth box*. Semakin tinggi *IoU* menunjukkan *bounding box* hasil prediksi hampir sama dengan *ground truth box*. Ilustrasi dari *Intersection over Union* bisa dilihat pada Gambar 2.21.



Gambar 2.20 Ilustrasi *Intersection over Union* (Maxwell dkk., 2021)

d) *Precision*

Precision mengukur seberapa bagus model dapat mengukur *True Positives (TP)* dari jumlah seluruh *Positive Predictions (TP+FP)*. Sebagai contoh, *precision* dihitung dengan menggunakan *IoU threshold* dalam deteksi objek. Jika *IoU* threshold bernilai 0.5, dan hasil *IoU* deteksi citra bernilai 0.3 *IoU*, maka deteksi tersebut digolongkan ke dalam *False Positive*. Sedangkan jika deteksi *IoU* deteksi citra bernilai 0.7 *IoU*, maka deteksi tersebut digolongkan ke dalam *True Positive (TP)*. Rumus untuk *Precision* dapat dilihat pada Persamaan (2.23).

$$Precision = \frac{TP}{TP + FP} \quad (2.23)$$

e) *Recall*

Berbeda dengan *precision*, nilai *recall* digunakan untuk mengukur seberapa bagus model dapat mengukur *True Positives* (TP) dari jumlah seluruh prediksi (TP+FN). Sehingga secara tidak langsung menunjukkan bahwa nilai *precision* cenderung mengukur kualitas data, sedangkan *recall* cenderung mengukur kuantitas data. Persamaan yang digunakan untuk menghitung *recall* dapat dilihat pada Persamaan (2.24).

$$Recall = \frac{TP}{TP + FN} \quad (2.24)$$

f) *F1 Score*

Nilai *F1 score* menjadi metrik utama pada model deteksi, karena metrik ini menggunakan *precision* dan *recall* untuk mengetahui objektivitas hasil deteksi. Nilai tersebut tidak mengabaikan kualitas data dengan menggunakan nilai *precision* serta tidak mengabaikan kuantitas data dengan menggunakan nilai *recall*. *F1 Score* dihitung dengan Persamaan (2.25).

$$F1 score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (2.25)$$

2.2.7 Evaluasi Model Transliterasi

Word Error Rate (WER) digunakan untuk mengevaluasi performansi mesin transliterasi citra segmen kata. WER menunjukkan matriks jarak penyuntingan antara teks target dan teks transliterasi yang dihasilkan. Jarak didefinisikan sebagai rasio aksi penyisipan, penghapusan, dan penggantian huruf dibandingkan dengan total panjang kata (Kesiman & Dermawan, 2021). Nilai WER bisa diekspresikan seperti pada Persamaan (2.26).

$$WER = \frac{S + D + I}{N} \quad (2.26)$$

S = Jumlah substitusi kata

D = Jumlah penghapusan kata (*deletion*)

I = Jumlah penyisipan kata (*insertion*)

N = Total jumlah kata

Selain itu juga terdapat metrik lainnya yang digunakan untuk melakukan pengujian, yaitu *Character Error Rate* (CER). Hampir sama dengan WER, hanya saja CER menghitung rasio perbedaan dalam sebuah kalimat berdasarkan karakternya. CER bisa diekspresikan pada Persamaan (2.27).

$$CER = \frac{S + D + I}{N} \quad (2.27)$$

S = Jumlah substitusi karakter

D = Jumlah penghapusan karakter (*deletion*)

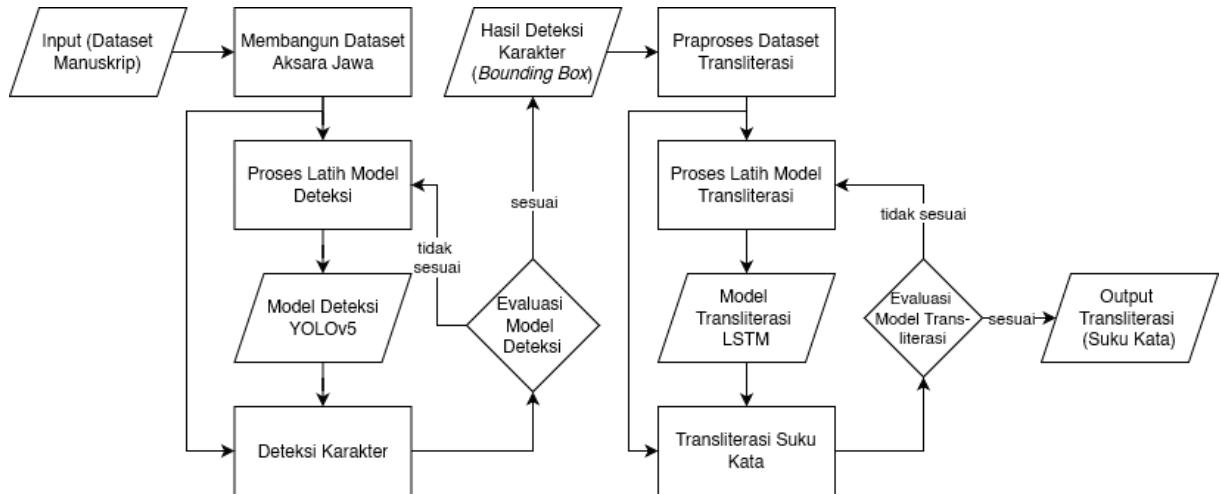
I = Jumlah penyisipan karakter (*insertion*)

N = Total jumlah karakter

BAB III METODOLOGI DAN IMPLEMENTASI

3.1 Perancangan Sistem

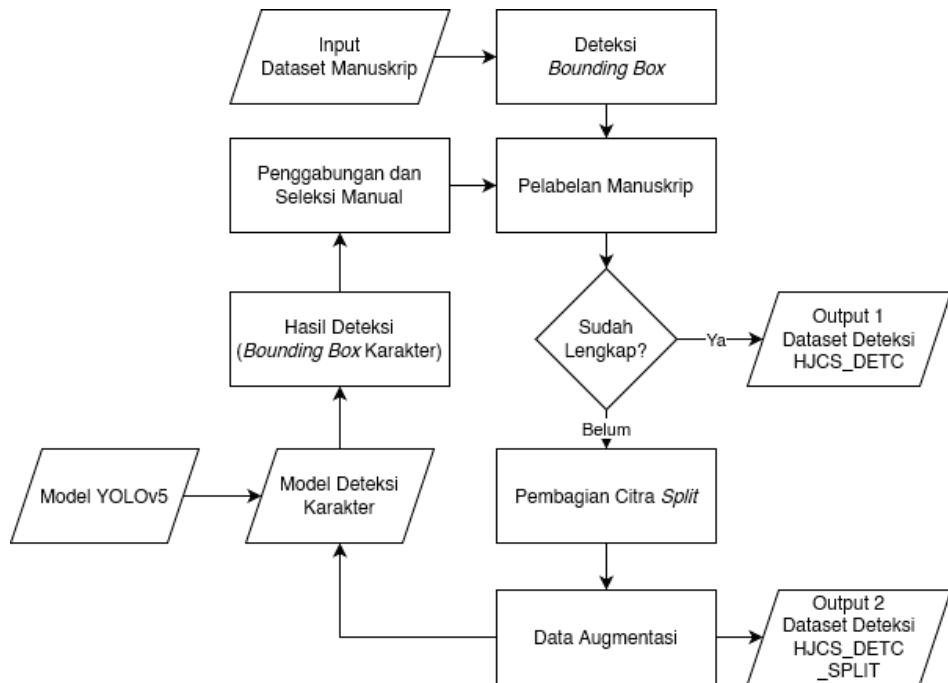
Sistem deteksi transliterasi yang akan digunakan dalam penelitian ini terbagi menjadi 2, yaitu tahapan deteksi karakter dan transliterasi karakter. *Input* sistem berbentuk citra manuskrip, kemudian sistem akan mendeteksi karakter dalam citra menggunakan model YOLOv5 yang telah dilatih menggunakan *dataset*. Kemudian hasil deteksi diolah kembali sebagai *input* dari model transliterasi LSTM untuk diproses. *Output* dari sistem ini berupa rangkaian suku kata dari sebuah kalimat.



Gambar 3.1 Rancangan sistem deteksi transliterasi

3.1.1 Membangun *Dataset* Aksara Jawa

Dataset manuskrip aksara Jawa merupakan sumber utama dalam penelitian ini. *Dataset* inilah yang akan digunakan pada tahapan deteksi karakter serta transliterasi. Untuk membangun *dataset*, diperlukan rangkaian pemrosesan seperti pada Gambar 3.2.

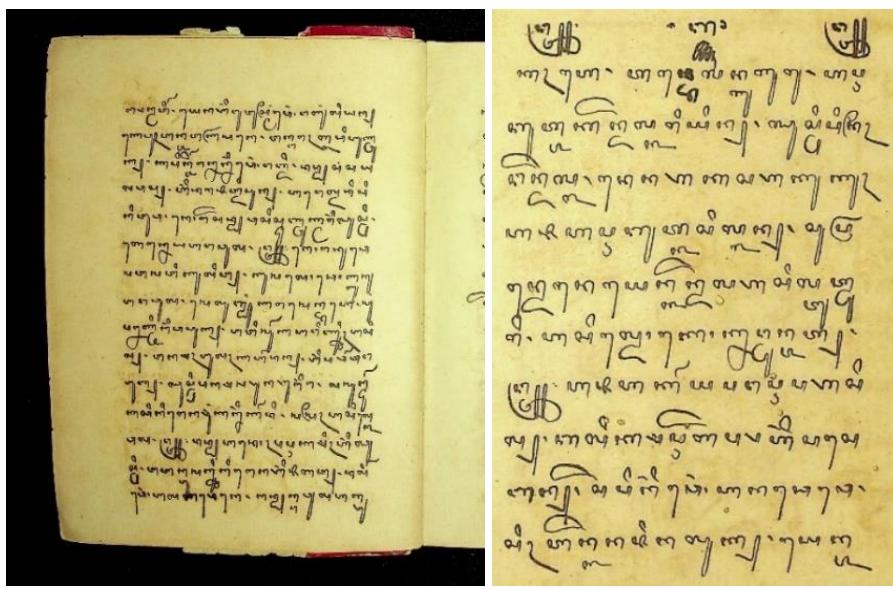


Gambar 3.2 Alur membangun *dataset* aksara Jawa

Dari pemrosesan tersebut, dihasilkan dua jenis *dataset* yakni *Handwritten Javanese Character on Sewaka Manuscripts Detection* (HJCS_DETC) dan HJCS_DETC_SPLIT. Masing-masing *dataset* digunakan untuk tujuan yang berbeda, berikut penjelasannya :

a. *Dataset HJCS_DETC*

Dataset HJCS_DETC ini merupakan *dataset* dasar dalam penelitian ini, terdiri dari 60 citra manuskrip dari Serat Sewaka yang berasal dari website KHASTARA Perpusnas RI (khastara.perpusnas.go.id). Data tersebut kemudian di potong pada bagian manuskripnya saja untuk mendapatkan karakter aksara. Citra yang telah dipotong dapat mengurangi kompleksitas latih deteksi seiring berkurangnya piksel yang dilatih serta menghindari model deteksi mendeteksi karakter di luar dari daerah manuskrip.



(a)

(b)

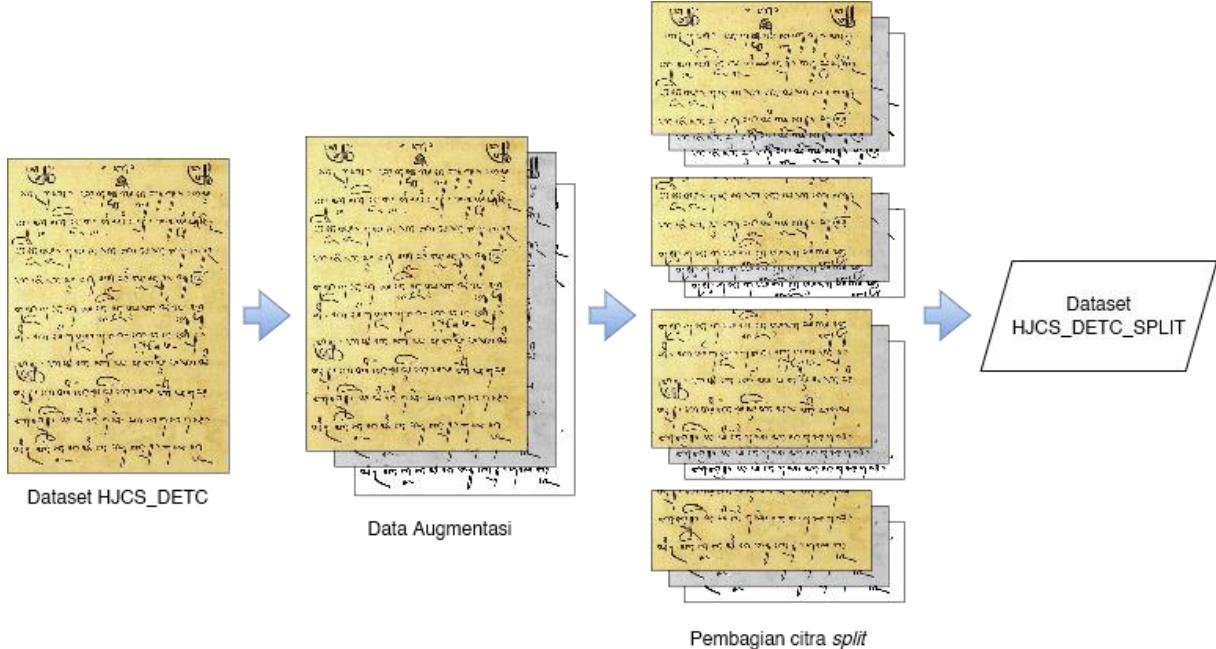
Gambar 3.3 (a) Citra asli manuskrip Serat Sewaka (b) Manuskrip yang sudah dipotong

Dataset ini terdiri dari citra dan anotasi karakter dari 74 kelas karakter aksara Jawa, yang terdiri dari aksara *wianjana*, sampai dengan aksara *rekan* dengan persebaran jumlah yang tidak merata. Proses pelabelan atau pemberian anotasi pada *dataset* ini melalui proses pelabelan semi-otomatis dengan cara bekerja sama dengan mahasiswa dari Universitas Negeri Surabaya jurusan Sastra Jawa. Pelabelan semi-otomatis dilakukan untuk mempercepat pengumpulan *dataset*. Pelabelan dilakukan dengan cara melakukan komputasi pada citra yaitu dengan deteksi *bounding box* pada karakter menggunakan *image processing*, serta penggunaan model deteksi YOLO. Pada penggunaan model deteksi, *threshold* yang digunakan adalah 0.8 untuk mendeteksi data yang masih belum dilakukan pelabelan. Kekurangan dari *dataset* ini adalah memiliki ukuran citra yang tergolong besar untuk proses latih, sehingga *dataset* ini tidak akan digunakan secara langsung untuk penelitian. Namun, *dataset* ini masih bisa digunakan untuk penelitian selanjutnya.

b. *Dataset HJCS_DETC_SPLIT*

Dataset ini merupakan *dataset* manuskrip yang telah di proses untuk dapat digunakan pada penelitian ini. Permasalahan yang dimiliki pada karakter aksara Jawa adalah karakter yang saling tumpang tindih dan jumlah citra pada masing-masing kelas yang tidak merata. Oleh karena itu, dibutuhkan *dataset* aksara Jawa yang siap digunakan untuk menunjang proses deteksi karakter aksara Jawa. Solusi yang ditawarkan pada penelitian ini adalah dengan menggunakan proses data augmentasi pada *dataset* untuk meningkatkan variasi citra.

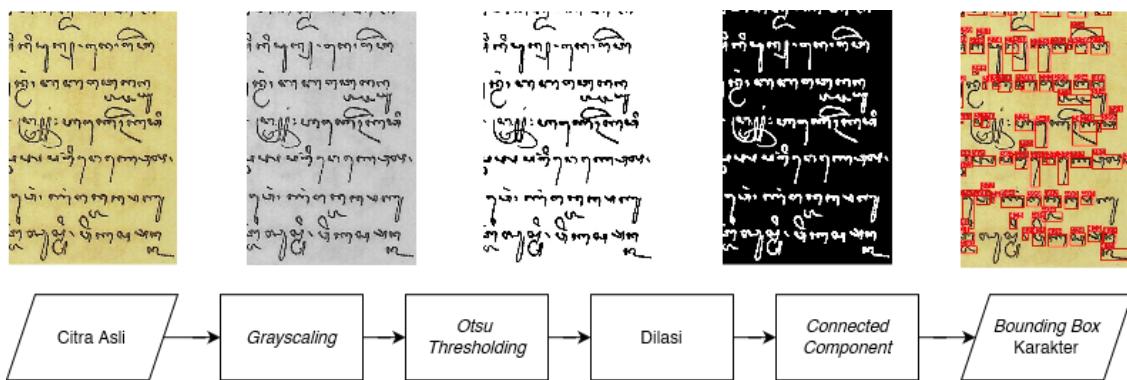
Permasalahan lainnya adalah ukuran citra pada *dataset* HJCS_DETC yang besar. Sehingga perlu diselesaikan dengan menggunakan pembagian citra ke dalam beberapa citra *split* agar proses latih deteksi dapat dilakukan pada penelitian ini dengan lebih efektif. Pemrosesan *dataset* HJCS_DETC_SPLIT dilakukan seperti pada Gambar 3.4. Proses data augmentasi dijelaskan lebih lanjut pada subbab 3.1.1.3 dan proses pembagian citra *split* dijelaskan lebih lanjut pada subbab 3.1.1.4.



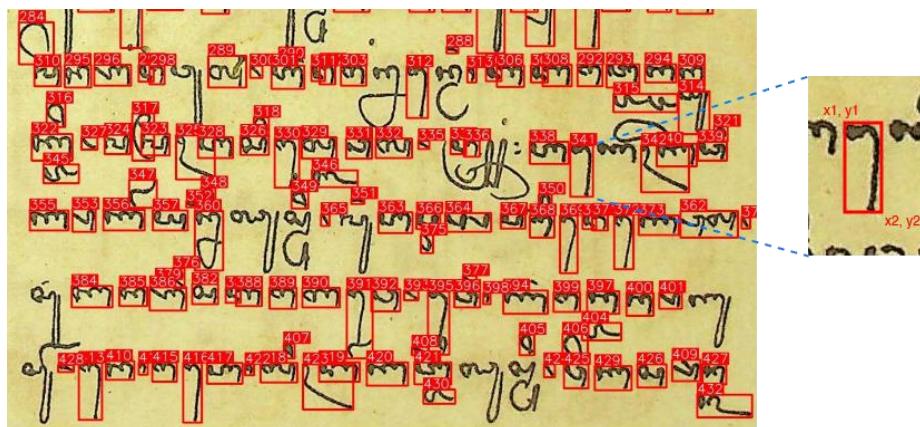
Gambar 3.4 Pemrosesan pada *dataset* HJCS_DETC_SPLIT

a) Deteksi *Bounding Box*

Untuk mempercepat proses pelabelan, deteksi *bounding box* perlu dilakukan. Deteksi otomatis *bounding box* dilakukan karena menghasilkan *bounding box* yang lebih presisi dibandingkan dengan proses manual, dan lebih cepat dalam membantu pelabelan. *Bounding box* diperoleh dengan melakukan serangkaian *image processing* pada citra, yakni *grayscale*, *otsu thresholding*, dan dilasi seperti pada Gambar 3.5. Kemudian menggunakan *connected component*, *bounding box* didapatkan dengan mendapatkan posisi paling kiri-atas dan paling kanan-bawah dari *component* seperti pada Gambar 3.6. Implementasi dari proses deteksi *bounding box* dapat dilihat pada subbab 3.3.1.1.



Gambar 3.5 Proses deteksi *bounding box*



Gambar 3.6 Ilustrasi hasil deteksi *bounding box* karakter

b) Pelabelan Manuskrip

Tahap selanjutnya adalah tahap pelabelan data menggunakan situs *online* roboflow.com seperti pada Gambar 3.7. Kelebihan menggunakan situs *online* tersebut adalah proses pelabelan dapat dilakukan secara bersama-sama, tanpa harus bergantian. Sehingga, proses pelabelan bisa dilakukan oleh lebih dari 1 orang. Salah satu tantangan dari proses ini adalah manuskrip tersebut memiliki densitas karakter yang cukup tinggi di tiap halamannya, sebagai contoh sebuah halaman bisa mencapai lebih dari 400 karakter. Sehingga membutuhkan waktu yang cukup lama untuk melakukan pelabelan. Tantangan lainnya adalah terdapat beberapa halaman yang terdampak atau rusak sehingga karakter menjadi cukup buram untuk dikenali.



Gambar 3.7 Proses pelabelan pada situs *online* roboflow.com

Pelabelan dilakukan terhadap 60 citra, dengan hasil berbentuk informasi posisi *bounding box* dan klasifikasi kelas dari setiap karakter atau disebut dengan anotasi. Setiap anotasi berisi kelas objek, koordinat, lebar, dan panjang dari karakter. Contoh ilustrasi dari anotasi pada sebuah citra pada Gambar 3.8.

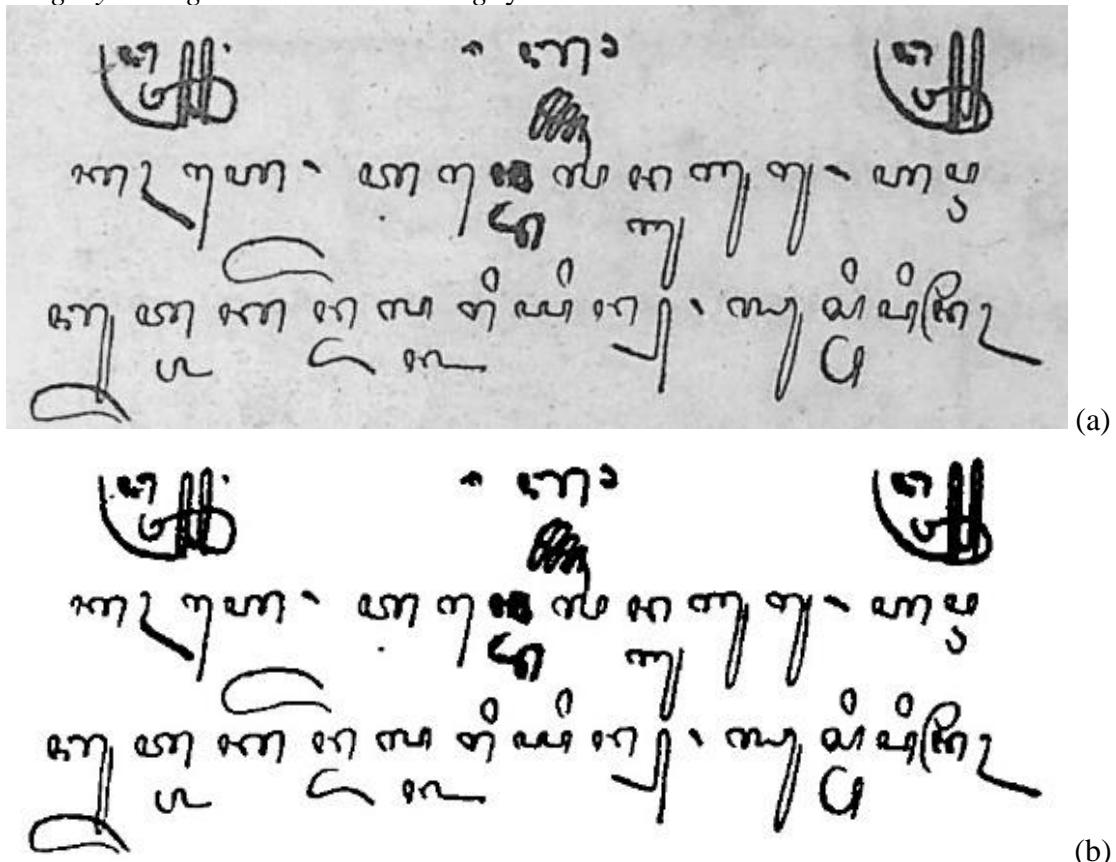
The image shows a manuscript page with characters labeled with their names in Indonesian. Labels include "Poda Modya", "Pura Poda", "Saka", "Poda Lingga", "Murha Ga", "Posongan Ga", "Posongan Ko", "Posongan Li", "Posongan Mu", "Posongan Si", "Posongan Ti", "Posongan Di", and "Posongan Lo". Each label corresponds to a specific character or group of characters on the page.

Gambar 3.8 Hasil ilustrasi anotasi pada citra

20

c) Data Augmentasi

Selain permasalahan densitas karakter yang tinggi, juga diperlukan variasi *dataset* untuk meningkatkan performa model deteksi. Pada penelitian ini, data augmentasi digunakan sebagai variasi *dataset*. Metode data augmentasi yang digunakan adalah *grayscale* dan *otsu thresholding*. Metode ini menghasilkan varian data baru dengan kualitas citra yang lebih baik. Sebanyak 60 citra dari *dataset* HJCS_DETC selanjutnya diproses dengan data augmentasi *grayscale* dan *otsu thresholding* seperti pada Gambar 3.9. Sehingga terdapat 3 variasi, yakni 60 *dataset* original saja, 120 *dataset* original beserta hasil *grayscale*nya, dan 180 *dataset* beserta *grayscale* dan *otsu thresholding*nya.

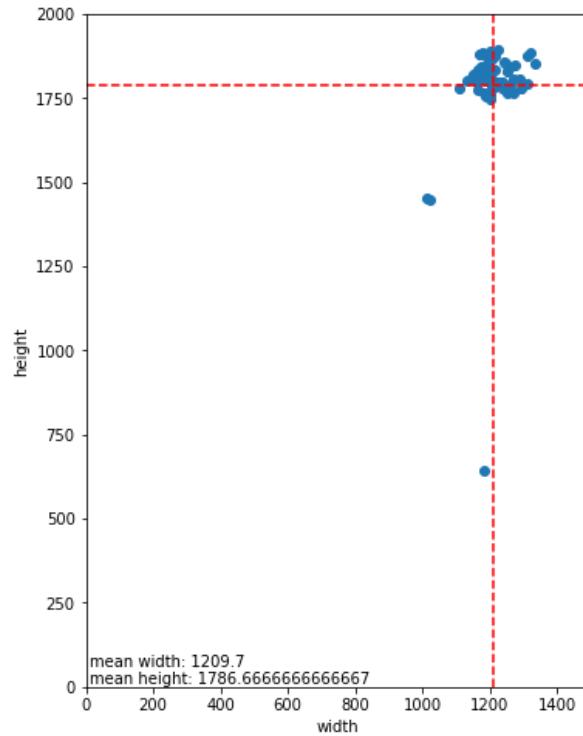


Gambar 3.9 (a) *Grayscale* pada citra (b) *Otsu thresholding* pada citra

d) Pembagian Citra *Split*

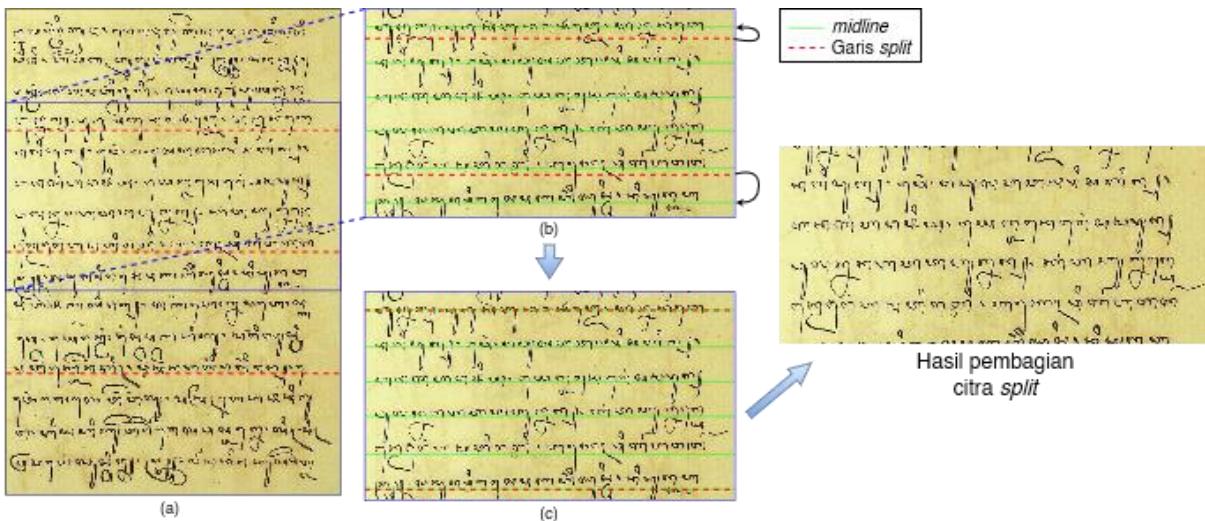
Permasalahan keterbatasan sumber daya latih deteksi, utamanya pada resolusi citra yang cukup besar, membuat *dataset* perlu untuk dibagi ke dalam beberapa citra *split*. Namun, yang menjadi permasalahan lagi adalah bagaimana melakukan pembagian citra *split* tanpa menghilangkan informasi pada citra. Karena jika melakukan *split* secara langsung, maka garis *split* bisa jadi berada di tengah baris karakter, sehingga perlu dilakukan mekanisme penentuan garis *split* agar tidak ada informasi yang hilang.

Dataset HJCS_DETC memiliki ukuran resolusi yang cukup tinggi, bisa dilihat pada Gambar 3.10 menunjukkan bahwa resolusi rata-rata sekitar 1200 x 1780, sedangkan rasio piksel rata-rata *input* yang dapat digunakan pada YOLOv5 adalah 640 piksel. Selain itu, penggunaan ukuran citra yang lebih kecil juga memudahkan dalam melakukan latih model pada kasus penelitian ini. Sehingga, diperlukan skenario pembagian citra menjadi citra dengan resolusi yang lebih kecil, sekaligus tidak menghilangkan informasi yang ada pada citra.



Gambar 3.10 Rata-rata ukuran citra pada dataset

Sebelum melakukan pembagian citra, perlu menentukan *midline* terlebih dahulu. *Midline* ditentukan berdasarkan *horizontal projection* dari anotasi yang ada. Kemudian, pembagian citra dilakukan berdasarkan *midline* pada setiap baris. Apabila garis *split* memotong baris anotasi, maka garis akan digeser ke arah *midline* terdekat di atas atau di bawahnya agar anotasi tidak hilang dari citra *split*. Implementasi dari pembagian citra *split* ini terdapat pada subbab 3.3.1.2.

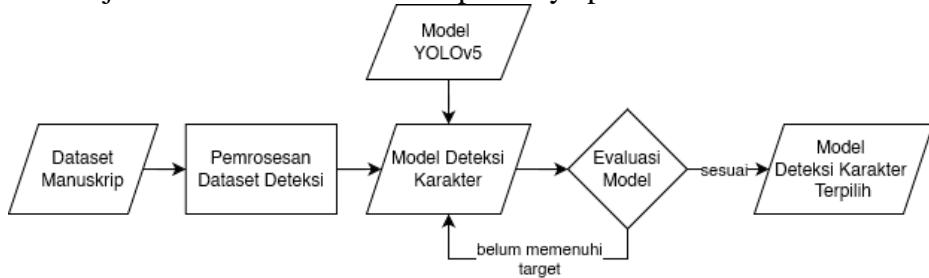


Gambar 3.11 Ilustrasi (a) citra dengan garis *split* (b) posisi garis *split* dan *midline* sebelum di geser (c) posisi garis *split* setelah di geser ke *midline* terdekat

3.1.2 Proses Latih Model Deteksi

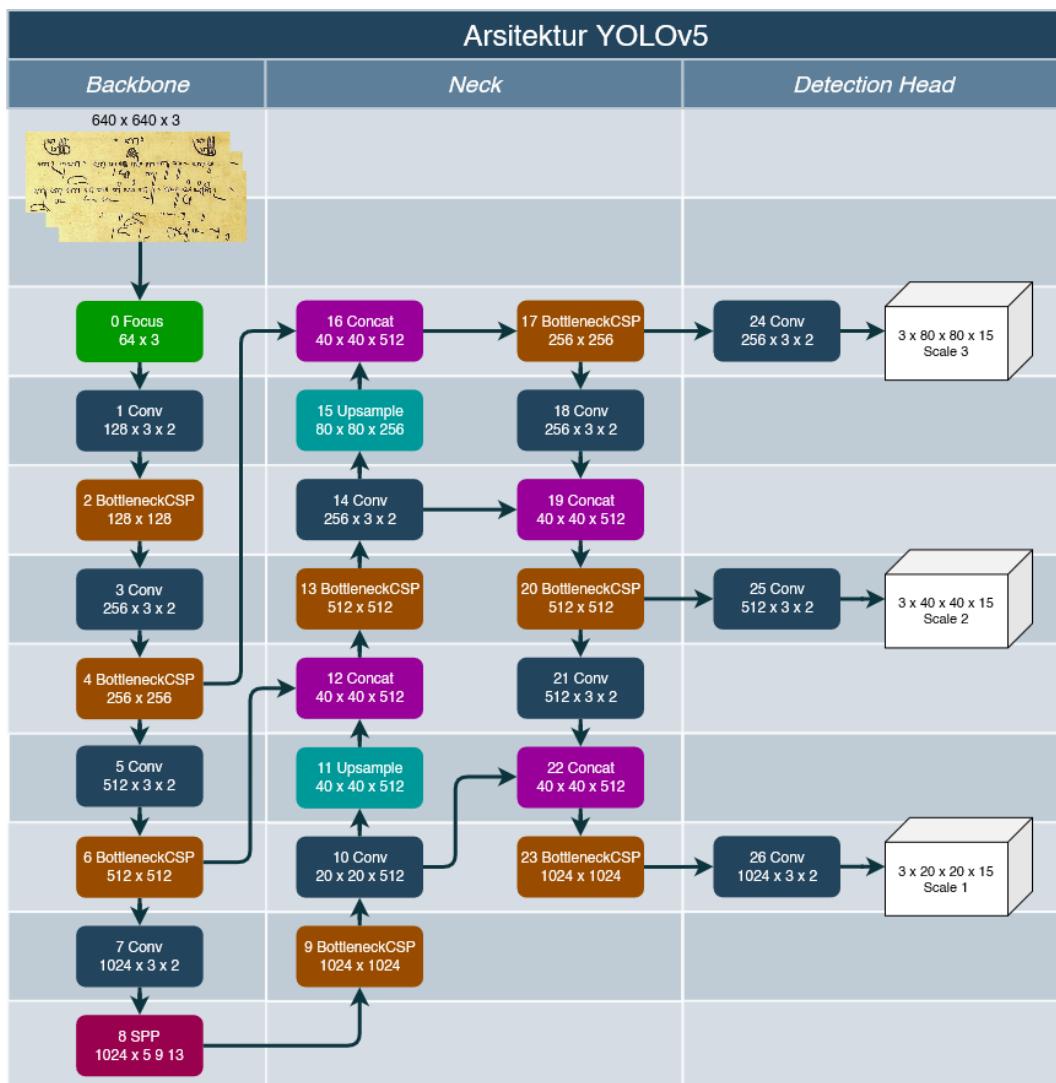
Model deteksi karakter berasal dari *pre-trained* model YOLOv5 karena model ini memiliki akurasi *real-time* dan kodennya yang bisa diakses secara publik. Teknik ini cocok digunakan untuk deteksi objek, salah satunya deteksi objek tulisan tangan. Model YOLOv5 bisa diakses secara publik melalui Github pada laman <https://github.com/ultralytics/yolov5>. Model dilatih seperti pada Gambar 3.12 untuk kemudian model deteksi karakter terbaik akan

disimpan dan digunakan untuk mendeteksi citra target. Dari model ini dihasilkan *bounding box* karakter berisikan jenis kelas karakter beserta posisinya pada citra.



Gambar 3.12 Rancangan proses latih model deteksi

Dalam melakukan proses latih model YOLOv5, pada penelitian ini terlebih dahulu diatur beberapa parameternya untuk mendapatkan proses latih serta hasil yang maksimal. Perlu penyesuaian untuk melakukan proses latih menggunakan GPU, agar proses latih lebih cepat. Arsitektur dari YOLOv5 yang digunakan pada penelitian ini dapat dilihat pada Gambar 3.13.

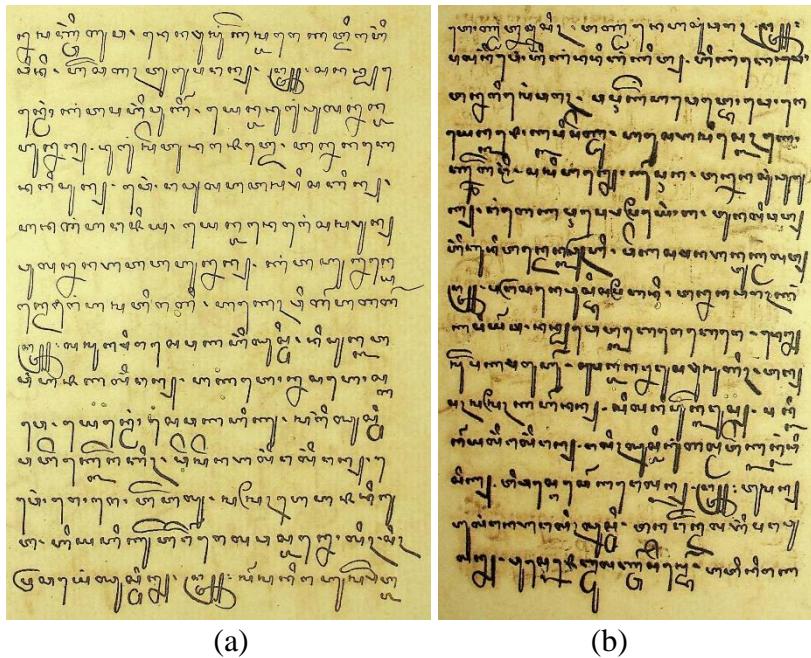


Gambar 3.13 Arsitektur Model Deteksi

Pada penelitian ini proses latih menggunakan jumlah standar *epoch* sebanyak 10.000 *epoch*. Jumlah *epoch* didapatkan berdasarkan dengan jumlah kelas yang ada sesuai dengan standar konfigurasi YOLOv5. Model dilatih untuk mendeteksi ke 67 kelas Aksara Jawa.

3.1.3 Evaluasi Model Deteksi

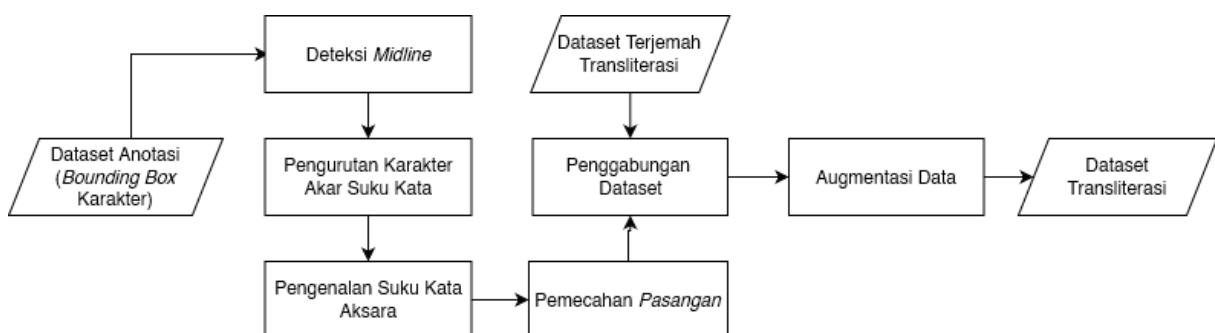
Model yang telah dilatih kemudian dievaluasi menggunakan *mean average precision* (*mAP*), *confusion matrix*, presisi, *recall*, dan *F1 Score* dari model selama proses latih. Selain itu, nilai tersebut juga dievaluasi pada masing-masing kelas untuk mengetahui kelas yang belum terdeteksi dengan sempurna. Pada evaluasi ini, kondisi citra yang akan dideteksi terbagi menjadi 2, kondisi cukup baik dan kondisi sedikit rusak. Kondisi citra dimungkinkan terjadi akibat penggunaan jenis tinta yang kurang baik pada manuskrip sehingga menyebabkan tulisan menjadi buram.



Gambar 3.14 Contoh citra yang akan dideteksi (a) kondisi baik (b) kondisi sedikit rusak

3.1.4 Praproses Dataset Transliterasi

Selain itu juga ada proses pengumpulan *dataset* transliterasi berbentuk urutan suku kata, di mana *dataset* ini bersumber pada *dataset* HJCS_DETC. Tantangan yang didapatkan juga serupa, di mana *dataset* suku kata berjumlah sangat sedikit jika dibandingkan dengan kebutuhan model LSTM yang seharusnya membutuhkan data yang cukup banyak untuk mengenali suku kata. Proses pengumpulan dilakukan dengan melakukan deteksi *midline* untuk setiap baris pada setiap citra. Kemudian dilanjutkan pada masing-masing baris dilakukan deteksi karakter aksara akar suku kata, seperti *wianjana*, *pasangan*, *murda*, serta *pasangan murda*. Kemudian dilanjutkan dengan mengidentifikasi *sandhangan*, serta karakter lainnya untuk melengkapi sebuah suku kata. Kemudian *dataset* digabungkan dengan data terjemah transliterasi yang telah dibuat oleh ahli agar *dataset* dapat digunakan sebagai *input* dari model LSTM.

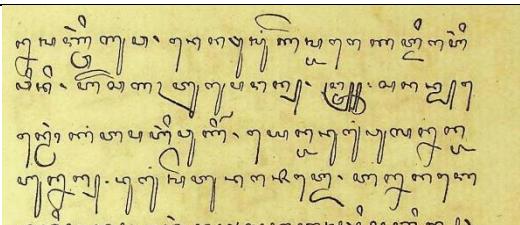
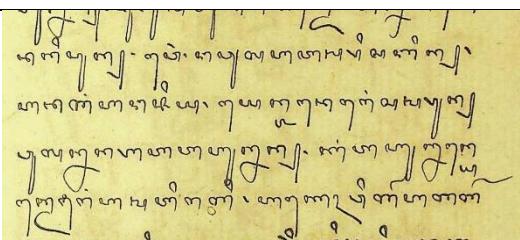


Gambar 3.15 Alur praproses *dataset* transliterasi

a) *Dataset* Terjemah Transliterasi

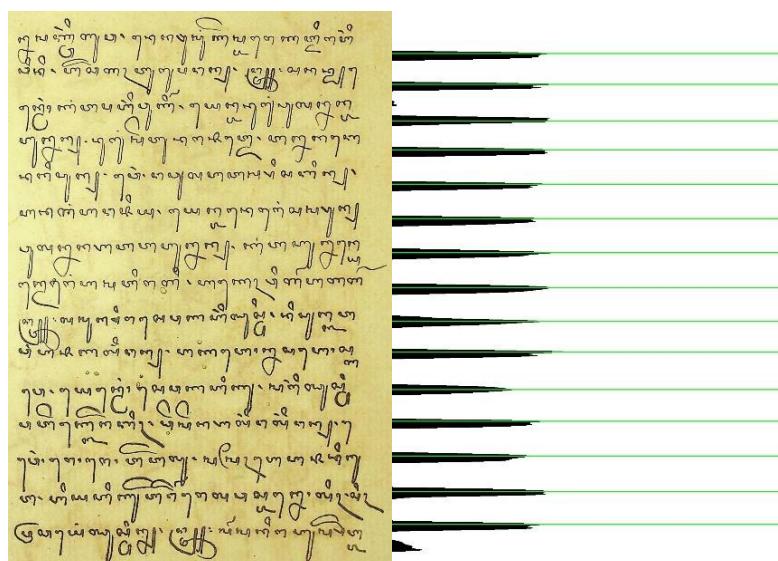
Dataset transliterasi didapatkan melalui proses pengumpulan anotasi pada citra Serat Sewaka, atau pada penelitian ini menggunakan *dataset* HJCS_DETC. *Dataset* didapatkan melalui pengurutan karakter-karakter aksara Jawa sebagai *input*, dan urutan suku kata pada baris yang sama sebagai *output*. *Dataset output* didapatkan melalui pengenalan secara manual oleh ahli. Beberapa contoh data terjemah transliterasi terdapat pada Tabel 3.1.

Tabel 3.1 Contoh *dataset* transliterasi

Citra	Terjemah Transliterasi
 orig_13_Im1.jpg	n na ma n jing gu wa / dè n su mung kē m ta ré ka t di ra ing wi di / a cê gah tu ru pa nga n / # sa na dya n wong kang ta pa ing wu kir / yè n ta du rung wu la n na n ta u n na n / du rung mê tu da ra ja t dé / ta n na na bé
 orig_13_Im2.jpg	da ni pu n / wong nga wu la a ta m pi sa bi n / a da gang a nga ji ya / yè n ta dè rèng sa m pu n wu la n na n a ta ta u n na n / kang ta u n na n yè n dè rèng a ma ti ra gi / a kèh wi gar a ga gar

b) Deteksi *Midline*

Untuk mengurutkan anotasi, diperlukan deteksi *midline* baris terlebih dahulu untuk menentukan posisi serta urutan karakter pada citra. Deteksi dilakukan dengan mencari nilai ekstrem pada proyeksi horizontal terhadap *bounding box* karakter-karakter akar suku kata (*wianjana*, *murda*, *tandha*), seperti pada Gambar 3.16 Karakter tersebut digunakan karena menjadi dasar terbentuknya suku kata, sehingga deteksi baris dapat berfokus pada pencarian akar suku kata.



(a)

(b)

Gambar 3.16 (a) citra (b) *horizontal projection* akar suku kata beserta *midline*

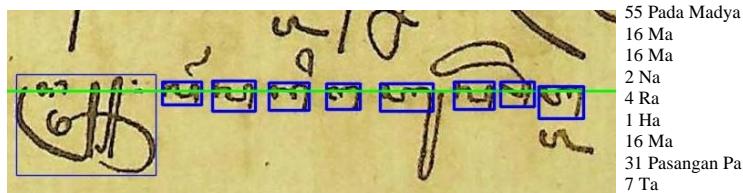
c) Pengurutan Karakter

Kemudian, tahapan selanjutnya adalah pengurutan karakter akar suku kata, berdasarkan urutan dari kiri ke kanan, serta berdasarkan perpotongan karakter tersebut dengan *midline*. Sehingga akar suku kata dapat diurutkan sesuai dengan barisnya. Klasifikasi karakter akar suku kata didasarkan pada posisi aksara, serta berdasarkan klasifikasinya pada aksara Jawa untuk memudahkan pengenalan suku kata pada proses selanjutnya seperti pada Tabel 3.2.

Tabel 3.2 Klasifikasi kelas karakter berdasarkan posisi

Klasifikasi	Sub klasifikasi	Kelas karakter	Klasifikasi	Sub klasifikasi	Kelas karakter
wianjana	base	1 Ha	<i>pasangan</i>	desc	38 Pasangan Ba
		2 Na			39 Pasangan Tha
		3 Ca			40 Pasangan Nga
		4 Ra		base-desc	35 Pasangan Nya
		5 Ka	<i>tandha</i>	base	55 Pada Lingsa
		6 Da			56 Pada Madya
		7 Ta			57 Purwa Pada
		8 Sa	<i>murda</i>	base	58 Murda Na
		9 Wa			59 Murda Ka
		10 La			60 Murda Ta
		11 Pa			61 Murda Sa
		12 Dha			62 Murda Pa
		13 Ja			Murda Nya
		14 Ya			63 Murda Ga
		15 Nya			64 Murda Ba
		16 Ma	<i>pasangan murda</i>	base	Pasangan Murda Pa
		17 Ga			Pasangan Murda Na
		18 Ba			Pasangan Murda Ka
		19 Tha			66 Pasangan Murda Ta
		20 Nga			Pasangan Murda Sa
pasangan	base	21 Pasangan Ha	<i>sandhangan</i>	base	Pasangan Murda Nya
		28 Pasangan Sa			65 Pasangan Murda Ga
		31 Pasangan Pa			Pasangan Murda Ba
		22 Pasangan Na		base	53 Pa Cerek
		23 Pasangan Ca			54 Nga Lelet
pasangan	desc	24 Pasangan Ra		pre	44 Taling
		25 Pasangan Ka		pre-asc	42 Pepet
		26 Pasangan Da		asc	41 Wulu
		27 Pasangan Ta			46 Cecak
		29 Pasangan Wa		post-asc	47 Layar
		30 Pasangan La		post-con	43 Suku
		31 Pasangan Pa			48 Pangkon
		32 Pasangan Dha			49 Pengkol
		33 Pasangan Ja		post	45 Taling Tarung
		34 Pasangan Ya			50 Wigyan
		36 Pasangan Ma		asc-base-desc	51 Cakra
		37 Pasangan Ga	<i>mahaprana</i>	base	67 Mahaprana Sha

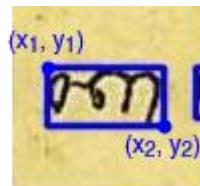
Pengurutan ini berdasarkan karakter akar suku kata, yakni karakter dengan kelas *wianjana*, *pasangan*, *murda*, *pasangan murda*, serta *tandha*. Kelas karakter tersebut dipilih karena karakter tersebut merupakan akar suku kata pada Bahasa Jawa. Pengurutan dilakukan berdasarkan posisi x pada setiap *midline* yang telah ditemukan seperti pada Gambar 3.17.



Gambar 3.17 Ilustrasi pengurutan karakter akar suku kata

d) Pengenalan Suku Kata

Tahapan selanjutnya adalah pengenalan suku kata menggunakan akar suku kata yang telah terdeteksi. Pengenalan suku kata didasarkan pada pengenalan karakter aksara lainnya yang berada di sekitar akar suku kata dan dibagi menggunakan klasifikasi kelas karakter sebelumnya. Untuk masing-masing klasifikasi memiliki batas (berwarna merah) seperti pada Tabel 3.3, dengan acuan posisi dari aksaranya, seperti pada Gambar 3.18. Batas tersebut digunakan untuk mengurangi *error* dan meminimalkan adanya kesalahan pengenalan.



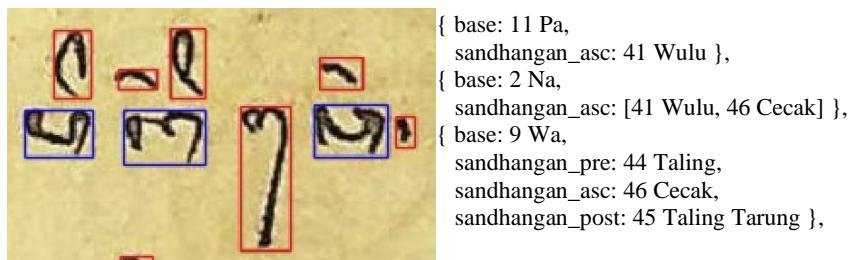
Gambar 3.18 Posisi x dan y yang digunakan untuk pengenalan sambungan

Tabel 3.3 Klasifikasi batas pengenalan karakter

Klasifikasi	Ilustrasi	Batas (x_1, y_1, x_2, y_2)	Kelas Karakter
<i>sandhangan_pre</i>		$x_1 - 50$ $y_1 - 10$ x_1 $y_2 + 80$	44 Taling
<i>sandhangan_pre_asc</i>		$x_1 - 110$ $y_1 - 60$ $x_2 + 10$ $y_1 + 20$	42 Pepet
<i>sandhangan_asc</i>		$x_1 - 20$ $y_1 - 60$ $x_2 + 20$ $y_1 + 10$	41 Wulu 46 Cecak
<i>sandhangan_post_asc</i>		x_1 $y_1 - 60$ $x_2 + 60$ $y_1 + 10$	47 Layar

Klasifikasi	Ilustrasi	Batas (x_1, y_1, x_2, y_2)	Kelas Karakter
<i>sandhangan_post</i> <i>_con</i>		$x_2 - 45$ $y_1 - 20$ $x_2 + 55$ $y_2 + 90$	43 Suku 48 Pangkon 49 Pengkol
<i>sandhangan_post</i>		x_2 $y_1 - 10$ $x_2 + 85$ $y_2 + 60$	45 Taling Tarung 50 Wignyan
<i>sandhangan_asc_</i> <i>base_desc</i>		$x_1 - 20$ $y_1 - 30$ $x_2 + 30$ $y_2 + 30$	51 Cakra
<i>pasangan_desc</i>		$x_1 - 30$ $y_2 - 20$ $x_2 + 60$ $y_2 + 80$	Semua <i>pasangan</i> kecuali yang ada pada <i>pasangan_base</i> dan <i>pasangan_base_desc</i>
<i>pasangan_base_d</i> <i>esc</i>		$x_1 - 10$ $y_1 - 10$ $x_2 + 40$ $y_2 + 60$	35 Pasangan Nya

Setelah dilakukan pengenalan, maka hasil pengenalan tersebut disimpan pada setiap karakter utama dalam bentuk objek seperti pada Gambar 3.19. Sehingga sambungan tersebut dapat dikonversi dan diurutkan sesuai dengan klasifikasinya, dan untuk menyeragamkan bentuk suku kata apabila terdapat perbedaan urutan *sambungan* dalam sebuah suku kata dengan posisi yang berbeda.



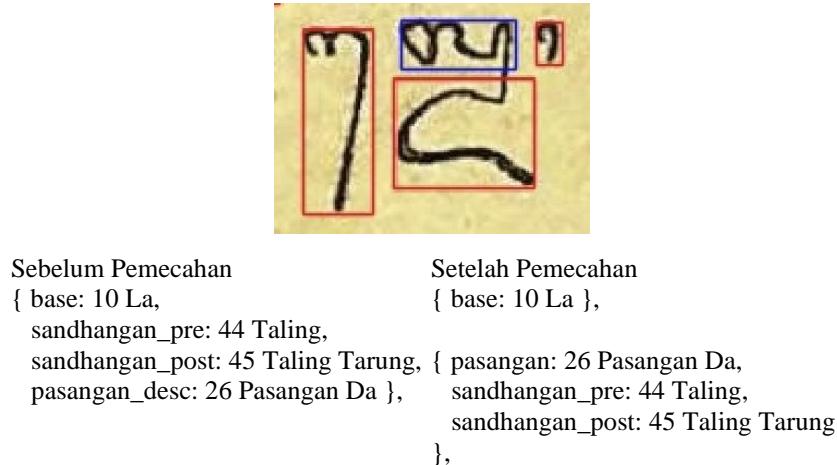
Gambar 3.19 Contoh hasil pengenalan suku kata

e) Pemecahan *Pasangan*

Tahapan yang tidak kalah penting selanjutnya adalah pemecahan *pasangan*, karena *pasangan* termasuk dalam aksara utama. Namun dalam proses pengenalan suku kata di atas, tidak semua *pasangan* berada dan memotong *midline*, sehingga akan susah untuk melakukan pengenalan *pasangan*. Maka dari itu, *pasangan* dalam proses di atas menjadi sambungan dari karakter utama lainnya, dan masuk ke dalam objek suku kata. Selain itu, penggunaan *pasangan* pada aksara Jawa menunjukkan pemisahan suku kata dari suku kata sebelumnya.

Perlu dilakukan pemecahan untuk menjadikan *pasangan* objek yang berdiri sendiri sehingga tidak mempengaruhi suku kata sebelumnya. Proses pemecahan objek dilakukan

dengan memindahkan seluruh properti objek awal, ke objek baru dengan *pasangan* sebagai karakter utamanya seperti pada Gambar 3.20. Sehingga karakter utama sebelumnya berubah menjadi objek sendiri karena tidak membutuhkan *sambungan*.



Gambar 3.20 Ilustrasi pemecahan *pasangan*

f) Penggabungan *Dataset*

Langkah selanjutnya adalah penggabungan *dataset*. *Dataset* yang awalnya berbentuk objek suku kata perlu dikonversi ke dalam bentuk yang lebih sederhana agar proses transliterasi dapat berjalan lebih mudah. Setiap objek suku kata harus digabungkan menjadi sebuah kata terlebih dahulu agar selaras dengan *dataset* terjemah transliterasi. *Dataset* tersebut kemudian digabungkan dengan *dataset* terjemah transliterasi berdasarkan urutan baris pada citra seperti pada Tabel 3.4.

Tabel 3.4 *Dataset* transliterasi setelah digabungkan

<i>Input</i> (suku kata)	<i>Output</i> (terjemah transliterasi)
10_La 14_Ya_46_Cecak 8_Sa_44_Taling 9_Wa 5_Ka 16_Ma 2_Na_41_Wulu 4_Ra 20_Nga 9_Wa_41_Wulu 54_Pada_Lingsa 57_Murda_Na 17_Ga	la yang sé wa ka ma ni ra nga wi / Na ga
18_Ba_44_Taling_45_Taling_Tarung 16_Ma 1_Ha 16_Ma 57_Murda_Na 32_Pasangan_Dha_41_Wulu 7_Ta_44_Taling_46_Cecak 13_Ja 17_Ga 7_Ta_48_Pangkon 6_Da_43_Suku 5_Ka 28_Pasangan_Sa_41_Wulu	bo ma a ma N dhi tèng ja ga t du k si
10_La 10_La_41_Wulu 7_Ta 7_Ta 2_Na 22_Pasangan_Na_44_Taling 9_Wa_44_Taling_46_Cecak_45_Taling_Tarung 2_Na 17_Ga 4_Ra_41_Wulu 54_Pada_Lingsa 7_Ta 7_Ta	la li ta ta n né wong na ga ri / ta ta

g) Augmentasi Data

Dari hasil pemrosesan transliterasi dihasilkan 881 baris *input output* transliterasi untuk proses latih pada model transliterasi. Untuk menghasilkan *dataset* yang siap untuk model transliterasi, perlu dilakukan augmentasi pada data tersebut. Augmentasi ini diperlukan untuk memperbanyak jumlah, serta meningkatkan variasi dalam *dataset*. Augmentasi yang digunakan dalam *dataset* transliterasi adalah pengacakan urutan pada setiap suku kata. Setiap interpretasi objek suku kata pada *input* mewakili terjemah sebuah suku kata pada *output*, sehingga proses pengacakan tidak mempengaruhi informasi utama pada *dataset*.

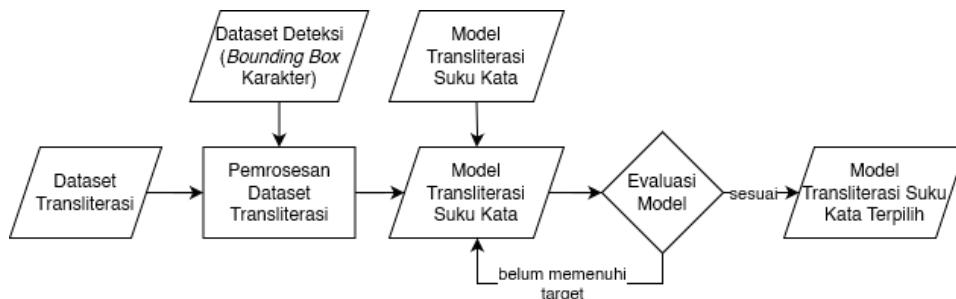
Data kemudian di proses dengan augmentasi menggunakan pengacakan urutan suku kata (seperti pada Tabel 3.5). Pengacakan diulangi sebanyak 40 kali pengacakan dan menghasilkan 24.284 baris. Hasil pengacakan tersebut kemudian digabungkan dengan *dataset* asli sehingga total terdapat 25.165 baris *dataset* yang siap digunakan pada proses latih transliterasi.

Tabel 3.5 Pengacakan urutan suku kata pada *dataset*

Sebelum pengacakan		Sesudah pengacakan	
<i>Input</i>	<i>Output</i>	<i>Input</i>	<i>Output</i>
10_La 14_Ya_46_Cecak 8_Sa_44_Taling 9_Wa 5_Ka 16_Ma 2_Na_41_Wulu 4_Ra 20_Nga 9_Wa_41_Wulu 54_Pada_Lingsa 57_Murda_Na 17_Ga	la yang sé wa ka ma ni ra nga wi / Na ga	5_Ka 54_Pada_Lingsa 14_Ya_46_Cecak 9_Wa 8_Sa_44_Taling 4_Ra 17_Ga 16_Ma 57_Murda_Na 2_Na_41_Wulu 10_La 20_Nga 9_Wa_41_Wulu	ka / yang wa sé ra ga ma Na ni la nga wi
18_Ba_44_Taling_45_Taling_T arung 16_Ma 1_Ha 16_Ma 57_Murda_Na 32_Pasangan_Dha_41_Wulu 7_Ta_44_Taling_46_Cecak 13_Ja 17_Ga 7_Ta_48_Pangkon 6_Da_43_Suku 5_Ka 28_Pasangan_Sa_41_Wulu	bo ma a ma N dhi teng ja ga t du k si	13_Ja 7_Ta_44_Taling_46_Cecak 5_Ka 18_Ba_44_Taling_45_Taling_Tar ung 16_Ma 32_Pasangan_Dha_41_Wulu 6_Da_43_Suku 17_Ga 1_Ha 28_Pasangan_Sa_41_Wulu 57_Murda_Na 7_Ta_48_Pangkon 16_Ma	ja tèng k bo ma dhi du ga a si N t ma
10_La 10_La_41_Wulu 7_Ta 7_Ta 2_Na 22_Pasangan_Na_44_Taling 9_Wa_44_Taling_46_Cecak_45 _Taling_Tarung 2_Na 17_Ga 4_Ra_41_Wulu 54_Pada_Lingsa 7_Ta 7_Ta	la li ta ta n né wong na ga ri / ta ta	9_Wa_44_Taling_46_Cecak_45_ Taling_Tarung 7_Ta 2_Na 7_Ta 54_Pada_Lingsa 4_Ra_41_Wulu 7_Ta 10_La 10_La_41_Wulu 17_Ga 7_Ta 22_Pasangan_Na_44_Taling 2_Na	wong ta n ta / ri ta la li ga ta né na

3.1.5 Proses Latih Model Transliterasi

Pada model transliterasi, data yang digunakan adalah rangkaian karakter pada sebuah baris berdasarkan urutan *bounding box* pada *dataset* sebelumnya. Setelah melakukan pemrosesan *dataset* transliterasi, *dataset* kemudian dilatih menggunakan model transliterasi seperti pada Gambar 3.21, model tersebut diuji dan dievaluasi, model dengan hasil akurasi paling baik akan disimpan dan digunakan untuk melakukan transliterasi hasil deteksi. Dari model inilah dihasilkan rangkaian suku kata sesuai dengan *input* deteksi.



Gambar 3.21 Rancangan proses latih model transliterasi

Dalam melakukan proses latih model transliterasi, perlu dilakukan *tokenisasi* dan *encoding* pada *dataset* terlebih dahulu. Sehingga model dapat melakukan proses latih menggunakan interpretasi dari suku kata. Selain itu juga dilakukan penambahan *post-padding* pada *dataset* agar *dataset* memiliki ukuran yang sama. Selain itu, proses latih juga

menggunakan berbagai jenis model *Neural Machine Translation* (NMT) dalam pengujian untuk menentukan jenis model dengan performa terbaik. Di antara NMT yang akan diuji adalah *Simple RNN*, *Bidirectional RNN* (BiRNN), LSTM, *Bidirectional LSTM* (BiLSTM), GRU, serta *Bidirectional GRU* (BiGRU).

3.1.6 Evaluasi Model Transliterasi

Model yang telah dilatih kemudian dievaluasi menggunakan nilai *Character Error Rate* (CER) dan *Word Error Rate* (WER). Semakin kecil nilai CER dan WER maka semakin akurat model yang dihasilkan. Selain itu nilai *val_loss* juga akan diperhatikan pada proses latih untuk menentukan model terbaik pada proses latih dan menghindari *overfitting*.

Pada evaluasi ini, nilai CER merepresentasikan tingkat kesalahan transliterasi pada level karakter, sedangkan nilai WER merepresentasikan tingkat kesalahan transliterasi pada level suku kata. Keduanya dihitung berdasarkan perbandingan *ground truth dataset* terjemah dengan hasil prediksi dari model transliterasi. Contoh perbedaan antara kedua metrik tersebut dijelaskan pada Tabel 3.6. Nilai CER cenderung lebih rendah dibandingkan dengan WER karena menghitung perbedaan karakter terjemahnya. Contoh pada Tabel 3.6 suku kata “na” dan “n” pada WER dihitung sebagai sebuah *error*, sedangkan pada CER *error* dihitung sebagai 1 penghapusan dan 1 penyisipan.

Tabel 3.6 Contoh perbedaan nilai metrik CER dan WER

Ground Truth		Hasil Prediksi	
	wong ta n ta / ri ta la li ga ta né na	wong ta na ta / ri ta la li ga ta né n	
		CER (%)	WER (%)
S (jumlah substitusi)		0	2
D (jumlah penghapusan)		1	0
I (jumlah penyisipan)		1	0
N (total)		38	13
Nilai akhir	$\frac{0 + 1 + 1}{38} = \frac{2}{38} \times 100\% = 5,3$	$\frac{2 + 0 + 0}{13} = \frac{2}{13} \times 100\% = 15,4$	

3.2 Peralatan Pendukung

Dibutuhkan peralatan pendukung yang cukup agar penelitian dapat berjalan dengan baik. Penelitian ini menggunakan perangkat keras berupa sebuah komputer dengan spesifikasi yang telah ditentukan. Spesifikasi perangkat keras tersebut dapat dilihat pada Tabel 3.7.

Tabel 3.7 Tabel Spesifikasi Perangkat Keras

No.	Sistem Operasi	Windows 10 Pro 64-bit (22H2, Build 19045)
1.	Memori	16384MB RAM
2.	CPU	AMD Ryzen 7 2700U Eight-Core Processor 3.20GHz
3.	GPU	NVIDIA GeForce RTX 2080 16 GB

3.3 Implementasi dan Uji Coba

Terdapat beberapa tahapan implementasi dan uji coba yang dilakukan pada penelitian ini. Di antaranya membangun *dataset* aksara Jawa, proses latih model deteksi, evaluasi model deteksi, praproses *dataset* transliterasi, proses latih model transliterasi, serta evaluasi model transliterasi. Pada subbab ini dijelaskan beberapa implementasi algoritma menggunakan *pseudocode* untuk memudahkan penjelasan.

3.3.1 Membangun Dataset Aksara Jawa

Sebelum melakukan proses latih model deteksi, *dataset* perlu diproses terlebih dahulu. Hal itu dilakukan agar *dataset* yang digunakan menjadi maksimal untuk setiap skenario pengujinya. Implementasi pemrosesan *dataset* meliputi deteksi *bounding box*, pembagian citra *split*, serta data augmentasi.

a) Deteksi *Bounding Box*

Deteksi *bounding box* dilakukan dengan melakukan serangkaian *image processing* terlebih dahulu, yakni *grayscale* dan *otsu thresholding*. Kemudian hasil *image processing* diproses dengan operasi dilasi. Selanjutnya mencari *connected components* untuk menentukan masing-masing objek *bounding box*. Implementasi deteksi *bounding box* seperti pada Algoritma 1.

Algoritma 1: Algoritma deteksi *bounding box*

Input : Image *img*; minimum area *min_area*; maximum area *max_area*;

1. *gray* = *convert_color*(*img*, *BGR2RGB*)
2. *gray* = *convert_color*(*gray*, *RGB2GRAY*)
// convert img color to *GRAY*
3. *gray* = *gaussian_blur*(*gray*, *kernel_size*)
4. *otsu_threshold* = *threshold*(*gray*, *thresh_val*, *max_val*, *type*)
// thresholding using otsu type
5. *dilated_img* = *dilate*(*otsu_threshold*, *kernel_size*)
// image dilatation using kernel size
6. *connected_component* = *connected*(*dilated_img*, *connectivity*)
// get connected component
7. *arr_bound_box* = []
8. For *i* < size of *connected_component* // loop through array
9. If *connected_component*[*i*].area > *min_area*
And *connected_component*[*i*] < *max_area*
10. Add *connected_component*[*i*] to *arr_bound_box*
// get array of bounding box from connected component
11. return *arr_bound_box*[]

Output: *arr_bound_box* array of bounding box;

Pada baris 1-2 dilakukan proses *grayscale* pada citra, kemudian dilanjutkan dengan proses *otsu thresholding* pada baris 3-4. Pada baris 5 dilakukan dilasi untuk meningkatkan hasil dari *connected components*. Lalu dilakukan pencarian *connected component* dengan memanggil fungsi *connected()* pada baris 6. Hasil pencarian berupa *array bounding box* kemudian difilter seperti pada baris 7-11 berdasarkan *min_area* dan *max_area* untuk mendapatkan *bounding box* yang sesuai.

b) Data Augmentasi

Sesuai dengan skenario *dataset*, perlu dilakukan *data augmentasi* pada citra. Proses augmentasi terbagi menjadi dua tahap, yaitu menggunakan *grayscale* seperti pada Algoritma 2. Sedangkan tahap kedua menggunakan *otsu thresholding* seperti pada Algoritma 3.

Algoritma 2: Algoritma *data augmentasi grayscale*

Input : Image *img*;

1. *gray* = *convert_color*(*img*, *BGR2RGB*) // convert image from *BGR* to *RGB* first
2. *gray_img* = *convert_color*(*gray*, *RGB2GRAY*) // convert img color to *GRAY*
3. Return *gray_img*

Output: *gray_img* grayscaled image;

Pada baris 1 dilakukan konversi warna dari BGR ke RGB terlebih dahulu. Hal itu disebabkan karena data citra dibaca dengan menggunakan sistem warna BGR oleh OpenCV. Kemudian citra baru dikonversi ke sistem warna GRAY pada baris 2 untuk mendapatkan mode *grayscale* dari citra.

Algoritma 3: Algoritma *data augmentasi otsu thresholding*

Input : Image *img*;

1. *blur = gaussian_blur(img, kernel_size)*
// *kernel_size* = (5,5) // kernel size of blurring
2. *otsu_threshold = threshold(blur, thresh_val, max_val, type)*
// *thresh_val* = 0 // threshold value to binarize the image
// *max_val* = 255 // maximum value using the max val of color, 255
7. Return *otsu_threshold*

Output: *otsu_threshold* otsu thresholded image;

Pada baris 1 dilakukan *image blurring* menggunakan *gaussian blur* terlebih dahulu untuk menghilangkan *noise* pada citra. Proses *image blurring* menggunakan *kernel size* 5 x 5 sebagai parameter dari *gaussian blur*. Kemudian hasil proses diproses dengan *thresholding* dengan *otsu threshold* pada baris 2 menggunakan parameter *threshold value* 0 untuk *binerisasi* hasil dan *maximum value* 255 sebagai nilai maksimal dalam warna citra.

c) Pembagian Citra *Split*

Pembagian citra *split* dilakukan dengan menggunakan algoritma *line detection*. Algoritma tersebut dilakukan untuk menentukan posisi paling tepat untuk melakukan pembagian, serta agar informasi *bounding box* tidak ikut hilang dari sebuah citra. Algoritma *line detection* dijalankan seperti pada Algoritma 4.

Algoritma 4: Algoritma *line detection* untuk membagi citra

Input : Horizontal Projection *h_{proj}*;

1. *arr_{peak} = argrelextrema(h_{proj}, ord)*
// get array of relative extrema with greater equal function
// *ord* = 90 // order of point comparison
2. Add 0 to *arr_{sep_line}*
3. *i* = 0
4. For *i* < size of *arr_{peak}* // loop through peak
5. Add *argmin(arr_{peak}[i] : arr_{peak}[i + 1])* to *arr_{sep_line}*
 // get minimum value between two peak
6. Add *h_{proj}.height* to *arr_{sep_line}*
7. Return *arr_{sep_line}*

Output: *arr_{sep_line}* array of separator line;

Pada baris 1 dilakukan pencarian nilai maksimum dari *horizontal projection* citra menggunakan fungsi ekstrem untuk menentukan posisi baris pada citra. Hasil dari pencarian tersebut berupa *array* posisi baris. Kemudian pada baris 2-6 dilakukan pencarian nilai minimum pada masing-masing baris untuk mendapatkan baris pemisah atau *separator line* dalam citra.

Dari Algoritma 4 didapatkan *array separator line* untuk menentukan batas antar baris. Batas ini kemudian dicocokkan dengan garis *split point*, sehingga apabila garis *split* memotong anotasi, maka garis *split* dapat di geser ke *separator line* terdekat di atas atau di bawahnya seperti pada Algoritma 5.

Algoritma 5: Algoritma pembagian citra menjadi n citra

Input : I Original Image; n_{split} Number of Split;

1. $h_{proj} = \text{horizontal projection of } I;$
2. $arr_{sep_line} = \text{line_detection}(h_{proj}) // \text{get array of midline}$
3. $i = 1; top_{val} = 0;$
4. For $i < n_{split} + 1 // \text{loop through number of splits}$
5. $point_{split} = \text{round}\left(\frac{I.height \times i}{n_{split}}\right) // \text{get } i\text{-split point}$
6. $j = 0; bottom_{val} = I.height // \text{set bottom point to max height}$
7. For $j < \text{size of } arr_{sep_line} // \text{loop find nearest separator bigger than } i\text{-split point}$
8. If $arr_{sep_line}[j] < bottom_{val}$ and $arr_{sep_line}[j] > point_{split}$
9. $bottom_{val} = arr_{sep_line}[j]$
10. Add $I[:, top_{val}: bottom_{val}]$ to $I_{split} // \text{save as new splitted image}$
11. $j = 0;$
12. For $j < \text{size of } arr_{sep_line} // \text{loop find nearest separator smaller than } i\text{-split point}$
13. If $arr_{sep_line}[j] > top_{val}$ and $arr_{sep_line}[j] < point_{split}$
14. $top_{val} = arr_{sep_line}[j]$
15. Return I_{split}

Output: $I_{split}[]$ Splitted Image;

Pada baris 1-2 dilakukan deteksi baris pada citra dengan mengonversi citra ke dalam *horizontal projection* untuk kemudian memanggil fungsi *line_detection()*. Kemudian pada baris 4-14 dilakukan iterasi untuk setiap n split untuk mendapatkan *split point*. Pada baris 7-9 dilakukan iterasi untuk mendapatkan *separator line* terkecil yang lebih besar dari *split point* untuk mendapatkan batas bawah, sedangkan baris 12-14 dilakukan iterasi untuk mendapatkan *separator line* terbesar yang lebih kecil dari *split point* untuk mendapatkan batas atas. Kemudian pada baris 10 dilakukan pengambilan pasangan batas atas dan batas bawah pada *split* ke- i sebagai sebuah citra *split*.

3.3.2 Proses Latih Model Deteksi

YOLOv5 menggunakan *PyTorch*, sehingga implementasi proses latih dapat menyesuaikan parameternya dengan *PyTorch*. *PyTorch* menyediakan beberapa parameter yang dapat disesuaikan untuk memaksimalkan proses latih menggunakan GPU. Selain itu juga diperlukan penyesuaian dengan skenario yang akan digunakan seperti pada Tabel 3.8.

Tabel 3.8 Parameter *PyTorch* pada proses latih YOLOv5

Parameter	Nilai	Deskripsi
--img	640	Ukuran input citra yang digunakan untuk proses latih
--batch	16	Jumlah <i>batch</i> yang digunakan
--epochs	10000	Jumlah <i>epoch</i> pada iterasi
--data	data.yaml	Lokasi dataset <i>input</i>
--cfg	yolov5s.yaml	Lokasi file <i>config</i> arsitektur model
--patience	0	Nilai EarlyStopping (<i>epoch</i> tanpa perubahan)
--save-period	200	Menyimpan <i>checkpoint</i> di setiap x epoch
--workers	4	Maksimal jumlah <i>dataloader</i>
--device	0	Indeks <i>device</i> yang digunakan

3.3.3 Evaluasi Model Deteksi

Evaluasi model deteksi dilakukan dengan menggunakan nilai *mAP* dan *F1 score*. Untuk menentukan nilai tersebut, batas *IoU* dibagi menjadi 2, yakni 0,5 dan 0,5-0,95. Kemudian untuk masing-masing objek *bounding box*, ditentukan nilai *precision* dan *recallnya*. Kemudian nilai rata-rata dari *precision* dan *recall* dihitung berdasarkan klasifikasi kelasnya untuk mendapatkan nilai *mAP* 0,5 dan *mAP* 0,5-0,95. Lalu terakhir, nilai *F1 score* dihitung menggunakan nilai *precision* dan *recall* sebagai pembanding utama akurasi pada setiap jenis kelas.

3.3.4 Praproses Dataset Transliterasi

Dataset transliterasi bersumber pada *dataset* deteksi maupun hasil deteksi dari model deteksi yang telah dibuat. Pemrosesan perlu dilakukan untuk mengubah data menjadi format yang diinginkan serta dapat mempengaruhi akurasi dalam pengenalan suku kata. Implementasi pemrosesan *dataset transliterasi* meliputi deteksi *midline*, pengurutan karakter, pengenalan suku kata, serta augmentasi data.

a) Deteksi *Midline*

Untuk mendapatkan rangkaian suku kata, diperlukan *midline* untuk menentukan akar suku katanya. Sehingga perlu untuk mendeteksi *midline* pada sebuah citra menggunakan deteksi *midline*. Algoritma deteksi *midline* dilakukan menggunakan Algoritma 6.

Algoritma 6: Algoritma deteksi *midline*

Input : Labels file *labels*;

1. For *label* in *labels*:
 2. If *label.class* in [*wianjana*, *pasangan*, *murda*, *tandha*] then
 // check if the label in root syllable class
 3. *h_{proj}*[*label.x₁*:*label.x₂*] += *label.width*
 // Add character width to projection
 4. *arr_{medial_textline}* = *line_detection(h_{proj})*
 // get array of midline
 5. Return *arr_{medial_textline}*

Output: *arr_{medial_textline}* array of midline;

Pada baris 1-3 dilakukan iterasi untuk membentuk *horizontal projection* berdasarkan anotasi kelas akar suku kata dan kelas *tandha*. Kelas akar suku kata terdiri dari kelas *wianjana*, *pasangan*, dan *murda*. Kemudian dilakukan *line detection* untuk mendeteksi baris pada hasil proyeksi tersebut seperti pada baris 4. Hasil deteksi dinamakan *midline* yang digunakan untuk mengenali posisi baris pada citra.

b) Pengurutan Karakter

Langkah selanjutnya adalah pengurutan karakter akar suku kata untuk mendapatkan suku kata yang tepat dan menghindari duplikasi deteksi karakter pada sebuah kalimat. Pengurutan dilakukan dengan mengurutkan karakter dari kiri ke kanan dan menggunakan hasil deteksi *midline* untuk mengetahui posisi baris suku kata pada sebuah citra. Pengurutan karakter mengikuti Algoritma 7.

Algoritma 7: Algoritma pengurutan karakter

Input : Labels file *labels*;

1. *labels* = *sort(labels)*
 // sort array of labels using the horizontal position
2. *syllable_seq* = []
 // array for collecting syllable from labels
3. For *label* in *labels*:

Algoritma 7: Algoritma pengurutan karakter

4. If *label.class* in [wianjana, pasangan, murda, tandha] then
 // check if the label in root syllable class
5. Add *label* to *syllable_seq*
 // Add root syllable to list
6. Return *syllable_seq*

Output: *syllable_seq* array of syllable sequence;

Pengurutan dilakukan dengan mengurutkan anotasi berdasarkan posisi horizontalnya terlebih dahulu, seperti pada baris 1. Kemudian anotasi yang telah diurutkan kemudian difilter berdasarkan jenis kelasnya berdasarkan akar suku katanya, sama seperti pada deteksi *midline*. Iterasi dilakukan pada baris 3-5 dan menghasilkan *output* berupa urutan kelas akar suku kata pada setiap barisnya.

c) Pengenalan Suku Kata

Setelah akar suku kata didapatkan, langkah selanjutnya adalah mengenali karakter lainnya yang berada di sekitar akar suku kata menggunakan pendekatan daerah *boundary* pada karakter. Batas *boundary* didefinisikan untuk setiap klasifikasi karakter di awal, sehingga setiap label dapat dikelompokkan berdasarkan posisi relatifnya terhadap akar suku kata. Proses pengenalan suku kata dilakukan seperti pada Algoritma 8.

Algoritma 8: Algoritma pengenalan suku kata

Input : Labels file *labels*; sequence syllables *syllable_seq*;

1. *bounds* = [*list_of_bound*]
 // define bound array of each class
2. *syllable_obj_seq* = []
 // array for collecting syllable object from labels
3. For *syllable* in *syllable_seq*
4. For *label* in *labels*:
5. For *bound* in *bounds*:
6. If *label.position* in *bound* then
 // check if the label in bound class
7. Add *label* to *syllable_obj_seq*[*bound.class*]
 // Add sambungan character to list of syllable
8. Return *syllable_obj_seq*

Output: *syllable_obj_seq* array of syllable object sequence;

Langkah pertama adalah pendefinisian *boundary* yang digunakan, seperti pada baris 1. Kemudian dilakukan iterasi untuk setiap akar suku kata yang sudah ditemukan lalu iterasi setiap anotasi yang ada, serta iterasi juga untuk setiap *boundary* seperti pada baris 3-7. Lalu dilakukan pengecekan posisi anotasi terhadap *boundary* yang ditentukan, apabila memenuhi maka anotasi akan ditambahkan ke dalam rangkaian objek suku kata seperti pada baris 6-7. Algoritma 8 mengembalikan objek suku kata berupa akar suku kata beserta objek lain yang ada di sekitarnya, atau disebut *sambungan*. Sehingga objek suku kata tersebut dapat diubah dan disesuaikan formatnya serta digunakan sebagai data *input* pada model transliterasi.

d) Augmentasi Data

Dalam augmentasi data, *dataset* objek suku kata dan terjemah suku kata diacak urutannya dengan urutan yang sama. Hal tersebut dilakukan menghasilkan *input* dan *output* yang dihasilkan tetap konsisten di setiap suku katanya. Proses pengacakan data dilakukan dengan mengikuti Algoritma 8.

Algoritma 8: Algoritma augmentasi data

Input : Number of shuffle $n_shuffle$; transliteration data $trans_data$;

1. $rand_input = []$; $rand_output = []$
2. For $i < n_shuffle$
 // loop as many as number of shuffle
3. For $input, output$ in $trans_data['input'], trans_data['output']$
4. If $len(input) == len(output)$
5. $rand_array = arange(len(input))$
 // get array interpretation
6. $shuffle(rand_array)$
7. Add $input[rand_array]$ to $rand_input$
8. Add $output[rand_array]$ to $rand_output$
 // Add randomized input and output to array
9. $result_data['input'] = rand_input$
10. $result_data['output'] = rand_output$
11. Return $result_data$

Output: $result_data$ randomized transliteration data;

Pada baris 3-8 dilakukan iterasi untuk mendapatkan *dataset* yang telah diacak dengan menggunakan indeks *random* yang sama pada data *input* dan data *output*. Kemudian pengacakan tersebut diulangi sebanyak $n\ shuffle$ yang ditentukan seperti pada baris 2. Proses pengacakan dilakukan apabila jumlah objek suku kata pada *input* dan jumlah suku kata pada *output* sama.

3.3.5 Proses Latih Model Transliterasi

Proses latih model transliterasi menggunakan beberapa jenis model *Neural Machine Translation* (NMT) memiliki parameter yang sama, bergantung pada skenario pengujian yang akan dilakukan. Terdapat beberapa jenis model yang diuji, yaitu *Simple RNN*, *Bidirectional RNN* (BiRNN), LSTM, *Bidirectional LSTM* (BiLSTM), GRU, serta *Bidirectional GRU* (BiGRU). Berikut implementasi untuk masing-masing jenis model pada Algoritma 9-14.

Algoritma 9: Implementasi model Simple RNN

Input : Number of input vocab in_vocab ; Number of output vocab out_vocab ;
Number of input timestep $in_timesteps$; Number of output timestep $out_timesteps$;
Number of units $units$;

1. $model = Sequential()$
2. $model.add(Embedding(in_vocab, out_vocab, in_timesteps, mask_zero = True))$
 # Encoder
3. $model.add(SimpleRNN(units))$
4. $model.add(RepeatVector(out_timesteps))$
 # Decoder
5. $model.add(SimpleRNN(units, return_sequences = True))$
6. $model.add(Dense(out_vocab, activation = 'softmax'))$
7. Return $model$

Output: $model$ SimpleRNN based transliteration model;

Terdapat 2 bagian utama dari *Simple RNN*, yakni *encoder* dan *decoder*. Baris 3-4 pada Algoritma 9 merupakan *encoder*, sedangkan baris 5 merupakan *decoder*. Sedangkan baris 6 merupakan *output layer* yang menentukan ukuran *output* dari model.

Algoritma 10: Implementasi model Bidirectional RNN

Input : Number of input vocab *in_vocab*; Number of output vocab *out_vocab*;
Number of input timestep *in_timesteps*; Number of output timestep *out_timesteps*;
Number of units *units*;

1. *model = Sequential()*
2. *model.add(Embedding(in_vocab, out_vocab, in_timesteps, mask_zero = True))*
Encoder
3. *model.add(Bidirectional(SimpleRNN(units)))*
4. *model.add(RepeatVector(out_timesteps))*
Decoder
5. *model.add(Bidirectional(SimpleRNN(units, return_sequences = True)))*
6. *model.add(Dense(4 * units, activation = 'relu'))*
7. *model.add(Dropout(0.5))*
8. *model.add(Dense(out_vocab, activation = 'softmax'))*
9. Return *model*

Output: *model* Bidirectional RNN based transliteration model;

Terdapat 2 bagian utama dari *Bidirectional RNN*, yakni *encoder* dan *decoder*. Baris 3-4 pada Algoritma 10 merupakan *encoder*, sedangkan baris 5-6 merupakan *decoder*. Sedangkan baris 7-8 merupakan *output layer* yang menentukan ukuran *output* dari model. Perbedaan model ini dengan *Simple RNN* adalah adanya layer *Bidirectional* yang membungkus model *Simple RNN*. Selain itu juga terdapat layer *Dropout* pada model untuk mencegah *overfitting* serta mempercepat proses latih.

Algoritma 11: Implementasi model LSTM

Input : Number of input vocab *in_vocab*; Number of output vocab *out_vocab*;
Number of input timestep *in_timesteps*; Number of output timestep *out_timesteps*;
Number of units *units*;

1. *model = Sequential()*
2. *model.add(Embedding(in_vocab, out_vocab, in_timesteps, mask_zero = True))*
Encoder
3. *model.add(LSTM(units))*
4. *model.add(RepeatVector(out_timesteps))*
Decoder
5. *model.add(LSTM(units, return_sequences = True))*
6. *model.add(Dense(out_vocab, activation = 'softmax'))*
7. Return *model*

Output: *model* LSTM based transliteration model;

Terdapat 2 bagian utama dari LSTM, yakni *encoder* dan *decoder*. Baris 3-4 pada Algoritma 11 merupakan *encoder*, sedangkan baris 5 merupakan *decoder*. Sedangkan baris 6 merupakan *output layer* yang menentukan ukuran *output* dari model.

Algoritma 12: Implementasi model Bidirectional LSTM

Input : Number of input vocab *in_vocab*; Number of output vocab *out_vocab*;
Number of input timestep *in_timesteps*; Number of output timestep *out_timesteps*;
Number of units *units*;

1. *model = Sequential()*

Algoritma 12: Implementasi model Bidirectional LSTM

```
2. model.add(Embedding(in_vocab, out_vocab, in_timesteps, mask_zero =  
True))  
# Encoder  
3. model.add(Bidirectional(LSTM(units)))  
4. model.add(RepeatVector(out_timesteps))  
# Decoder  
5. model.add(Bidirectional(LSTM(units, return_sequences = True)))  
6. model.add(Dense(4 * units, activation = 'relu'))  
7. model.add(Dropout(0.5))  
8. model.add(Dense(out_vocab, activation = 'softmax'))  
9. Return model
```

Output: *model* Bidirectional LSTM based transliteration model;

Terdapat 2 bagian utama dari *Bidirectional LSTM*, yakni *encoder* dan *decoder*. Baris 3-4 pada Algoritma 12 merupakan *encoder*, sedangkan baris 5-6 merupakan *decoder*. Sedangkan baris 7-8 merupakan *output layer* yang menentukan ukuran *output* dari model. Perbedaan model ini dengan LSTM adalah adanya layer *Bidirectional* yang membungkus model LSTM. Selain itu juga terdapat layer *Dropout* pada model untuk mencegah *overfitting* serta mempercepat proses latih.

Algoritma 13: Implementasi model GRU

Input : Number of input vocab *in_vocab*; Number of output vocab *out_vocab*;
Number of input timestep *in_timesteps*; Number of output timestep *out_timesteps*;
Number of units *units*;

```
1. model = Sequential()  
2. model.add(Embedding(in_vocab, out_vocab, in_timesteps, mask_zero =  
True))  
# Encoder  
3. model.add(GRU(units))  
4. model.add(RepeatVector(out_timesteps))  
# Decoder  
5. model.add(GRU(units, return_sequences = True))  
6. model.add(Dense(out_vocab, activation = 'softmax'))  
7. Return model
```

Output: *model* GRU based transliteration model;

Terdapat 2 bagian utama dari GRU, yakni *encoder* dan *decoder*. Baris 3-4 pada Algoritma 13 merupakan *encoder*, sedangkan baris 5 merupakan *decoder*. Sedangkan baris 6 merupakan *output layer* yang menentukan ukuran *output* dari model.

Algoritma 14: Implementasi model Bidirectional GRU

Input : Number of input vocab *in_vocab*; Number of output vocab *out_vocab*;
Number of input timestep *in_timesteps*; Number of output timestep *out_timesteps*;
Number of units *units*;

```
1. model = Sequential()  
2. model.add(Embedding(in_vocab, out_vocab, in_timesteps, mask_zero =  
True))  
# Encoder  
3. model.add(Bidirectional(GRU(units)))
```

Algoritma 14: Implementasi model Bidirectional GRU

```
4.   model.add(RepeatVector(out_timesteps))
    # Decoder
5.   model.add(Bidirectional(GRU(units, return_sequences = True)))
6.   model.add(Dense(4 * units, activation = 'relu'))
7.   model.add(Dropout(0.5))
8.   model.add(Dense(out_vocab, activation = 'softmax'))
9.   Return model
```

Output: *model* Bidirectional GRU based transliteration model;

Terdapat 2 bagian utama dari *Bidirectional GRU*, yakni *encoder* dan *decoder*. Baris 3-4 pada Algoritma 14 merupakan *encoder*, sedangkan baris 5-6 merupakan *decoder*. Sedangkan baris 7-8 adalah *output layer* yang menentukan ukuran *output* dari model. Perbedaan model ini dengan GRU adalah adanya layer *Bidirectional* yang membungkus model GRU. Selain itu juga terdapat layer *Dropout* pada model untuk mencegah *overfitting* serta mempercepat proses latih.

3.3.6 Evaluasi Model Transliterasi

Evaluasi model dilakukan dengan membandingkan nilai CER dan WER terhadap data uji. Evaluasi dilakukan dengan menggunakan *library* JiWER. Implementasi dari penentuan nilai CER dan WER dilakukan seperti pada Algoritma 15.

Algoritma 15: Implementasi evaluasi model transliterasi

Input : Prediction dataset *data_predict*;

```
1.   CER_list = []; WER_list = []
2.   For index, data in data_predict :
3.       Add CER(data['actual'], data['predicted']) to CER_list
4.       Add WER(data['actual'], data['predicted']) to WER_list
5.   Return mean(CER_list), mean(WER_list)
```

Output: *cer_rate* CER rate; *wer_rate* WER rate;

Dilakukan iterasi seperti pada baris 2-4 untuk mendapatkan nilai CER dan WER pada setiap baris. Nilai CER dan WER diperoleh dengan memanggil masing-masing fungsi seperti pada baris 3 dan 4. Kemudian nilai tersebut disimpan dalam sebuah *array* untuk dihitung nilai rata-rata akhirnya seperti pada baris 5.

BAB IV HASIL DAN PEMBAHASAN

4.1 Dataset Aksara Jawa

Keterbatasan *dataset* merupakan permasalahan dalam penelitian ini. Sehingga solusi dari penelitian ini adalah membangun *dataset* aksara Jawa yang dapat digunakan pada tahapan berikutnya. *Dataset* yang tersusun pada penelitian ini terbagi menjadi dua jenis, yaitu *Handwritten Javanese Character on Sewaka Manuscripts Detection* (HJCS_DETC) dan HJCS_DETC_SPLIT.

4.1.1 Dataset HJCS_DETC

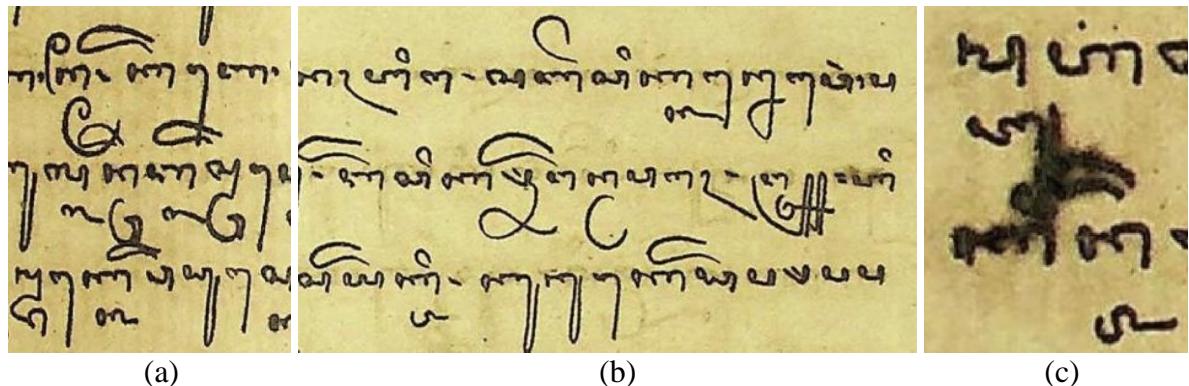
Dataset ini adalah *dataset* dasar yang digunakan untuk membangun *dataset* selanjutnya. Oleh karena itu, *dataset* ini harus melewati pemrosesan citra untuk mendapatkan label karakter beserta posisi anotasinya. Dari hasil pemrosesan, diperoleh kurang lebih 26.000 karakter yang terbagi ke dalam 74 kelas, dengan perincian jumlah per kelas pada citra seperti pada Tabel 4.1. Rata-rata terdapat 433 anotasi per citra, namun terlihat jika persebaran karakter setiap kelas yang sangat tidak seimbang pada dokumen, sehingga menjadi tantangan pada penelitian ini.

Tabel 4.1 Jumlah karakter Aksara Jawa pada setiap kelas

ID	Kelas Karakter	Jumlah Karakter	ID	Kelas Karakter	Jumlah Karakter	ID	Kelas Karakter	Jumlah Karakter
1	Ha	884	26	Pasangan Da	158	51	Cakra	243
2	Na	2,056	27	Pasangan Ta	247	52	Cakra Keret	6
3	Ca	122	28	Pasangan Sa	209	53	Pa Cerek	161
4	Ra	651	29	Pasangan Wa	108	54	Nga Lelet	94
5	Ka	1,225	30	Pasangan La	130	55	Pada Lingsa	1,355
6	Da	489	31	Pasangan Pa	182	56	Pada Madya	149
7	Ta	789	32	Pasangan Dha	46	57	Purwa Pada	3
8	Sa	987	33	Pasangan Ja	38	58	Murda Na	45
9	Wa	654	34	Pasangan Ya	10	59	Murda Ka	1
10	La	711	35	Pasangan Nya	46	60	Murda Ta	4
11	Pa	575	36	Pasangan Ma	115	61	Murda Sa	10

ID	Kelas Karakter	Jumlah Karakter	ID	Kelas Karakter	Jumlah Karakter	ID	Kelas Karakter	Jumlah Karakter	
12		80	37		Pasangan Ga	34		Murda Pa	1
13		204	38		Pasangan Ba	93		Murda Ga	96
14		379	39		Pasangan Tha	22		Murda Ba	1
15		79	40		Pasangan Nga	31		Pasangan Murda Ga	14
16		810	41		Wulu	2,278		Pasangan Murda Ta	104
17		386	42		Pepet	749		Mahaprana Sha	81
18		281	43		Suku	1,715		Swara A	10
19		32	44		Taling	1,644		Swara I	4
20		391	45		Taling Tarung	524		Swara U	1
21		239	46		Cecak	1,260		Swara E	2
22		284	47		Layar	392		Rekan Kha	5
23		47	48		Pangkon	405		Rekan Za	1
24		14	49		Pengkol	86		Pasangan Rekan Za	1
25		197	50		Wignyan	443			

Selain itu *dataset* ini juga memiliki beberapa karakteristik, seperti (1) densitas karakter yang tinggi, banyak karakter yang memotong karakter lainnya (2) penulisan karakter yang tegak dan cenderung lurus (3) terdapat beberapa karakter yang buram akibat penggunaan tinta yang kurang baik. Contoh karakteristik pada *dataset* ini terlihat pada Gambar 4.1. Karakter lain dari *dataset* ini adalah persebaran jenis karakternya yang tidak merata.



Gambar 4.1 Beberapa karakteristik dataset HJCS_DETC (a) contoh karakter yang memotong (b) penulisan karakter yang tegak dan cenderung lurus (c) karakter yang buram

4.1.2 Dataset HJCS_DETC_SPLIT

Dataset ini bertujuan sebagai *dataset* yang digunakan untuk proses latih model deteksi aksara Jawa pada manuskrip Jawa. Oleh karena itu pada *dataset* ini dilakukan *data augmentation* pada citra untuk menghasilkan varian berbeda, serta adanya pembagian citra *split* untuk mengatasi permasalahan sumber daya latih model deteksi yang tidak terlalu efektif untuk citra dengan resolusi besar. Pada *dataset* ini juga terdapat pengurangan jumlah kelas dengan mempertimbangkan jumlah klasifikasi kluster kelasnya seperti pada Tabel 4.2. *Dataset* yang awalnya memiliki 74 kelas karakter berkurang hingga 67 kelas untuk digunakan pada proses deteksi selanjutnya.

Tabel 4.2 Persebaran klasifikasi kluster kelas

No.	Kluster Kelas	Indeks Kelas	Jumlah Karakter	Keterangan
1.	Wianjana	1-20	11.785	Digunakan
2.	Pasangan Wianjana	21-40	2.250	Digunakan
3.	Sandhangan	41-54	10.000	Digunakan
4.	Tandha	55-57	1.507	Digunakan
5.	Murda	58-64	158	Digunakan
6.	Pasangan Murda	65-66	118	Digunakan
7.	Mahaprana	67	81	Digunakan
8.	Swara	68-71	17	Tidak Digunakan
9.	Rekan	72-73	6	Tidak Digunakan
10.	Pasangan Rekan	74	1	Tidak Digunakan

Dari klasifikasi kelas pada Tabel 4.2, kelas yang digunakan pada pemrosesan selanjutnya adalah kelas *Wianjana*, *Pasangan Wianjana*, *Sandhangan*, *Tandha*, *Murda*, *Pasangan Murda*, dan *Mahaprana*. Kelas-kelas tersebut digunakan karena mencukupi jumlah karakter minimalnya. Sementara itu, kelas lainnya dikecualikan dan tidak digunakan pada proses selanjutnya.

Kemudian hasil dari proses data augmentation menggunakan *grayscale* dan *otsu thresholding*. Selain itu juga dilakukan proses pembagian citra *split*, yakni dengan *split 3*, *split 4*, dan *split 5* terhadap *dataset* menghasilkan variasi *dataset* berjumlah 9 variasi, seperti pada Tabel 4.3. Masing-masing variasi *dataset* kemudian dinamakan dengan skenario S1-S9 sesuai dengan Tabel 4.3.

Tabel 4.3 Variasi dataset HBCL_DETC_SPLIT

Data Augmentasi	Jenis Split	Kode Dataset	Jumlah Citra
Original Dataset	Tanpa split	HJCS_DETC	60
	Split 3 (S1)	HJCS_DETC_SPLIT_3_ORIG	180
Original Dataset	Split 4 (S2)	HJCS_DETC_SPLIT_4_ORIG	240
	Split 5 (S3)	HJCS_DETC_SPLIT_5_ORIG	300
Original Dataset + Grayscale	Split 3 (S4)	HJCS_DETC_SPLIT_3_ORIG_GRAY	360
	Split 4 (S5)	HJCS_DETC_SPLIT_4_ORIG_GRAY	480
	Split 5 (S6)	HJCS_DETC_SPLIT_5_ORIG_GRAY	600
Original Dataset + Grayscale + Otsu Thresholding	Split 3 (S7)	HJCS_DETC_SPLIT_3_ORIG_GRAY_THRESH	540
	Split 4 (S8)	HJCS_DETC_SPLIT_4_ORIG_GRAY_THRESH	720
	Split 5 (S9)	HJCS_DETC_SPLIT_5_ORIG_GRAY_THRESH	900

Untuk membangun model deteksi, *dataset* HJCS_DETC_SPLIT kemudian dibagi menjadi 70% data *training*, 20% data *validation* dan 10% data *testing* dengan persebaran jumlah citra seperti pada Tabel 4.4. Pada proses ini tidak dilakukan pemisahan data berdasarkan augmentasinya. Namun pemisahan dilakukan secara acak berdasarkan rasio pembagiannya.

Tabel 4.4 Persebaran data *training*, *validation*, dan *testing*

Data Augmentasi	Jenis Split	Kode Dataset	Dataset Train	Dataset Valid	Dataset Test
Original Dataset	Split 3 (S1)	HJCS_DETC_SPLIT_3_ORIG	126	36	18
	Split 4 (S2)	HJCS_DETC_SPLIT_4_ORIG	252	72	36
	Split 5 (S3)	HJCS_DETC_SPLIT_5_ORIG	378	108	54
Original Dataset + Grayscale	Split 3 (S4)	HJCS_DETC_SPLIT_3_ORIG_GRAY	168	48	24
	Split 4 (S5)	HJCS_DETC_SPLIT_4_ORIG_GRAY	336	96	48
	Split 5 (S6)	HJCS_DETC_SPLIT_5_ORIG_GRAY	504	144	72
Original Dataset + Grayscale + Otsu Thresholding	Split 3 (S7)	HJCS_DETC_SPLIT_3_ORIG_GRAY_THRESH	210	60	30
	Split 4 (S8)	HJCS_DETC_SPLIT_4_ORIG_GRAY_THRESH	420	120	60
	Split 5 (S9)	HJCS_DETC_SPLIT_5_ORIG_GRAY_THRESH	632	179	89

4.2 Hasil dan Pembahasan Evaluasi Model Deteksi

Dari hasil proses latih model deteksi, dilakukan evaluasi terhadap pengaruh *dataset* pada model serta analisis kelas karakternya. Selain itu juga dilakukan evaluasi *overfitting* pada model sebagai validasi karakter dari model yang dihasilkan. Pada evaluasi dan pembahasan ini metrik utama yang digunakan adalah *mAP* dan *F1 score*. Contoh hasil deteksi pada citra tes seperti pada Gambar 4.2.



Gambar 4.2 Contoh hasil deteksi pada citra tes

4.2.1 Evaluasi Model dan Pengaruh Dataset

Untuk menentukan model terbaik, model dievaluasi berdasarkan beberapa skenario. Di antaranya adalah berdasarkan data augmentasi, dan jumlah data *image splittingnya*. Tabel 4.5 menunjukkan hasil evaluasi akurasi pada masing-masing skenario pengujian.

Tabel 4.5 Evaluasi Model Deteksi

	Data Skenario	TP	FP	FN	Precision	Recall	mAP50	<i>mAP50-95</i>	F1 score
Original Dataset	Split 3 (S1)	3975	935	1810	0,810	0,687	0,757	0,580	0,743
	Split 4 (S2)	4657	1161	1595	0,801	0,745	0,792	0,605	0,772
	Split 5 (S3)	3927	774	2551	0,835	0,606	0,660	0,518	0,703
Original Dataset + Grayscale	Split 3 (S4)	7564	739	4076	0,911	0,650	0,706	0,589	0,759
	Split 4 (S5)	7593	1084	4448	0,875	0,631	0,683	0,563	0,733
	Split 5 (S6)	9409	1000	3486	0,904	0,730	0,763	0,635	0,807
Original Dataset + Grayscale + Otsu Thresholding	Split 3 (S7)	12760	1379	4896	0,902	0,723	0,785	0,667	0,803
	Split 4 (S8)	12679	829	6141	0,939	0,674	0,721	0,621	0,784
	Split 5 (S9)	14808	1590	4391	0,903	0,771	0,814	0,696	0,832

Dari data hasil evaluasi dari Tabel 4.5, didapatkan bahwa untuk skenario menggunakan *Original Dataset*, model terbaik didapatkan oleh skenario S2 pada *split* 4, dengan *F1 score* 77,2% dan *mAP* 79,2%. Sedangkan rata-rata model yang menggunakan *dataset Original Dataset* mendapatkan *F1 score* 73,9% dan *mAP* 73,7%. Sedangkan skenario menggunakan *Original Dataset + Grayscale*, model terbaik didapatkan pada skenario S6 pada *split* 5, dengan *F1 score* 80,7% dan *mAP* 76,3%. Untuk rata-rata model yang menggunakan *dataset Original Dataset + Grayscale* mendapatkan *F1 score* 76,6% dan *mAP* 71,8%. Kemudian skenario menggunakan *Original Dataset + Grayscale + Otsu Thresholding*, model terbaik didapatkan pada skenario S9 pada *split* 5, dengan *F1 score* 83,2% dan *mAP* 81,4%. Kemudian rata-rata model yang menggunakan *dataset Original Dataset + Grayscale + Otsu Thresholding* mendapatkan *F1 score* 80,6% dan *mAP* 77,3%.

Dari ketiga jenis *dataset*, dapat disimpulkan bahwa *data augmentasi* varian *Original Dataset + Grayscale + Otsu Thresholding* mendapatkan hasil yang terbaik. Hal ini juga menunjukkan bahwa permasalahan varian *dataset* dapat diselesaikan menggunakan *data augmentasi dataset*. Semakin tinggi variansi *dataset* maka model yang dihasilkan akan memiliki akurasi yang lebih baik. Tentunya dengan risiko jumlah data yang semakin besar serta proses data latih yang lebih lama dari *dataset* normalnya.

Selain itu, berdasarkan pembagian *split*nya, nilai *F1 score* dan *mAP* tidak memiliki peningkatan yang pasti, ditunjukkan dengan skenario pada *data augmentasi* yang sama, model terbaik masih naik turun. Skenario S2 untuk *Original Dataset*, S6 untuk *Original Dataset + Grayscale*, dan S9 untuk *Original Dataset + Grayscale + Otsu Thresholding*. Hal tersebut kemungkinan terjadi karena beberapa faktor, seperti perbedaan jumlah anotasi, perbedaan ukuran citra, atau faktor lainnya.

Pada proses analisis ditemukan fakta bahwa adanya perbedaan jumlah anotasi. Mekanisme pembagian *split* mengakibatkan perbedaan jumlah anotasi. Metode *line detection* yang digunakan untuk menghindari hilangnya informasi menyebabkan jumlah karakter per skenario menjadi berbeda, seperti pada Tabel 4.6.

Tabel 4.6 Perbedaan jumlah anotasi pada skenario

No.	Jenis Dataset	Jumlah Anotasi / Jumlah Augmentasi	Selisih Jumlah dengan Data Original
1	Data <i>Original</i>	25999	-
2	<i>Original Dataset Split</i> 3 (S1)	29201	3202
3	<i>Original Dataset Split</i> 4 (S2)	30580	4581
4	<i>Original Dataset Split</i> 5 (S3)	32533	6534
5	<i>Original Dataset + Grayscale Split</i> 3 (S4)	58402 / 2 = 29201	3202
6	<i>Original Dataset + Grayscale Split</i> 4 (S5)	61160 / 2 = 30580	4581
7	<i>Original Dataset + Grayscale Split</i> 5 (S6)	65066 / 2 = 32533	6534
8	<i>Original Dataset + Grayscale + Otsu Thresholding Split</i> 3 (S7)	87603 / 3 = 29201	3202
9	<i>Original Dataset + Grayscale + Otsu Thresholding Split</i> 4 (S8)	65066 / 3 = 30580	4581
10	<i>Original Dataset + Grayscale + Otsu Thresholding Split</i> 5 (S9)	97599 / 3 = 32533	6534

Dari data pada Tabel 4.6 diketahui bahwa semakin besar jumlah *split*, maka semakin besar juga selisih jumlah anotasinya dengan data *original*, hal inilah yang menjadi penyebab perbedaan akurasi pada setiap *dataset*. Semakin banyak perbedaan jumlah anotasinya, maka semakin sulit untuk ditentukan kelas apa saja yang mengalami penambahan karakter, dan berapa persen penambahannya.

Selain itu, faktor lain juga yang kemungkinan mempengaruhi adalah ukuran citra *input*, perbedaan *split* pada citra menyebabkan semakin beragamnya juga ukuran citra yang digunakan pada *dataset* seperti pada Tabel 4.7. Sedangkan model YOLOv5 akan otomatis melakukan *resizing* citra ke ukuran 640 x 640. Sehingga citra yang ideal seharusnya mendekati citra *input* dari YOLOv5 yakni 640 x 640.

Tabel 4.7 Rata-rata tinggi citra pada skenario

No.	Jenis Dataset	Rata-Rata Tinggi Citra (piksel)
1	Data <i>Original</i>	1786,6
2	<i>Original Dataset Split 3 (S1)</i>	674,4
3	<i>Original Dataset Split 4 (S2)</i>	530,4
4	<i>Original Dataset Split 5 (S3)</i>	454,5
5	<i>Original Dataset + Grayscale Split 3 (S4)</i>	674,4
6	<i>Original Dataset + Grayscale Split 4 (S5)</i>	530,4
7	<i>Original Dataset + Grayscale Split 5 (S6)</i>	454,5
8	<i>Original Dataset + Grayscale + Otsu Thresholding Split 3 (S7)</i>	674,4
9	<i>Original Dataset + Grayscale + Otsu Thresholding Split 4 (S8)</i>	530,4
10	<i>Original Dataset + Grayscale + Otsu Thresholding Split 5 (S9)</i>	454,5

Dari Tabel 4.7, citra yang paling mendekati *input* YOLOv5 adalah *dataset* dengan *split* 3, sedangkan akurasi model menunjukkan sebaliknya. Model dengan *split* 3 justru menunjukkan performa terendah sedangkan model dengan *split* 5 menunjukkan performa terbaiknya. Dari berbagai faktor tersebut, menunjukkan bahwa jumlah *split* pada citra tidak mempengaruhi akurasi pada model, namun faktor-faktor lain tersebut yang mempengaruhi akurasi pada model. Direkomendasikan apabila citra dapat menyesuaikan *input* dari model YOLOv5 untuk mendapatkan akurasi yang maksimal.

4.2.2 Analisis Kelas Karakter

Secara keseluruhan, model terbaik ditunjukkan pada skenario S9, dengan nilai *F1 score* 83,2% dan *mAP* 81,4%. Sedangkan rata-rata kesembilan model mendapatkan nilai *F1 score* 77,1% dan *mAP* 74,2%. Hal ini menunjukkan jika model sudah dapat mendekripsi karakter aksara Jawa dengan baik. Hanya saja, jika dilihat lebih lanjut, terdapat beberapa kelas yang mendapatkan nilai yang cukup buruk pada kesembilan model, seperti pada Tabel 4.8.

Tabel 4.8 Kelas dengan *mAP* cukup buruk

Kelas	Jumlah Karakter	S1	S2	S3	S4	S5	S6	S7	S8	S9
12 Dha	80	0,532	0,195	0,515	0,371	0,682	0,802	0,664	0,748	0,778
24 Pasangan Ra	14	0,077	0,27	0,443	0,305	0,069	0,153	0,132	0,028	0,194
34 Pasangan Ya	10	0,074	0,054	0,046	0,083	0,137	0,031	0,045	0,022	0,045
40 Pasangan Nga	31	0,235	0,653	0,453	0,332	0,676	0,715	0,78	0,411	0,557
56 Purwa Pada	3	0	0	0	0,995	0,995	0,665	0,995	0	0,995
58 Murda Ka	1	0	0	0	0,002	0	0	0,166	0	0
59 Murda Ta	4	0	0,027	0	0,001	0,056	0,007	0	0,683	0
60 Murda Sa	10	0,012	0,029	0,31	0,576	0,131	0,218	0,382	0,253	0,354
61 Murda Pa	1	0	0	0	0,062	0	0,009	0,011	0	0
64 Murda Ba	1	0	0	0	0	0	0	0,497	0	0
67 Pasangan Murda Ga	14	0,266	0,404	0,83	0,269	0,124	0,312	0,524	0,398	0,388

Terdapat 10 dari 11 di antara kelas pada Tabel 4.8 memiliki nilai *mAP* yang cukup buruk karena memiliki jumlah karakter yang sangat sedikit dibandingkan dengan karakter lainnya. Hal itu menunjukkan jika permasalahan persebaran karakter yang tidak rata masih belum tertangani sepenuhnya. Sebagian besar dari kelas-kelas tersebut memang karakter yang sangat jarang muncul, seperti aksara *murda*.

Sementara itu pada beberapa kelas jika diamati memiliki akurasi yang rendah. Hal itu disebabkan karena beberapa aksara memiliki bentuk yang sama dengan aksara lainnya, seperti pada Tabel 4.9. Aksara-aksara tersebut bahkan memang memiliki bentuk yang sama sehingga muncul kemungkinan jika karakter tersebut memiliki nilai rendah akibat kesamaan fitur citra.

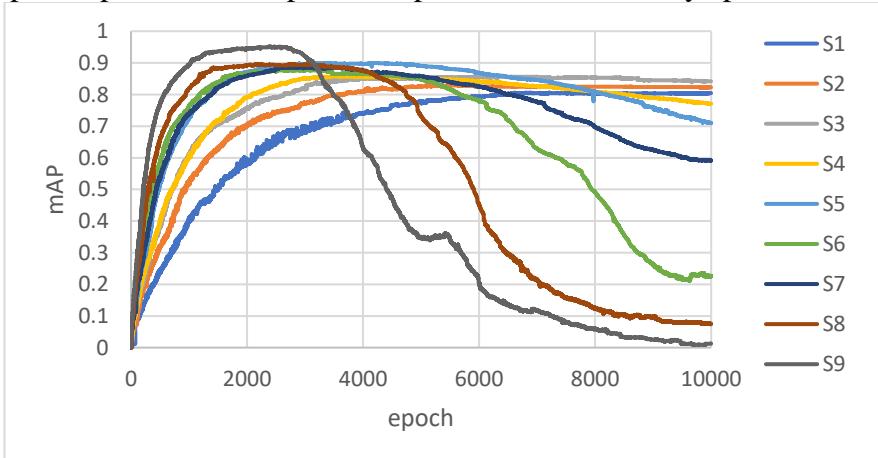
Tabel 4.9 Kelas dengan kesamaan bentuk citra

No.	Aksara 1	Aksara 2	Keterangan
1			Berbeda bentuk aksara, hanya bentuknya yang serupa Dha Wa
2			Bentuk sama, hanya posisi yang beda Pasangan Ra Ra
3			Bentuk sama, hanya posisi yang beda Pasangan Ya Ya
4			Bentuk sama, hanya posisi yang beda Pasangan Nga Nga
5			Bentuk sama, hanya posisi yang beda Pasangan Murda Ga Murda Ga

Dari beberapa analisis pada Tabel 4.9, dapat disimpulkan bahwa *dataset* yang digunakan pada penelitian ini masih belum memenuhi target minimal jumlah data yang dibutuhkan. Sehingga diperlukan penambahan *dataset* untuk mengatasinya. Selain itu penggunaan metode augmentasi yang tepat juga dapat mengatasi masalah tersebut.

4.2.3 Analisis *Overfitting* Model

Dalam melakukan latih data, sistem dari YOLOv5 menyarankan untuk setidaknya terdapat 1500 objek dalam *dataset* untuk sebuah kelas, jadi setidaknya apabila jumlah data tidak memenuhi target minimal tersebut, maka seharusnya target tersebut dicapai dengan meningkatkan *epoch* pada pelatihan model. Namun, belum ada jumlah *epoch* ideal yang pasti untuk mencukupi target minimal tersebut, sehingga pada penelitian ini standar 10.000 *epoch* digunakan untuk melakukan proses latih data. Oleh karena itu, dilakukan perbandingan akurasi *mAP* selama proses pelatihan data pada setiap skenario dan hasilnya pada Gambar 4.3.



Gambar 4.3 Perbandingan *mAP validation* selama proses latih masing-masing skenario

Dari Gambar 4.2 didapatkan bahwa pada skenario S6, S8, dan S9 terdapat penurunan *mAP* secara drastis, dan terlihat menurun mendekati 0 di *epoch* 10.000. Hal itu menunjukkan bahwa pada skenario dengan citra semakin banyak (data augmentasi serta *split* banyak) maka akan semakin cepat terjadi *overfitting* pada model. Pada skenario S6 terlihat mulai terjadi *overfitting* pada sekitar *epoch* 6000, pada S8 pada sekitar 5000, dan S9 pada sekitar 3000. Hal itu menunjukkan dampak lain bagi penggunaan *dataset* yang besar tanpa diimbangi dengan variasi data yang cukup.

Oleh karena itu dari data tersebut dapat disimpulkan bahwa pada penelitian ini, proses pelatihan model menggunakan 10.000 *epoch* sudah optimal. Apabila menggunakan *epoch* yang lebih besar lagi maka kemungkinan model akan mengalami *overfitting* akan semakin besar. Sehingga, perlunya mencari solusi yang tepat dan lebih sesuai agar permasalahan data yang kurang, dapat diimbangi dengan model yang tidak *overfit*.

4.3 Hasil dan Pembahasan Evaluasi Model Transliterasi

Dari hasil proses latih model transliterasi, model kemudian dievaluasi berdasarkan variasi data, jenis model, serta *hyperparameter*nya. Pengukuran hasil evaluasi pada pembahasan ini menggunakan metrik utama *Character Error Rate* (CER) dan *Word Error Rate* (WER). Pada masing-masing pengujian didapatkan skenario optimal yang selanjutnya digunakan pada pengujian selanjutnya. Contoh hasil transliterasi terdapat pada Gambar 4.4.

co	column 2	column 3	column 4
	source	actual	predicted
0	1_Ha_44_Taling_28_Pasangan_Sa_11_Pa_11_Pa_44_Taling_7_Ta_46_Cecak_43_Suku_10_La_46_Cecak_41_Wulu_10_La_46_Cecak_41_Wulu_11_Pa_20_Nga_44_Taling_54_Pada_Lingsa_11_Pa_11_Pa_44_Taling_11_Pa_11_Pa_41_Wulu_8_Sa_44_Taling	é sé pa pé tung ling ling pa ngé / pa pé pa lí s	é sé pa tung ling pa ngé / pa lí s
1	5_Ka_1_Ha_7_Ta_41_Wulu_17_Ga_41_Wulu_54_Pada_Lingsa_2_Na_2_Na_10_La_44_Taling_50_Wignyan_14_Ya_2_Na_43_Suku_7_Ta_18_Ba_22_Pasangan_Na_44_Taling_21_Pasangan_Ha_43_Suku_8_Sa_44_Taling_46_Cecak	kè ta ti gi / n na lèh ya nu tang bē n u sèng	kè ta ti gi / nu n lèh ya tang bē n u sèng
2	2_Na_11_Pa_11_Pa_43_Suku_5_Ka_7_Ta_5_Ka_42_Pepet_54_Pada_Lingsa_27_Pasangan_Ta_44_Taling_22_Pasangan_Na_22_Pasangan_Na_41_Wulu_11_Pa_43_Suku_20_Nga_46_Cecak_41_Wulu_7_Ta_2_Na_48_Pangkon_2_Na_2_Na_4_Ra_2_Na_54_Pada_Lingsa_8_Sa_43_Suku	n pa pu ka t kē / té na ni pu nging ta n n na ra na / su	na pu ka t ka / té na ni nging ta na / su
3	1_Ha_11_Pa_46_Cecak_29_Pasangan_Wa_2_Na_22_Pasangan_Na_42_Pepet_14_Ya_1_Ha_30_Pasangan_La_41_Wulu_3_Ca_42_Pepet_2_Na_22_Pasangan_Na_6_Da_44_Taling_45_Taling_Tarung_1_Ha_20_Nga_54_Pada_Lingsa_17_Ga_50_Wignyan_7_Ta_42_Pepet_46_Cecak_2_Na_41_Wulu_4_Ra_20_Nga_42_Pepet_47_Layar	n pang wa ha na y n ling gah cé doh hé a ng / na tèng ni ra ngér	a pang wa na cé y a ng ngér cé doh ng n ng / tèng ra gah
4	5_Ka_25_Pasangan_Ka_20_Nga_17_Ga_46_Cecak_43_Suku_23_Pasangan_Ca_2_Na_31_Pasangan_Pa_2_Na_16_Ma_9_Wa_44_Taling_46_Cecak_45_Taling_Tarung_8_Sa_7_Ta_31_Pasangan_Pa_54_Pada_Lingsa_7_Ta_51_Cakra_9_Wa_2_Na_44_Taling_50_Wignyan	n tra ng a gung pa na kē ka m wong sa ta pa / p wé néh	n nga pa na pa n m wong sa ta pa / p néh
5	22_Pasangan_Ga_9_Wa_14_Taling_24_Pasangan_Pa_44_Taling_6_Da_44_Taling_8_Sa_2_Na_11_Pa_46_Cecak_41_Wulu_8_Sa_44_Taling_46_Cecak_45_Taling_Tarung_8_Sa_7_Ta_31_Pasangan_Pa_54_Pada_Lingsa_7_Ta_51_Cakra_9_Wa_2_Na_44_Taling_50_Wignyan	n tra ng a gung pa na kē ka m wong sa ta pa / p wé néh	n nga pa na pa n m wong sa ta pa / p néh

Gambar 4.4 Contoh hasil transliterasi

4.3.1 Perbandingan Model berdasarkan Variasi Data

Dilakukan pengujian untuk membandingkan variasi data, yaitu data *original* dengan data menggunakan augmentasi. Setiap pengujian dilakukan dengan menggunakan jenis model LSTM dan parameter yang sama dengan tujuan untuk mengetahui pengaruh variasi data pada performa model sebagai acuan untuk pengujian-pengujian selanjutnya. Adapun pengukuran yang digunakan nilai rata-rata *Character Error Rate* (CER) dan *Word Error Rate* (WER).

Tabel 4.10 Perbandingan akurasi model berdasarkan jenis variasi data

Variasi Data	Jumlah Data	CER (%)	WER (%)
<i>Data Original</i>	881	88,4	90,8
<i>Data Original + Augmentasi</i>	25.164	48,8	60,6

Dari hasil pengujian pada Tabel 4.10, didapatkan bahwa nilai CER dan WER terhadap data *original* dengan augmentasi lebih baik dibandingkan data *original*. Pada model dengan data *original*, nilai CER dan WER cukup tinggi yakni 88,4% dan 90,8%, karena perbedaan jumlah data yang cukup besar. Penambahan data pada model dengan augmentasi menyebabkan peningkatan performa pada model. Sehingga, pada pengujian selanjutnya data yang digunakan adalah data *original* dengan augmentasi.

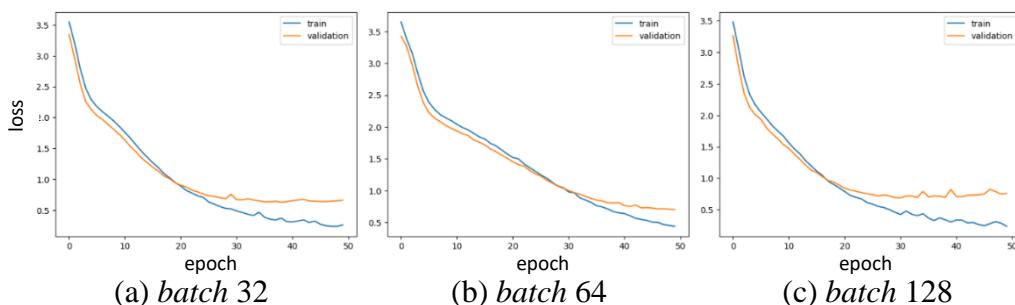
4.3.2 Perbandingan Model berdasarkan Jenis Model

Dilakukan pengujian untuk membandingkan jenis model, yaitu pada model *Simple RNN*, *LSTM*, dan *GRU*. Selain itu, juga dilakukan pengujian untuk model-model tersebut dengan menambahkan skema *bidirectional* (*BiRNN*, *BiLSTM*, dan *BiGRU*). Setiap pengujian dilakukan dengan menggunakan data yang sama dan menggunakan parameter *number of units* 256, dengan tujuan untuk mengetahui pengaruh jenis model dan *batch size* pada performa model dan sebagai acuan untuk pengujian-pengujian selanjutnya. Adapun pengukuran yang digunakan adalah nilai rata-rata *Character Error Rate* (CER) dan *Word Error Rate* (WER).

Tabel 4.11 Perbandingan akurasi model berdasarkan jenis model dan *batch size*

Jenis Model	<i>batch size</i>	CER (%)	WER (%)
<i>Simple RNN</i>	32	98	94,7
	64	94,5	91,5
	128	88,1	91,9
<i>BiRNN</i>	32	96	94,2
	64	94,6	93,4
	128	90,6	92,2
<i>LSTM</i>	32	45,2	56,6
	64	48,8	60,6
	128	58,6	73,8
<i>BiLSTM</i>	32	18,9	26,3
	64	16,7	23,3
	128	19,8	27,3
<i>GRU</i>	32	47,8	59,2
	64	53,4	66,3
	128	59,2	74,4
<i>BiGRU</i>	32	24,4	32,9
	64	32,5	42
	128	30,2	39,5

Dari hasil pengujian pada Tabel 4.11, didapatkan bahwa nilai CER dan WER paling tinggi didapatkan pada skenario jenis model *BiLSTM*, dengan skenario *batch size* 64. Skenario tersebut mendapatkan nilai CER 16,7% dan WER 23,3% jauh lebih baik dibandingkan dengan jenis model lainnya yang berada di kisaran 30-90%. Hal tersebut dapat terjadi karena arsitektur model *BiLSTM* dapat menyimpan informasi secara dua arah, masa depan (*forward*) dan sebaliknya (*backward*) sehingga model menghasilkan prediksi yang lebih akurat.



Gambar 4.5 Perbandingan nilai *loss* dan *val_loss* pada proses latih model *BiLSTM*

Dari grafik perbandingan pada Gambar 4.5 dapat diketahui jika nilai *loss* dan *val_loss* yang didapatkan pada proses latih model tidak mengalami *overfitting* yang terlalu besar Terlihat model mulai mengalami *overfit* pada *epoch* 20-30 dengan nilai yang tidak terlalu besar. Hal itu menunjukkan jika model BiLSTM dapat digunakan pada proses selanjutnya.

Tabel 4.12 Contoh perbandingan hasil deteksi BiLSTM

No.	<i>Ground Truth</i>	<i>batch size</i>	Hasil prediksi	CER (%)	WER (%)
1	sé ma la yang nga wi ra ka wa / Na ni ga	32	sé ma la ga nga la ra wa ka / Na ni ga	17,5	30,8
		64	sé ma la yang nga wi ra ka wa / Na ni ga	0	0
		128	sé ma la yang nga ga ra ka wa / Na ni ga	5	7,7
2	bo t N ma si ga k dhi du ja tèng ma a	32	bo t N ma si ga k dhi du ja tèng ma a	0	0
		64	bo t N ma si ga k dhi du ja tèng ma a	0	0
		128	bo t N ma si ga k dhi du ja tèng ma a	0	0
3	wong ta n ta / ri ta la li ga ta né na	32	wong ta na ta / ri ta la li ga ta né n	5,3	15,4
		64	wong ta n ta / ri ta la li ga ta né n	2,6	7,7
		128	wong ta n ta / ri ta la li ga ta né n	2,6	7,7

Dari contoh hasil perbandingan pada Tabel 4.12 didapatkan bahwa pada contoh no.1 hasil prediksi dengan menggunakan model BiLSTM *batch* 32 dan 128 memiliki nilai CER dan WER > 0 karena terdapat beberapa perbedaan suku kata, sedangkan model BiLSTM *batch* 64 memiliki nilai CER dan WER 0. Kemudian pada Tabel 4.12 contoh no.2 hasil prediksi menggunakan model BiLSTM di semua skenario *batch* memiliki nilai CER dan WER 0. Sedangkan pada Tabel 4.12 contoh no.3, setiap model memiliki nilai CER dan WER > 0. Hal itu menunjukkan bahwa model BiLSTM dengan *batch size* 64 memiliki hasil transliterasi yang paling baik pada skenario ini, sehingga dapat digunakan untuk skenario pengujian selanjutnya.

Selain itu pada Tabel 4.12 contoh no.3, diketahui bahwa semua model menghasilkan prediksi yang salah. Hal itu merupakan akibat dari augmentasi data berupa pengacakan urutan suku kata serta pemecahan *pasangan*. Pada *dataset* tersebut, terindikasi beberapa kemunculan dari objek suku kata yang sama, namun memiliki *output* yang berbeda seperti pada Tabel 4.13.

Tabel 4.13 Beberapa kemunculan objek suku kata yang tidak konsisten

<i>Input</i>	<i>Output</i>	<i>Output seharusnya</i>	Keterangan
9_Wa_44_Taling_46_Cecak_45_Taling _Tarung 7_Ta 2_Na 7_Ta 54_Pada_Lingsa 4_Ra_41_Wulu 7_Ta 10_La 10_La_41_Wulu 17_Ga 7_Ta 22_Pasangan_Na_44_Taling 2_Na	wong ta n ta / ri ta la li ga ta né n	wong ta n ta / ri ta la li ga ta né na	Seharusnya karakter 2_Na memiliki <i>output</i> “na” karena tidak memiliki properti 48_Pangkon serta tidak diiringi dengan <i>pasangan</i>

<i>Input</i>	<i>Output</i>	<i>Output seharusnya</i>	Keterangan
18_Ba_47_Layar 55_Pada_Madya 7_Ta 2_Na 16_Ma_47_Layar 1_Ha_44_Taling_45_Taling_Tarung 7_Ta 2_Na_41_Wulu 34_Pasangan_Ya_44_Taling 21_Pasangan_Ha 54_Pada_Lingsa 4_Ra 5_Ka 17_Ga 16_Ma	bar # ta n mar o t ni yè a / ra ka ga ma	bar # ta na mar o ta ni yè a / ra ka ga ma	Seharusnya karakter 2_Na memiliki <i>output</i> “na” dan 7_Ta memiliki <i>output</i> “ta” karena <i>tidak</i> memiliki properti 48_Pangkon serta tidak diiringi dengan <i>pasangan</i>

Dari beberapa kesalahan pada Tabel 4.13 dapat dilihat bahwa kesalahan *output* terjadi akibat beberapa akar suku kata yang tidak diiringi dengan karakter *pangkon* atau *pasangan* memiliki *output* mati/tanpa vokal. Hal itu bisa terjadi karena pada proses pengacakan urutan, karakter yang awalnya diiringi dengan *pasangan* menjadi berubah urutannya sehingga tidak terdeteksi apakah seharusnya *outputnya* memiliki huruf vokal atau tidak. Kesalahan ini bisa dihindari dengan menambahkan karakter 48_Pangkon pada proses pemecahan *pasangan* atau penambahan aturan pada proses pengacakan urutan *dataset* agar tidak terjadi *output* yang ambigu.

4.3.3 Perbandingan Model berdasarkan Hyperparameter

Dilakukan 2 skenario pengujian untuk menentukan parameter terbaik, yaitu pada parameter *optimizer* dan *learning rate*. Setiap pengujian dilakukan dengan menggunakan data yang sama, dengan tujuan untuk mengetahui pengaruh penggunaan parameter pada performa model. Adapun pengukuran yang digunakan adalah menggunakan nilai rata-rata *Character Error Rate* (CER) dan *Word Error Rate* (WER).

Pada pengujian pertama, dilakukan pengujian terhadap parameter *optimizer*. Pengujian dilakukan dengan menggunakan variasi antara *optimizer* Adagrad, Adadelta, Adam, RMSProp, dan SGD. Pengujian ini menggunakan model BiLSTM dengan *number of units* 256 dan *batch size* 64. Adapun pengukuran yang digunakan pada pengujian ini adalah menggunakan nilai CER dan WER.

Tabel 4.14 Perbandingan akurasi model berdasarkan parameter *optimizer*

<i>Optimizer</i>	CER (%)	WER (%)
Adagrad	96,2	94
Adadelta	98	94,4
Adam	16,7	23,3
RMSProp	36,1	46,5
SGD	98	94,4

Dari Tabel 4.14 diperoleh bahwa perubahan parameter *optimizer* berpengaruh terhadap performa model. Performa model terbaik diperoleh dari model dengan *optimizer* Adam yang memiliki nilai CER 16,7% dan WER 23,3%. Sedangkan model dengan *optimizer* RMSProp memiliki CER 36,1% dan WER di atas 46,5%. Sedangkan model dengan *optimizer* Adagrad, Adadelta, dan SGD memiliki CER dan WER di atas 90%.

Pada pengujian kedua, dilakukan pengujian terhadap parameter *learning rate*. Pengujian dilakukan dengan melakukan variasi antara *learning rate* dengan nilai 0,0005; 0,001; 0,002; 0,005 dan 0,01. Selain itu juga dilakukan variasi *batch size* untuk mengetahui apakah pengaruh perbedaan parameter tersebut terpengaruh pada jumlah *batch* yang berbeda.

Tabel 4.15 Perbandingan akurasi model berdasarkan parameter *learning rate*

<i>learning rate</i>	CER (%)	WER (%)
0,0005	23,7	31,5
0,001	16,7	23,3
0,002	19,7	27,4
0,005	22,7	32,5
0,01	64,4	82,5

Dari Tabel 4.15 diperoleh bahwa perubahan parameter *learning rate* berpengaruh terhadap performa model. Model dengan *learning rate* lebih dari 0,001 cenderung memiliki CER 19-64% dan WER 27-82%, sedangkan model dengan *learning rate* kurang dari 0,001 cenderung memiliki CER 23-48% dan WER 31-60%. Performa model terbaik diperoleh dari model dengan *learning rate* 0,001 yang menghasilkan nilai CER 16,7% dan WER 23,3%.

Halaman ini sengaja dikosongkan

BAB V KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil pengujian dan implementasi yang telah dilakukan, diperoleh beberapa kesimpulan terhadap penelitian ini, sebagai berikut :

1. Pemrosesan aksara Jawa terhadap 60 citra manuskrip *Serat Sewaka* menghasilkan *dataset* baru yang dinamakan *Handwritten Javanese Character on Sewaka Manuscripts Detection* (HJCS_DETC). *Dataset* ini terdiri dari 74 kelas yang terdiri dari aksara *wianjana*, *sandhangan*, *murda*, *pasangan*, *pasangan murda* serta sebagian aksara *murda*, *pasangan murda*, *swara*, dan *rekan*.
2. Pembuatan model deteksi aksara Jawa dilakukan menggunakan model YOLOv5. Model deteksi ini menggunakan *dataset* HJCS_DETC_SPLIT, yakni *dataset* dengan proses data augmentasi dan pembagian citra *split* serta pengurangan jumlah kelas menjadi 67 kelas sebagai upaya peningkatan performa model pada beberapa fokus kelas. Proses latih dilakukan berdasarkan 9 skenario *dataset* serta pembagian terhadap data *training*, *validation*, dan *testing* sebesar 70%, 20%, dan 10%.
3. Pembuatan model transliterasi suku kata aksara Jawa dilakukan menggunakan model *Neural Machine Translation* (NMT) seperti RNN, LSTM, dan GRU. *Dataset* yang digunakan pada model membutuhkan pemrosesan terlebih dahulu, yakni melalui proses deteksi *midline*, pengurutan karakter, pengenalan suku kata, dan augmentasi data.
4. Hasil evaluasi terhadap model deteksi menggunakan model YOLOv5 menunjukkan hasil sebagai berikut:
 - a. Menghasilkan model dengan akurasi terbaik pada skenario 9 (S9) dengan nilai *F1 score* 83,2% dan *mAP* 81,4% sehingga model deteksi dapat dikatakan cukup baik dalam melakukan pengenalan karakter terhadap 67 kelas aksara Jawa.
 - b. Peningkatan variasi *dataset* (data augmentasi) terbukti dapat meningkatkan akurasi pada model, sedangkan variasi *split* pada citra tidak berpengaruh terhadap akurasi pada model akibat pengaruh dari banyak faktor.
 - c. Penggunaan 10.000 *epoch* pada pelatihan model menghasilkan model yang *overfit* pada beberapa skenario, sehingga perlu untuk meningkatkan variasi data dengan menambah *dataset* atau menggunakan metode augmentasi lainnya.
5. Hasil evaluasi terhadap model transliterasi menggunakan LSTM menunjukkan hasil sebagai berikut:
 - a. Menghasilkan model berdasarkan jenis model dengan akurasi terbaik pada jenis model BiLSTM dengan nilai CER 16,7% dan WER 23,3% menggunakan data dengan augmentasi.
 - b. Terdapat beberapa kesalahan yang terjadi pada *dataset* akibat augmentasi data dan pemecahan pasangan, sehingga membutuhkan metode augmentasi yang lebih tepat dalam pemrosesan *dataset* transliterasi.
 - c. Menghasilkan model berdasarkan parameter dengan akurasi terbaik pada parameter *optimizer* Adam dan *learning rate* 0,001 dengan nilai CER 16,7% dan WER 23,3%.

5.2 Saran

Baik selama penggerjaan penelitian ini maupun berdasarkan pada pengamatan, kami mengusulkan beberapa saran sebagai berikut :

1. Meningkatkan jumlah data serta variasi data yang digunakan dalam penelitian, untuk meningkatkan performa model deteksi serta transliterasi.
2. Melibatkan lebih banyak ahli Aksara Jawa atau mengembangkan metode alternatif

yang lebih efisien untuk mempercepat proses pelabelan citra.

3. Melakukan pengujian pada *dataset* yang telah dibuat agar kualitas *dataset* dapat divalidasi.
4. Meningkatkan metode augmentasi dan pengenalan yang lebih baik pada proses transliterasi untuk meningkatkan performa model transliterasi.
5. Implementasi model ke dalam *website* atau *mobile* atau perangkat lainnya untuk meningkatkan kegunaan dari model deteksi transliterasi.
6. Akan lebih baik menggunakan *dataset original* saja pada evaluasi model deteksi, untuk mendapatkan hasil evaluasi yang lebih akurat.
7. Pengujian sebaiknya dilakukan dengan membandingkan kinerja *dataset original* dengan setelah augmentasi untuk mempengaruhi pengaruh penggunaan augmentasi dengan lebih akurat.

DAFTAR PUSTAKA

- Abdulwahab, S., Moreno, A., & Jabreel, M. (2018). *Deep Learning Models for Paraphrases Identification*. <https://doi.org/10.13140/RG.2.2.15743.46240>
- Arifianto, T. (2016). *Segmentasi Aksara pada Tulisan Aksara Jawa Menggunakan Adaptive Threshold* [Institut Teknologi Sepuluh Nopember]. <https://repository.its.ac.id/41389/>
- Basak, H., Kundu, R., & Sarkar, R. (2022). *MFSNet: A Multi Focus Segmentation Network for Skin Lesion Segmentation*. <https://doi.org/10.1016/j.patcog.2022.108673>
- Boudraa, O., Hidouci, W., & Michelucci, D. (2019). *Degraded Historical Documents Images Binarization Using a Combination of Enhanced Techniques*. https://www.researchgate.net/publication/330701266_Degraded_Historical_Documents_Images_Binarization_Using_a_Combination_of_Enhanced_Techniques
- Christian Adi Pradhana, S., Novia Wisesty, U., & Sthevanie, F. (2020). Pengenalan Aksara Jawa dengan Menggunakan Algoritma Convolutional Neural Network. *e-Proceeding of Engineering*. <https://openlibrary.telkomuniversity.ac.id/home/catalog/id/156876/slug/pengenalan-aksara-jawa-dengan-menggunakan-algoritma-convolutional-neural-network.html>
- Damayanti, F., Suprapto, Y. K., & Yuniarso, E. M. (2020). Segmentation of Javanese Character in Ancient Manuscript using Connected Component Labeling. *CENIM 2020 - Proceeding: International Conference on Computer Engineering, Network, and Intelligent Multimedia 2020*, 412–417. <https://doi.org/10.1109/CENIM51130.2020.9297954>
- Dias, J., Santos, P. A., Cordeiro, N., Antunes, A., Martins, B., Baptista, J., & Gonçalves, C. (2022). *State of the Art in Artificial Intelligence applied to the Legal Domain*. <http://arxiv.org/abs/2204.07047>
- Fauziyah, Y., Ilyas, R., Kasyidi, F., Achmad Yani, J., Sains dan Informatika, F., Jenderal Achmad Yani, U., & Terusan Sudirman, J. (2022). *Mesin Penterjemah Bahasa Indonesia-Bahasa Sunda Menggunakan Recurrent Neural Networks* (Vol. 16, Nomor 2). <https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index>
- Gonzalez, R. C., & Woods, R. E. (Richard E. (2018). *Digital image processing*.
- Hendrawati, T. (2018). Digitalisasi Manuskrip Nusantara Sebagai Pelestari Intelektual Leluhur Bangsa. *Media Pustakawan*, 25, 24–32. <https://doi.org/https://doi.org/10.37014/medpus.v25i4.196>
- Himamunanto, A. R., & Setyowati, E. (2017). Partisi Blok Teks Menuju Restorasi Kerusakan Aksara Jawa. *Jurnal Infact*, x, No.x, 1–5. http://www.e-jurnal.ukrimuniversity.ac.id/detail.php?id_konten=257&id_jurnal=5&id_volume=53
- Indrawijaya, M., & Adipranata, R. (2015). Aplikasi Ekstraksi Fitur Citra Hufur Jawa Berdasarkan Morfologinya. *Jurnal Infra*, 3, 260–266. <https://publication.petra.ac.id/index.php/teknik-informatika/article/view/2722>
- Islam, M. N. Al, & Khan, S. K. (2019). *HishabNet: Detection, Localization and Calculation of Handwritten Bengali Mathematical Expressions*. <http://arxiv.org/abs/1909.00823>
- Jonnalagadda, V. K. (2019, Januari 31). *Object Detection YOLO v1 , v2, v3*. Medium. <https://medium.com/@venkatakrishna.jonnalagadda/object-detection-yolo-v1-v2-v3-c3d5eca2312a>
- Kesiman, M. W. A., & Dermawan, K. T. (2021). AKSALont: Automatic Transliteration Application for Balinese Palm Leaf Manuscripts with LSTM Model. *Jurnal Teknologi dan Sistem Komputer*, 9(3), 142–149. <https://doi.org/10.14710/jtsiskom.2021.13969>
- Li, Z., Tian, X., Liu, X., Liu, Y., & Shi, X. (2022). A Two-Stage Industrial Defect Detection Framework Based on Improved-YOLOv5 and Optimized-Inception-ResnetV2 Models. *Applied Sciences (Switzerland)*, 12(2). <https://doi.org/10.3390/app12020834>

- Maxwell, A. E., Warner, T. A., & Guillén, L. A. (2021). Accuracy Assessment in Convolutional Neural Network-based Deep Learning Remote Sensing Studies—Part 1: Literature Review. Dalam *Remote Sensing* (Vol. 13, Nomor 13). MDPI AG. <https://doi.org/10.3390/rs13132450>
- Novaliandy, R., & Widiarti, A. R. (2022). Klasifikasi Aksara Jawa Cetak Menggunakan Jaringan Syaraf Tiruan Backpropagation. *EJournal UP Batam*. <https://ejurnal.upbatam.ac.id/index.php/prosiding/article/view/5351>
- Nugroho, H., Hakimah, M., Augusta, T., & Teknologi Adhi Tama Surabaya, I. (2021). Pengenalan Pola Dengan Penggunaan Metode Ekestraksi Fitur Zernike Moment Pada Citra Aksara Jawa Kontemporer dan Aksara Jawa Kawi. *Seminar Nasional Sains dan Teknologi Terapan IX 2021*. <http://ejurnal.itats.ac.id/sntekpan/article/view/2254>
- Paoletti, M. E., Haut, J. M., Plaza, J., & Plaza, A. (2018). A New Deep Convolutional Neural Network for Fast Hyperspectral Image Classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145, 120–147. <https://doi.org/https://doi.org/10.1016/j.isprsjprs.2017.11.021>
- Pasaribu, D. J. M., Kusrini, & Sudarmawan. (2023). Peningkatan Akurasi Klasifikasi Sentimen Ulasan Makanan Amazon Dengan Bidirectional LSTM Dan Bert Embedding. *Inspiration : Jurnal Teknologi Informasi dan Komunikasi*, 10(1), 9–20. <https://jurnal.akba.ac.id/index.php/inspiration/article/view/2568>
- Paterson, J. (2023). *Statistik Situs Sastra Jawa*. Sastra Jawa. <https://www.sastra.org/arsip-dan-sejarah/editorial/1691-statistik-situs>
- Paulus, E., Suryani, M., Hadi, S., & Natsir, F. (2018). An initial study to solve imbalance sundanese handwritten dataset in character recognition. *Proceedings of the 3rd International Conference on Informatics and Computing, ICIC 2018*. <https://doi.org/10.1109/IAC.2018.8780496>
- Puteri, D. I. (2023). Implementasi Long Short Term Memory (LSTM) dan Bidirectional Long Short Term Memory (BiLSTM) Dalam Prediksi Harga Saham Syariah. *Euler : Jurnal Ilmiah Matematika, Sains dan Teknologi*, 11(1), 35–43. <https://doi.org/10.34312/euler.v11i1.19791>
- Putra, A. S. (2020). *Identifikasi Aksara Jawa pada Naskah Kuno dengan Metode CNN*.
- Schreurs, A. (2021). *Automatic School Handwriting Detection and Classification based on YOLO and Vision Transformers models*.
- Sucayyo, N. (2023, Mei 27). *Digitalisasi Naskah Kuno: Mengabadikan Tulisan-Tulisan dari Abad Silam*. VOA Indonesia. <https://www.voaindonesia.com/a/digitalisasi-naskah-kuno-mengabadikan-tulisan-tulisan-dari-abad-silam-/7111752.html>
- Suciati, N., Sutramiani, N. P., & Siahaan, D. (2022). LONTAR-DETC: Dense and High Variance Balinese Character Detection Method in Lontar Manuscripts. *IEEE Access*, 10, 14600–14609. <https://doi.org/10.1109/ACCESS.2022.3147069>
- Sugianela, Y., & Suciati, N. (2019). Ekstraksi Fitur pada Pengenalan Karakter Aksara Jawa Berbasis Histogram of Oriented Gradient. Dalam *JUTI: Jurnal Ilmiah Teknologi Informasi* (Vol. 17, Nomor 1).
- Sutramiani, N. P. (2022). *Pengenalan Suku Kata Aksara Bali Berbasis Aturan Pasang Aksara Dan Deep Learning Untuk Transliterasi Naskah Lontar Kuno* [Institut Teknologi Sepuluh Nopember]. <https://repository.its.ac.id/94879/>
- Wang, Y., Feng, Z., Song, L., Liu, X., & Liu, S. (2021). Multiclassification of Endoscopic Colonoscopy Images Based on Deep Transfer Learning. *Computational and Mathematical Methods in Medicine*, 2021. <https://doi.org/10.1155/2021/2485934>
- Widiarti, A. R., Harjoko, A., Marsono, & Hartati, S. (2014). Preprocessing Model of Manuscripts in Javanese Characters. *Journal of Signal and Information Processing*,

- 05(04), 112–122. <https://doi.org/10.4236/jsip.2014.54014>
- Widiarti, A. R., & Hartati, S. (2013). Line Segmentation of Javanese Image of Manuscripts in Javanese Scripts. *International Journal of Engineering Innovation & Research*, 2(3), 2277–5668. <https://www.researchgate.net/publication/265167159>
- Widiarti, A. R., Pulungan, R., Harjoko, A., Marsono, & Hartati, S. (2018). A Proposed Model for Javanese Manuscript Images Transliteration. *Journal of Physics: Conference Series*, 1098(1). <https://doi.org/10.1088/1742-6596/1098/1/012014>
- Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional Neural Networks an Overview and Application in Radiology. Dalam *Insights into Imaging* (Vol. 9, Nomor 4, hlm. 611–629). Springer Verlag. <https://doi.org/10.1007/s13244-018-0639-9>
- Zain, F. H., & Santoso, H. (2021). Sistem Deteksi Kerusakan Gedung Menggunakan Algoritma YOU ONLY LOOK ONCE Dengan Unmanned Aero Vehicle. *Jurnal Politeknik Negeri Jakarta*.

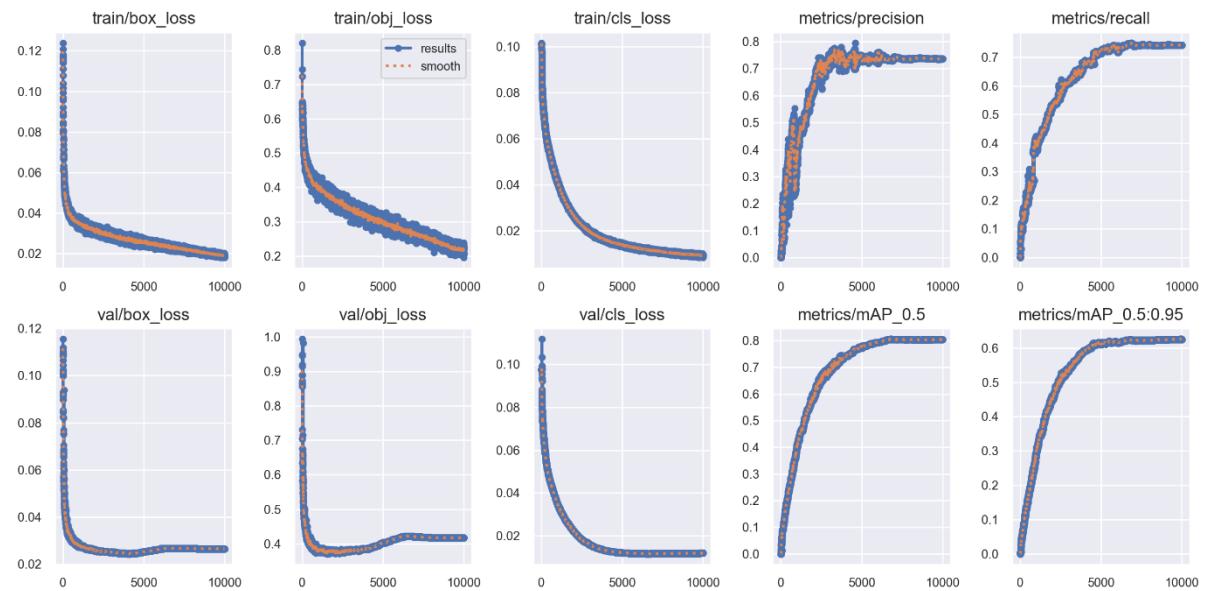
Halaman ini sengaja dikosongkan

LAMPIRAN-LAMPIRAN

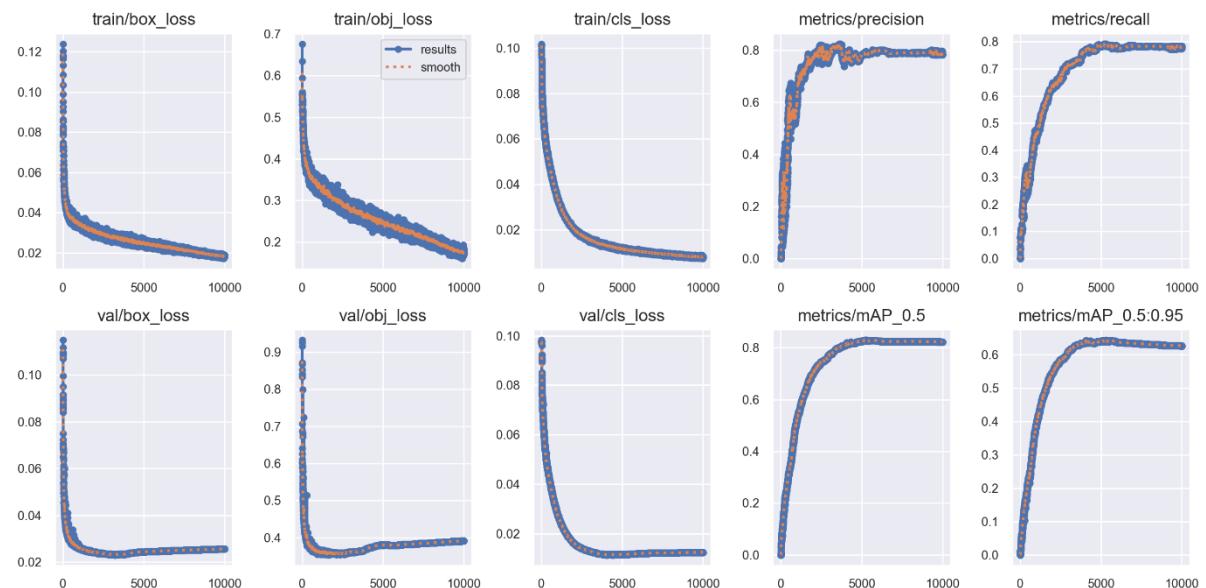
Lampiran 1 :

Hasil pelatihan model deteksi YOLOv5 pada berbagai skenario

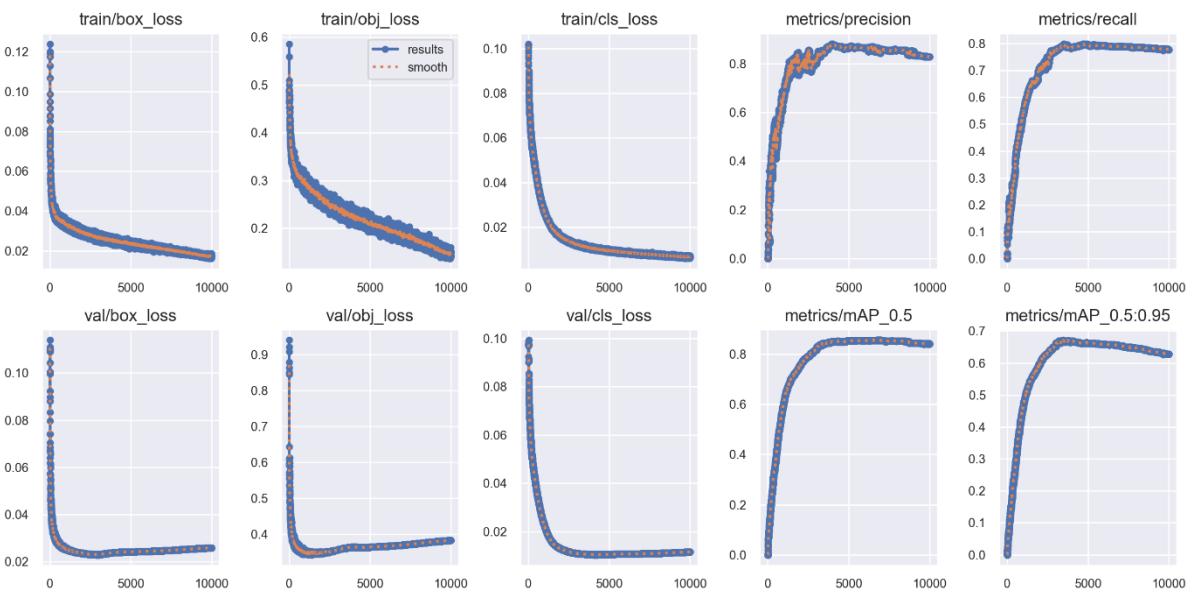
Skenario 1 (S1)



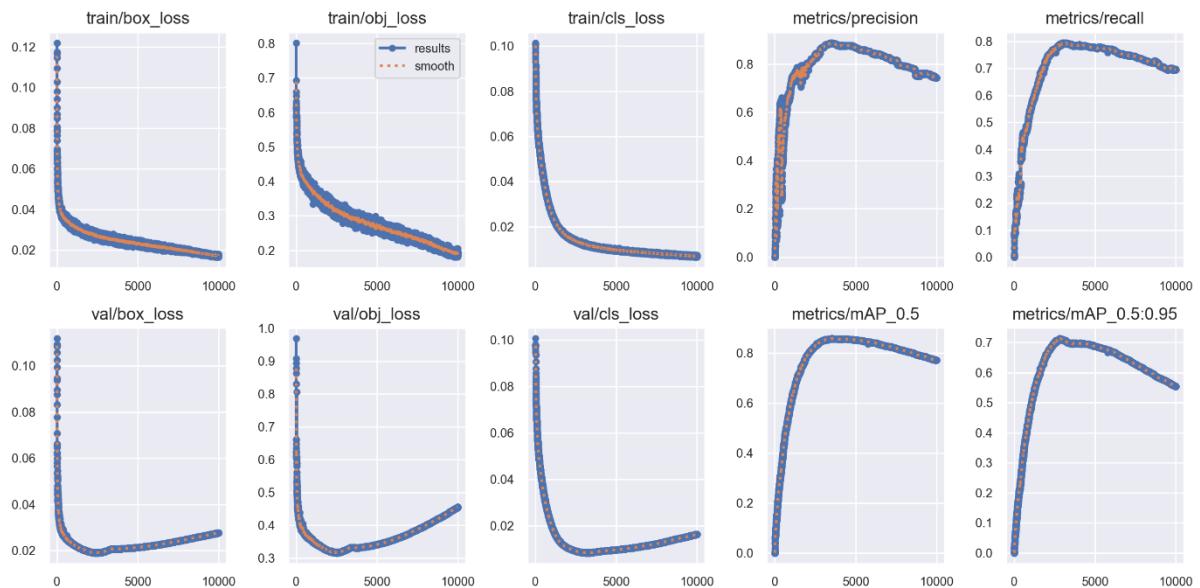
Skenario 2 (S2)



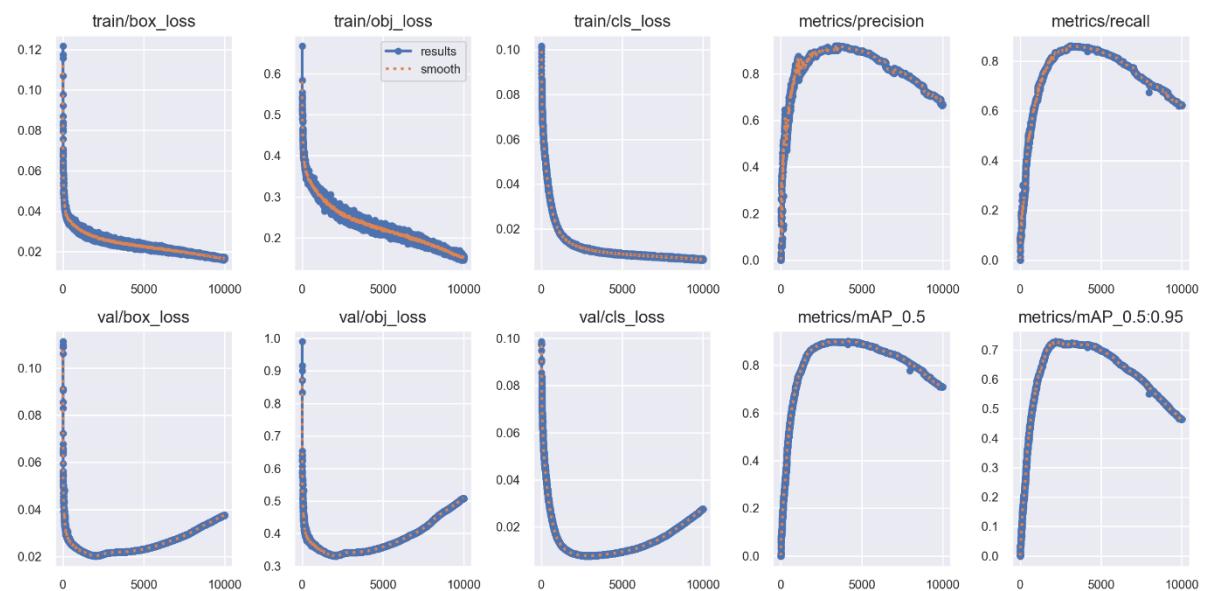
Skenario 3 (S3)



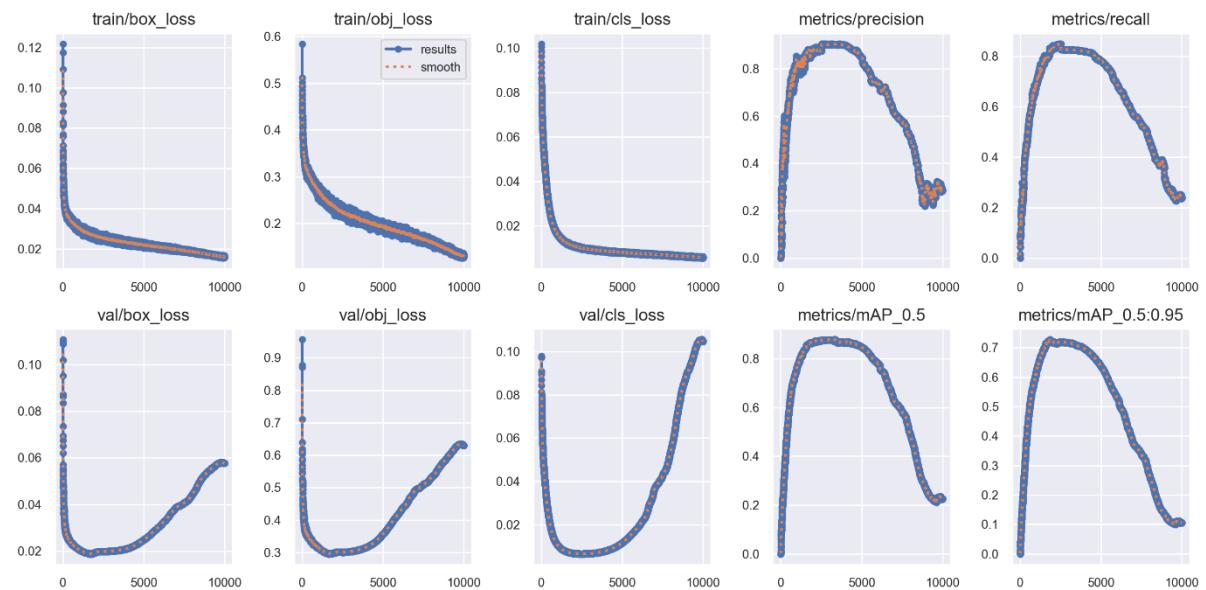
Skenario 4 (S4)



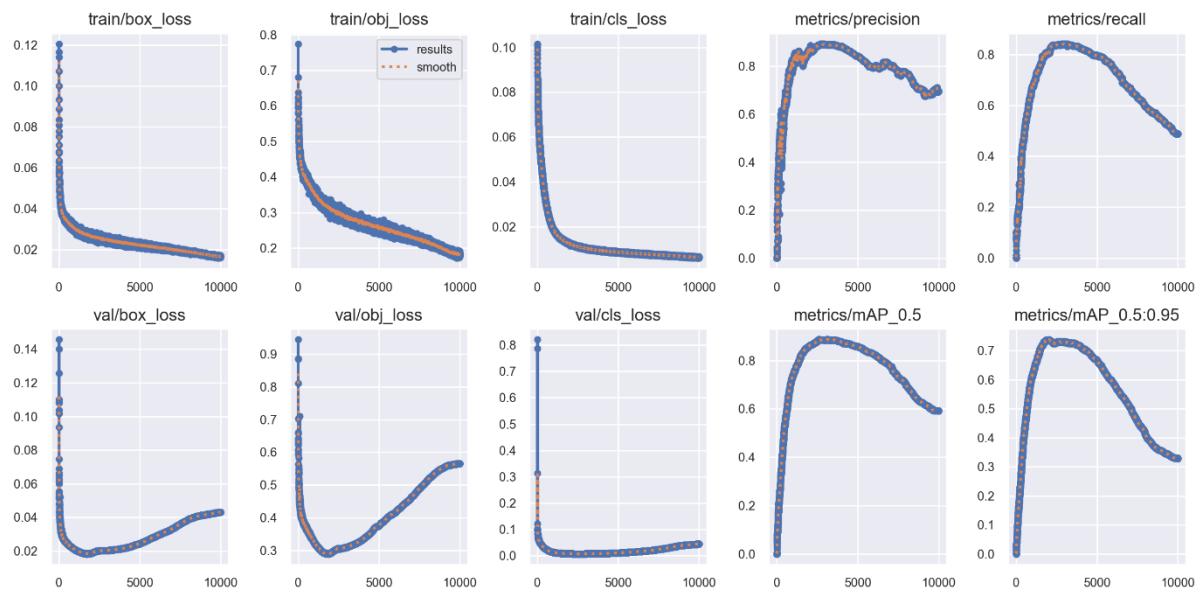
Skenario 5 (S5)



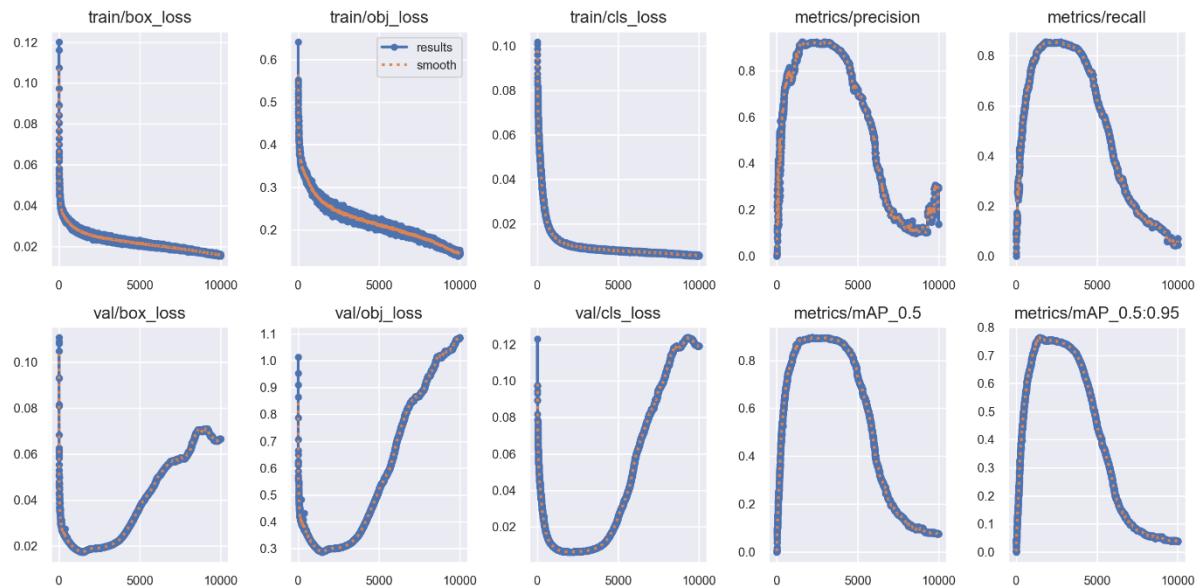
Skenario 6 (S6)



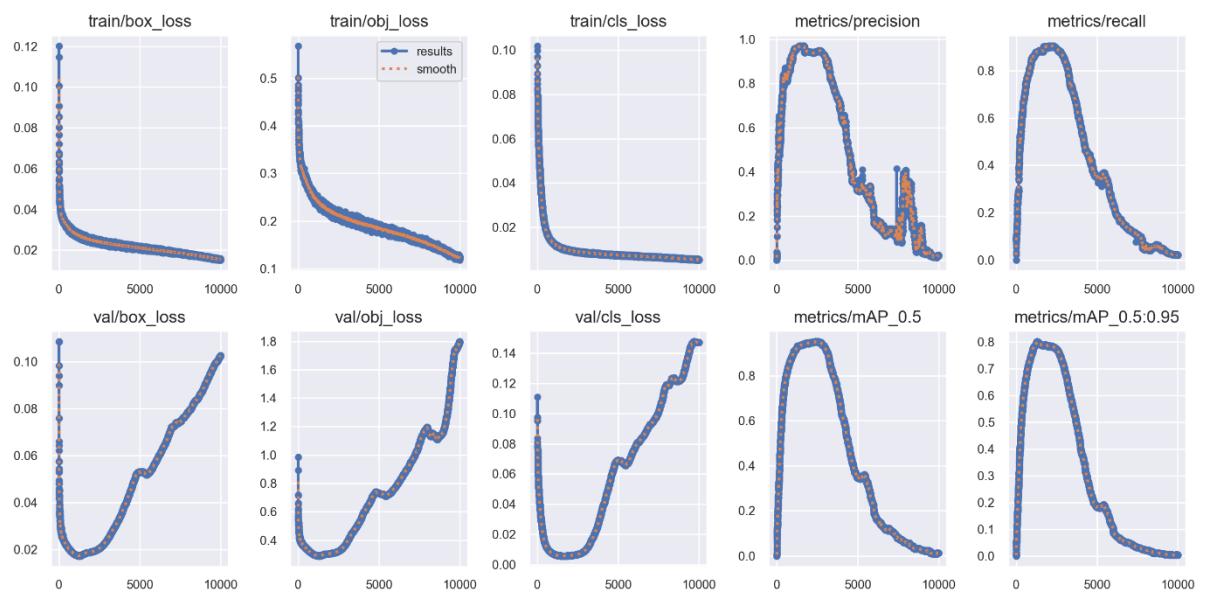
Skenario 7 (S7)



Skenario 8 (S8)



Skenario 9 (S9)



Lampiran 2 :

Rincian akurasi proses latih per kelas

Skenario 1 (S1)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	36	5785	0.848	0.72	0.793	0.608
1 Ha	36	199	0.87	0.915	0.937	0.777
10 La	36	172	0.909	0.942	0.953	0.753
11 Pa	36	139	0.804	0.899	0.879	0.668
12 Dha	36	24	1	0.0532	0.532	0.428
13 Ja	36	46	0.792	0.913	0.938	0.738
14 Ya	36	81	0.975	0.97	0.985	0.798
15 Nya	36	18	0.637	0.78	0.802	0.621
16 Ma	36	164	0.852	0.921	0.937	0.742
17 Ga	36	97	0.814	0.948	0.954	0.763
18 Ba	36	70	0.878	0.929	0.933	0.69
19 Tha	36	7	1	0.723	0.869	0.733
2 Na	36	448	0.829	0.951	0.957	0.785
20 Nga	36	86	0.779	0.821	0.842	0.615
21 Pasangan Ha	36	55	0.788	0.909	0.918	0.723
22 Pasangan Na	36	57	0.952	0.895	0.938	0.702
23 Pasangan Ca	36	7	0.611	0.857	0.835	0.661
24 Pasangan Ra	36	2	1	0	0.0765	0.0612
25 Pasangan Ka	36	42	0.579	0.905	0.869	0.622
26 Pasangan Da	36	38	0.958	0.921	0.948	0.715
27 Pasangan Ta	36	68	0.923	0.882	0.928	0.729
28 Pasangan Sa	36	57	0.734	0.93	0.899	0.68
29 Pasangan Wa	36	17	0.702	0.647	0.825	0.65
3 Ca	36	27	0.581	0.741	0.762	0.619
30 Pasangan La	36	36	0.816	0.247	0.725	0.483
31 Pasangan Pa	36	48	0.86	0.708	0.81	0.608
32 Pasangan Dha	36	15	1	0.737	0.958	0.786
33 Pasangan Ja	36	12	1	0.164	0.775	0.68
34 Pasangan Ya	36	4	1	0	0.0743	0.0662
35 Pasangan Nya	36	7	0.877	1	0.995	0.826
36 Pasangan Ma	36	30	0.769	0.967	0.914	0.736
37 Pasangan Ga	36	10	1	0	0.844	0.778
38 Pasangan Ba	36	16	0.937	0.688	0.773	0.605
39 Pasangan Tha	36	3	0.777	1	0.995	0.813
4 Ra	36	167	0.889	0.91	0.92	0.722
40 Pasangan Nga	36	11	1	0	0.235	0.187
41 Wulu	36	511	0.867	0.933	0.931	0.681
42 Pepet	36	180	0.811	0.95	0.937	0.721
43 Suku	36	383	0.95	0.953	0.968	0.599
44 Taling	36	361	0.919	0.975	0.973	0.798

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
45 Taling Tarung	36	128	0.944	0.928	0.929	0.494
46 Cecak	36	260	0.82	0.742	0.78	0.327
47 Layar	36	68	0.844	0.912	0.928	0.675
48 Pangkon	36	93	0.969	0.957	0.944	0.765
49 Pengkol	36	20	1	0.994	0.995	0.858
5 Ka	36	274	0.893	0.942	0.956	0.765
50 Wignyan	36	95	0.919	0.958	0.956	0.759
51 Cakra	36	56	0.829	0.929	0.915	0.64
52 Pa Cerek	36	31	0.49	0.387	0.515	0.416
53 Nga Lelet	36	16	0.401	0.375	0.417	0.318
54 Pada Lingsa	36	296	0.881	0.939	0.933	0.45
55 Pada Madya	36	35	0.795	0.971	0.952	0.735
56 Purwa Pada	36	1	1	0	0	0
57 Murda Na	36	10	0.867	0.4	0.752	0.566
59 Murda Ta	36	1	1	0	0	0
6 Da	36	89	0.811	0.854	0.875	0.704
60 Murda Sa	36	3	1	0	0.0123	0.00873
63 Murda Ga	36	19	0.643	1	0.969	0.728
67 Pasangan Murda Ga	36	5	1	0	0.266	0.156
7 Ta	36	160	0.794	0.925	0.916	0.73
71 Pasangan Murda Ta	36	22	0.768	1	0.995	0.701
8 Sa	36	238	0.748	0.979	0.978	0.804
9 Wa	36	134	0.762	0.94	0.925	0.727
95 Mahaprana Sha	36	16	0.81	0.535	0.735	0.59

Skenario 2 (S2)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	48	6253	0.852	0.793	0.843	0.644
1 Ha	48	234	0.815	0.953	0.972	0.83
10 La	48	163	0.898	0.982	0.986	0.795
11 Pa	48	138	0.73	0.877	0.829	0.638
12 Dha	48	17	0	0	0.195	0.16
13 Ja	48	54	0.839	0.981	0.985	0.79
14 Ya	48	105	0.949	0.981	0.986	0.793
15 Nya	48	16	0.624	0.938	0.926	0.648
16 Ma	48	180	0.932	0.994	0.992	0.818
17 Ga	48	84	0.91	0.869	0.951	0.768
18 Ba	48	74	0.958	0.946	0.976	0.75
19 Tha	48	11	1	0.504	0.926	0.734
2 Na	48	511	0.894	0.982	0.99	0.818
20 Nga	48	109	0.808	0.888	0.877	0.664
21 Pasangan Ha	48	63	0.888	0.873	0.934	0.758
22 Pasangan Na	48	59	0.951	0.949	0.976	0.72
23 Pasangan Ca	48	8	0.887	0.985	0.982	0.672
24 Pasangan Ra	48	3	1	0	0.27	0.217
25 Pasangan Ka	48	53	0.774	0.967	0.954	0.659
26 Pasangan Da	48	42	0.977	0.952	0.992	0.746
27 Pasangan Ta	48	62	0.7	0.977	0.97	0.75
28 Pasangan Sa	48	46	0.711	0.848	0.88	0.741
29 Pasangan Wa	48	29	0.824	0.897	0.952	0.801
3 Ca	48	28	0.728	0.765	0.801	0.638
30 Pasangan La	48	31	0.729	0.581	0.762	0.521
31 Pasangan Pa	48	48	0.773	0.708	0.814	0.629
32 Pasangan Dha	48	16	0.935	0.902	0.971	0.792
33 Pasangan Ja	48	8	0.612	1	0.757	0.664
34 Pasangan Ya	48	3	1	0	0.0543	0.0425
35 Pasangan Nya	48	14	0.95	1	0.995	0.811
36 Pasangan Ma	48	24	0.81	1	0.968	0.774
37 Pasangan Ga	48	7	1	0.839	0.995	0.773
38 Pasangan Ba	48	19	0.959	1	0.995	0.836
39 Pasangan Tha	48	4	0.446	1	0.674	0.58
4 Ra	48	141	0.943	0.935	0.977	0.788
40 Pasangan Nga	48	7	1	0	0.653	0.507
41 Wulu	48	539	0.907	0.957	0.972	0.686
42 Pepet	48	162	0.868	0.981	0.968	0.743
43 Suku	48	378	0.945	0.987	0.987	0.614
44 Taling	48	415	0.963	0.998	0.993	0.808
45 Taling Tarung	48	145	0.947	0.952	0.948	0.476
46 Cecak	48	299	0.859	0.791	0.831	0.336
47 Layar	48	89	0.898	0.865	0.926	0.702

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	48	84	0.861	1	0.991	0.738
49 Pengkol	48	18	0.811	1	0.995	0.832
5 Ka	48	303	0.888	0.983	0.988	0.802
50 Wignyan	48	105	0.934	0.99	0.982	0.807
51 Cakra	48	63	0.898	0.937	0.944	0.6
52 Pa Cerek	48	41	0.842	0.39	0.58	0.452
53 Nga Lelet	48	24	0.646	0.458	0.538	0.389
54 Pada Lingsa	48	328	0.93	0.966	0.955	0.483
55 Pada Madya	48	35	0.924	0.971	0.954	0.743
57 Murda Na	48	17	1	0.642	0.909	0.661
59 Murda Ta	48	2	1	0	0.0269	0.0236
6 Da	48	121	0.792	0.849	0.907	0.705
60 Murda Sa	48	1	1	0	0.0293	0.0293
63 Murda Ga	48	35	0.882	0.943	0.981	0.764
67 Pasangan Murda Ga	48	4	1	0	0.404	0.299
7 Ta	48	174	0.791	0.994	0.974	0.791
71 Pasangan Murda Ta	48	33	0.913	1	0.995	0.753
8 Sa	48	248	0.744	0.988	0.977	0.79
9 Wa	48	158	0.828	0.968	0.96	0.759
95 Mahaprana Sha	48	21	0.808	0.476	0.651	0.538

Skenario 3 (S3)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	60	6478	0.875	0.635	0.692	0.543
1 Ha	60	209	0.899	0.761	0.773	0.646
10 La	60	175	0.922	0.783	0.784	0.646
11 Pa	60	134	0.753	0.731	0.687	0.542
12 Dha	60	22	0.726	0.243	0.515	0.404
13 Ja	60	61	0.947	0.836	0.854	0.692
14 Ya	60	96	0.947	0.719	0.717	0.599
15 Nya	60	25	0.682	0.6	0.647	0.471
16 Ma	60	190	0.935	0.795	0.804	0.662
17 Ga	60	98	0.808	0.776	0.8	0.646
18 Ba	60	76	0.868	0.763	0.741	0.614
19 Tha	60	6	0.911	1	0.995	0.784
2 Na	60	531	0.943	0.786	0.801	0.66
20 Nga	60	107	0.84	0.636	0.675	0.523
21 Pasangan Ha	60	61	0.977	0.703	0.743	0.622
22 Pasangan Na	60	65	0.954	0.754	0.768	0.6
23 Pasangan Ca	60	8	0.66	0.875	0.708	0.572
24 Pasangan Ra	60	4	1	0	0.443	0.361
25 Pasangan Ka	60	56	0.809	0.75	0.753	0.543
26 Pasangan Da	60	37	0.964	0.724	0.741	0.595
27 Pasangan Ta	60	59	0.777	0.847	0.84	0.664
28 Pasangan Sa	60	64	0.867	0.719	0.73	0.605
29 Pasangan Wa	60	22	0.809	0.727	0.763	0.634
3 Ca	60	17	0.497	0.765	0.68	0.561
30 Pasangan La	60	27	0.926	0.466	0.613	0.451
31 Pasangan Pa	60	37	0.917	0.593	0.676	0.55
32 Pasangan Dha	60	13	0.837	0.769	0.833	0.726
33 Pasangan Ja	60	17	0.891	0.471	0.685	0.544
34 Pasangan Ya	60	2	1	0	0.0463	0.0417
35 Pasangan Nya	60	11	0.908	0.727	0.725	0.651
36 Pasangan Ma	60	17	0.615	0.588	0.586	0.482
37 Pasangan Ga	60	10	0.752	0.5	0.679	0.528
38 Pasangan Ba	60	24	0.928	0.833	0.861	0.725
39 Pasangan Tha	60	12	0.902	0.769	0.827	0.701
4 Ra	60	175	0.941	0.84	0.85	0.702
40 Pasangan Nga	60	11	1	0	0.453	0.366
41 Wulu	60	538	0.965	0.784	0.8	0.587
42 Pepet	60	173	0.888	0.78	0.789	0.614
43 Suku	60	423	0.957	0.804	0.817	0.537
44 Taling	60	421	0.961	0.753	0.767	0.659
45 Taling Tarung	60	133	0.922	0.684	0.679	0.36
46 Cecak	60	317	0.872	0.688	0.695	0.283
47 Layar	60	88	0.863	0.739	0.732	0.564

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	60	104	0.978	0.75	0.753	0.601
49 Pengkol	60	17	0.719	0.706	0.665	0.585
5 Ka	60	304	0.897	0.742	0.75	0.615
50 Wignyan	60	118	0.963	0.797	0.797	0.663
51 Cakra	60	60	0.989	0.8	0.796	0.558
52 Pa Cerek	60	45	0.739	0.441	0.515	0.393
53 Nga Lelet	60	19	0.461	0.263	0.297	0.227
54 Pada Lingsa	60	344	0.93	0.756	0.766	0.383
55 Pada Madya	60	36	0.928	0.75	0.746	0.658
57 Murda Na	60	14	0.883	0.54	0.68	0.481
58 Murda Ka	60	1	1	0	0	0
6 Da	60	128	0.943	0.805	0.813	0.65
60 Murda Sa	60	4	1	0	0.31	0.198
61 Murda Pa	60	1	1	0	0	0
63 Murda Ga	60	20	0.786	0.9	0.893	0.66
67 Pasangan Murda Ga	60	3	1	0	0.83	0.614
7 Ta	60	221	0.939	0.801	0.804	0.663
71 Pasangan Murda Ta	60	23	0.861	0.957	0.955	0.772
8 Sa	60	233	0.839	0.785	0.777	0.648
9 Wa	60	187	0.912	0.759	0.778	0.626
95 Mahaprana Sha	60	24	0.821	0.375	0.57	0.494

Skenario 4 (S4)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	72	11641	0.925	0.66	0.717	0.598
1 Ha	72	418	0.898	0.828	0.844	0.73
10 La	72	299	0.946	0.88	0.887	0.761
11 Pa	72	258	0.866	0.848	0.886	0.72
12 Dha	72	39	1	0	0.371	0.313
13 Ja	72	81	0.932	0.691	0.714	0.574
14 Ya	72	194	0.956	0.778	0.778	0.661
15 Nya	72	29	0.767	0.793	0.786	0.628
16 Ma	72	361	0.966	0.798	0.818	0.71
17 Ga	72	169	0.898	0.805	0.802	0.685
18 Ba	72	117	0.914	0.829	0.838	0.708
19 Tha	72	19	1	0.524	0.734	0.633
2 Na	72	945	0.951	0.82	0.836	0.731
20 Nga	72	169	0.866	0.817	0.856	0.688
21 Pasangan Ha	72	111	0.977	0.765	0.793	0.689
22 Pasangan Na	72	135	0.969	0.881	0.878	0.729
23 Pasangan Ca	72	17	0.95	0.824	0.825	0.744
24 Pasangan Ra	72	4	1	0	0.305	0.266
25 Pasangan Ka	72	83	0.846	0.795	0.823	0.643
26 Pasangan Da	72	79	0.956	0.873	0.875	0.752
27 Pasangan Ta	72	91	0.893	0.828	0.834	0.679
28 Pasangan Sa	72	112	0.892	0.741	0.753	0.624
29 Pasangan Wa	72	37	0.924	0.811	0.816	0.686
3 Ca	72	53	0.766	0.66	0.724	0.613
30 Pasangan La	72	35	0.835	0.657	0.707	0.556
31 Pasangan Pa	72	66	0.898	0.606	0.696	0.581
32 Pasangan Dha	72	15	0.892	0.552	0.784	0.684
33 Pasangan Ja	72	15	0.996	0.533	0.59	0.519
34 Pasangan Ya	72	5	1	0	0.0827	0.0735
35 Pasangan Nya	72	24	0.946	0.792	0.796	0.71
36 Pasangan Ma	72	46	0.795	0.675	0.681	0.613
37 Pasangan Ga	72	27	1	0	0.63	0.529
38 Pasangan Ba	72	30	0.978	0.7	0.696	0.614
39 Pasangan Tha	72	7	0.56	0.857	0.713	0.597
4 Ra	72	303	0.939	0.825	0.827	0.702
40 Pasangan Nga	72	9	1	0	0.332	0.274
41 Wulu	72	1044	0.965	0.811	0.831	0.648
42 Pepet	72	328	0.921	0.79	0.798	0.669
43 Suku	72	753	0.978	0.838	0.847	0.669
44 Taling	72	760	0.971	0.845	0.853	0.753
45 Taling Tarung	72	237	0.975	0.873	0.878	0.544
46 Cecak	72	562	0.944	0.782	0.808	0.418
47 Layar	72	164	0.955	0.787	0.79	0.634

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	72	168	0.941	0.911	0.899	0.767
49 Pengkol	72	30	0.868	0.833	0.833	0.753
5 Ka	72	539	0.959	0.844	0.849	0.742
50 Wignyan	72	200	0.974	0.86	0.856	0.753
51 Cakra	72	114	0.968	0.868	0.862	0.684
52 Pa Cerek	72	84	0.89	0.607	0.695	0.567
53 Nga Lelet	72	50	0.934	0.567	0.7	0.599
54 Pada Lingsa	72	626	0.973	0.84	0.848	0.541
55 Pada Madya	72	68	0.954	0.838	0.836	0.769
56 Purwa Pada	72	1	0.727	1	0.995	0.895
57 Murda Na	72	13	0.851	0.385	0.674	0.592
58 Murda Ka	72	2	1	0	0.00202	0.00161
59 Murda Ta	72	3	1	0	0.00122	0.000976
6 Da	72	204	0.901	0.843	0.861	0.743
60 Murda Sa	72	6	1	0.487	0.576	0.497
61 Murda Pa	72	2	1	0	0.0622	0.056
63 Murda Ga	72	47	0.974	0.681	0.686	0.568
67 Pasangan Murda Ga	72	6	1	0	0.269	0.198
7 Ta	72	390	0.944	0.797	0.823	0.707
71 Pasangan Murda Ta	72	53	0.97	0.717	0.716	0.618
8 Sa	72	445	0.882	0.827	0.837	0.714
9 Wa	72	304	0.86	0.859	0.867	0.735
95 Mahaprana Sha	72	36	0.877	0.611	0.742	0.589

Skenario 5 (S5)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	96	12042	0.917	0.661	0.716	0.589
1 Ha	96	424	0.934	0.769	0.795	0.688
10 La	96	384	0.936	0.81	0.818	0.707
11 Pa	96	269	0.771	0.647	0.632	0.521
12 Dha	96	44	1	0.0668	0.682	0.598
13 Ja	96	106	0.921	0.764	0.766	0.65
14 Ya	96	155	0.955	0.839	0.839	0.722
15 Nya	96	39	0.801	0.824	0.854	0.717
16 Ma	96	400	0.961	0.78	0.79	0.667
17 Ga	96	207	0.925	0.836	0.836	0.703
18 Ba	96	119	0.896	0.899	0.892	0.756
19 Tha	96	5	0.807	0.4	0.395	0.356
2 Na	96	942	0.968	0.8	0.814	0.694
20 Nga	96	179	0.755	0.67	0.641	0.533
21 Pasangan Ha	96	102	0.865	0.794	0.807	0.683
22 Pasangan Na	96	119	0.956	0.79	0.787	0.674
23 Pasangan Ca	96	17	0.971	0.588	0.585	0.498
24 Pasangan Ra	96	5	1	0	0.069	0.0485
25 Pasangan Ka	96	82	0.83	0.744	0.737	0.592
26 Pasangan Da	96	76	0.908	0.855	0.851	0.724
27 Pasangan Ta	96	116	0.81	0.808	0.802	0.656
28 Pasangan Sa	96	106	0.85	0.692	0.685	0.556
29 Pasangan Wa	96	62	0.921	0.806	0.805	0.681
3 Ca	96	39	0.499	0.641	0.634	0.559
30 Pasangan La	96	65	0.946	0.769	0.861	0.636
31 Pasangan Pa	96	79	0.852	0.729	0.832	0.672
32 Pasangan Dha	96	26	0.962	0.615	0.671	0.618
33 Pasangan Ja	96	22	1	0.719	0.955	0.832
34 Pasangan Ya	96	7	1	0	0.137	0.124
35 Pasangan Nya	96	14	0.919	0.714	0.715	0.656
36 Pasangan Ma	96	54	0.905	0.759	0.753	0.664
37 Pasangan Ga	96	16	1	0.67	0.891	0.744
38 Pasangan Ba	96	54	0.874	0.63	0.625	0.506
39 Pasangan Tha	96	10	1	0.594	0.595	0.542
4 Ra	96	279	0.946	0.781	0.784	0.667
40 Pasangan Nga	96	11	1	0	0.676	0.541
41 Wulu	96	1033	0.941	0.805	0.816	0.618
42 Pepet	96	328	0.949	0.78	0.789	0.668
43 Suku	96	774	0.981	0.771	0.782	0.582
44 Taling	96	773	0.968	0.816	0.831	0.73
45 Taling Tarung	96	253	0.96	0.802	0.808	0.478
46 Cecak	96	584	0.919	0.726	0.754	0.382
47 Layar	96	185	0.918	0.726	0.748	0.607

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	96	191	0.987	0.796	0.804	0.696
49 Pengkol	96	33	0.97	0.788	0.786	0.722
5 Ka	96	575	0.947	0.798	0.811	0.699
50 Wignyan	96	214	0.979	0.827	0.825	0.726
51 Cakra	96	78	0.947	0.846	0.846	0.673
52 Pa Cerek	96	86	0.829	0.506	0.642	0.507
53 Nga Lelet	96	58	0.845	0.469	0.727	0.572
54 Pada Lingsa	96	631	0.965	0.794	0.811	0.492
55 Pada Madya	96	60	0.974	0.8	0.796	0.71
56 Purwa Pada	96	2	0.831	1	0.995	0.697
57 Murda Na	96	27	0.921	0.407	0.527	0.425
59 Murda Ta	96	3	1	0	0.0561	0.0454
6 Da	96	202	0.912	0.797	0.807	0.682
60 Murda Sa	96	5	1	0	0.131	0.11
63 Murda Ga	96	55	0.876	0.855	0.856	0.698
67 Pasangan Murda Ga	96	7	1	0	0.124	0.0781
7 Ta	96	344	0.905	0.831	0.827	0.718
71 Pasangan Murda Ta	96	62	0.953	0.839	0.844	0.694
8 Sa	96	482	0.877	0.803	0.811	0.673
9 Wa	96	320	0.881	0.838	0.844	0.694
95 Mahaprana Sha	96	43	0.876	0.395	0.675	0.578

Skenario 6 (S6)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	120	12895	0.904	0.73	0.763	0.635
1 Ha	120	456	0.888	0.864	0.875	0.772
10 La	120	368	0.96	0.913	0.917	0.807
11 Pa	120	292	0.825	0.822	0.808	0.671
12 Dha	120	40	0.819	0.85	0.802	0.717
13 Ja	120	91	0.862	0.813	0.824	0.706
14 Ya	120	193	0.995	0.891	0.896	0.78
15 Nya	120	41	0.672	0.927	0.926	0.729
16 Ma	120	379	0.927	0.913	0.917	0.804
17 Ga	120	191	0.96	0.853	0.859	0.74
18 Ba	120	137	0.911	0.854	0.837	0.713
19 Tha	120	21	0.915	0.81	0.805	0.68
2 Na	120	1043	0.928	0.877	0.883	0.772
20 Nga	120	221	0.88	0.814	0.835	0.683
21 Pasangan Ha	120	119	0.934	0.798	0.858	0.742
22 Pasangan Na	120	131	0.979	0.832	0.831	0.713
23 Pasangan Ca	120	18	0.864	1	0.995	0.852
24 Pasangan Ra	120	6	1	0	0.153	0.106
25 Pasangan Ka	120	99	0.768	0.848	0.834	0.701
26 Pasangan Da	120	101	0.847	0.901	0.91	0.77
27 Pasangan Ta	120	141	0.772	0.908	0.911	0.742
28 Pasangan Sa	120	111	0.838	0.838	0.835	0.716
29 Pasangan Wa	120	52	0.936	0.885	0.885	0.795
3 Ca	120	53	0.624	0.717	0.737	0.633
30 Pasangan La	120	65	0.883	0.846	0.901	0.708
31 Pasangan Pa	120	94	0.876	0.851	0.87	0.725
32 Pasangan Dha	120	21	0.791	0.905	0.9	0.817
33 Pasangan Ja	120	24	1	0.858	0.915	0.829
34 Pasangan Ya	120	5	1	0	0.0305	0.0254
35 Pasangan Nya	120	27	0.948	0.889	0.885	0.78
36 Pasangan Ma	120	59	0.893	0.949	0.945	0.822
37 Pasangan Ga	120	20	0.936	0.729	0.825	0.703
38 Pasangan Ba	120	37	0.898	0.838	0.849	0.714
39 Pasangan Tha	120	13	0.833	0.846	0.845	0.759
4 Ra	120	325	0.931	0.886	0.894	0.768
40 Pasangan Nga	120	17	1	0.217	0.715	0.533
41 Wulu	120	1105	0.916	0.861	0.87	0.674
42 Pepet	120	337	0.947	0.828	0.837	0.704
43 Suku	120	817	0.967	0.902	0.908	0.712
44 Taling	120	792	0.956	0.88	0.889	0.788
45 Taling Tarung	120	258	0.958	0.857	0.865	0.54
46 Cecak	120	642	0.915	0.798	0.814	0.42
47 Layar	120	199	0.911	0.859	0.869	0.695

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	120	204	0.992	0.877	0.877	0.765
49 Pengkol	120	39	0.899	0.923	0.902	0.805
5 Ka	120	626	0.94	0.866	0.875	0.77
50 Wignyan	120	226	0.95	0.881	0.886	0.783
51 Cakra	120	112	0.922	0.902	0.903	0.668
52 Pa Cerek	120	83	0.787	0.735	0.785	0.657
53 Nga Lelet	120	43	0.676	0.674	0.702	0.607
54 Pada Lingsa	120	682	0.941	0.856	0.848	0.525
55 Pada Madya	120	75	0.956	0.867	0.866	0.761
56 Purwa Pada	120	3	1	0	0.665	0.349
57 Murda Na	120	25	0.875	0.56	0.751	0.622
58 Murda Ka	120	1	1	0	0.000238	0.000166
59 Murda Ta	120	2	1	0	0.00673	0.00577
6 Da	120	221	0.904	0.869	0.876	0.767
60 Murda Sa	120	10	0.84	0.2	0.218	0.179
61 Murda Pa	120	1	1	0	0.0085	0.0051
63 Murda Ga	120	49	0.946	0.918	0.915	0.719
64 Murda Ba	120	2	1	0	0	0
67 Pasangan Murda Ga	120	3	1	0	0.312	0.249
7 Ta	120	429	0.926	0.916	0.924	0.813
71 Pasangan Murda Ta	120	46	0.979	0.913	0.915	0.743
8 Sa	120	490	0.907	0.863	0.866	0.753
9 Wa	120	328	0.959	0.845	0.856	0.74
95 Mahaprana Sha	120	34	0.597	0.765	0.67	0.582

Skenario 7 (S7)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	108	17656	0.916	0.734	0.797	0.677
1 Ha	108	637	0.932	0.881	0.888	0.783
10 La	108	499	0.972	0.878	0.884	0.776
11 Pa	108	405	0.876	0.844	0.858	0.714
12 Dha	108	49	0.857	0.367	0.664	0.572
13 Ja	108	122	0.936	0.836	0.843	0.7
14 Ya	108	265	0.992	0.89	0.896	0.788
15 Nya	108	56	0.697	0.911	0.884	0.771
16 Ma	108	500	0.964	0.89	0.895	0.773
17 Ga	108	271	0.939	0.845	0.847	0.745
18 Ba	108	190	0.935	0.879	0.884	0.767
19 Tha	108	21	0.869	0.905	0.905	0.775
2 Na	108	1438	0.948	0.867	0.873	0.763
20 Nga	108	278	0.877	0.802	0.836	0.707
21 Pasangan Ha	108	147	0.941	0.875	0.885	0.767
22 Pasangan Na	108	190	0.966	0.847	0.847	0.744
23 Pasangan Ca	108	30	0.963	0.933	0.935	0.836
24 Pasangan Ra	108	12	1	0	0.132	0.11
25 Pasangan Ka	108	142	0.742	0.803	0.791	0.645
26 Pasangan Da	108	111	0.948	0.901	0.905	0.793
27 Pasangan Ta	108	163	0.879	0.822	0.837	0.716
28 Pasangan Sa	108	136	0.825	0.831	0.826	0.695
29 Pasangan Wa	108	84	0.95	0.821	0.834	0.732
3 Ca	108	82	0.807	0.817	0.828	0.698
30 Pasangan La	108	77	0.684	0.225	0.584	0.456
31 Pasangan Pa	108	123	0.958	0.772	0.872	0.736
32 Pasangan Dha	108	27	0.757	0.852	0.916	0.815
33 Pasangan Ja	108	30	0.965	0.928	0.959	0.818
34 Pasangan Ya	108	9	1	0	0.0448	0.0382
35 Pasangan Nya	108	29	0.965	0.931	0.935	0.852
36 Pasangan Ma	108	65	0.955	0.969	0.965	0.84
37 Pasangan Ga	108	19	0.755	0.684	0.865	0.751
38 Pasangan Ba	108	58	0.982	0.81	0.826	0.709
39 Pasangan Tha	108	25	0.962	1	0.995	0.875
4 Ra	108	450	0.922	0.858	0.853	0.739
40 Pasangan Nga	108	15	1	0	0.78	0.641
41 Wulu	108	1554	0.958	0.896	0.901	0.72
42 Pepet	108	508	0.948	0.87	0.878	0.761
43 Suku	108	1160	0.981	0.87	0.871	0.734
44 Taling	108	1114	0.965	0.905	0.91	0.819
45 Taling Tarung	108	376	0.953	0.886	0.891	0.612
46 Cecak	108	878	0.945	0.843	0.852	0.508
47 Layar	108	271	0.947	0.856	0.874	0.707

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	108	277	0.982	0.874	0.877	0.763
49 Pengkol	108	52	0.966	0.846	0.846	0.754
5 Ka	108	793	0.953	0.87	0.879	0.776
50 Wigyan	108	291	0.969	0.907	0.907	0.809
51 Cakra	108	151	0.965	0.881	0.885	0.722
52 Pa Cerek	108	99	0.86	0.889	0.906	0.728
53 Nga Lelet	108	66	0.929	0.598	0.715	0.595
54 Pada Lingsa	108	931	0.975	0.879	0.889	0.631
55 Pada Madya	108	107	0.974	0.907	0.906	0.838
56 Purwa Pada	108	3	0.866	1	0.995	0.852
57 Murda Na	108	28	0.359	0.18	0.464	0.404
58 Murda Ka	108	1	1	0	0.166	0.149
6 Da	108	335	0.894	0.857	0.863	0.746
60 Murda Sa	108	6	0.837	0.333	0.382	0.359
61 Murda Pa	108	1	1	0	0.0108	0.00865
63 Murda Ga	108	77	0.987	0.922	0.925	0.774
64 Murda Ba	108	1	1	0	0.497	0.398
67 Pasangan Murda Ga	108	10	1	0	0.524	0.421
7 Ta	108	539	0.92	0.879	0.884	0.78
71 Pasangan Murda Ta	108	83	0.957	0.928	0.926	0.771
8 Sa	108	682	0.928	0.919	0.925	0.799
9 Wa	108	445	0.879	0.885	0.89	0.761
95 Mahaprana Sha	108	62	0.944	0.544	0.774	0.67

Skenario 8 (S8)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	144	18821	0.953	0.684	0.732	0.63
1 Ha	144	630	0.966	0.84	0.856	0.775
10 La	144	556	0.96	0.858	0.858	0.76
11 Pa	144	381	0.938	0.789	0.815	0.708
12 Dha	144	57	0.958	0.281	0.748	0.686
13 Ja	144	156	1	0.755	0.757	0.65
14 Ya	144	267	0.987	0.846	0.848	0.761
15 Nya	144	66	0.822	0.727	0.732	0.628
16 Ma	144	614	0.958	0.834	0.839	0.739
17 Ga	144	259	0.919	0.819	0.825	0.743
18 Ba	144	227	0.951	0.877	0.885	0.778
19 Tha	144	22	0.994	0.909	0.905	0.758
2 Na	144	1550	0.966	0.832	0.844	0.752
20 Nga	144	277	0.922	0.729	0.754	0.639
21 Pasangan Ha	144	175	0.936	0.838	0.856	0.76
22 Pasangan Na	144	208	0.984	0.865	0.867	0.766
23 Pasangan Ca	144	40	0.973	0.55	0.567	0.527
24 Pasangan Ra	144	11	1	0	0.0284	0.0212
25 Pasangan Ka	144	137	0.834	0.854	0.857	0.722
26 Pasangan Da	144	100	0.972	0.83	0.845	0.737
27 Pasangan Ta	144	201	0.845	0.866	0.862	0.733
28 Pasangan Sa	144	142	0.822	0.782	0.785	0.663
29 Pasangan Wa	144	83	0.951	0.795	0.799	0.705
3 Ca	144	88	0.879	0.773	0.807	0.715
30 Pasangan La	144	100	0.903	0.65	0.75	0.607
31 Pasangan Pa	144	128	0.973	0.812	0.872	0.733
32 Pasangan Dha	144	39	1	0.75	0.851	0.734
33 Pasangan Ja	144	27	0.99	0.815	0.854	0.753
34 Pasangan Ya	144	7	1	0	0.0216	0.0181
35 Pasangan Nya	144	33	0.974	0.818	0.815	0.75
36 Pasangan Ma	144	86	0.978	0.791	0.801	0.725
37 Pasangan Ga	144	22	1	0.776	0.865	0.771
38 Pasangan Ba	144	73	0.979	0.795	0.796	0.701
39 Pasangan Tha	144	18	0.925	0.611	0.615	0.545
4 Ra	144	508	0.941	0.815	0.817	0.71
40 Pasangan Nga	144	26	1	0	0.411	0.329
41 Wulu	144	1566	0.966	0.837	0.85	0.69
42 Pepet	144	494	0.969	0.798	0.801	0.7
43 Suku	144	1184	0.983	0.814	0.823	0.704
44 Taling	144	1200	0.971	0.838	0.847	0.771
45 Taling Tarung	144	420	0.967	0.828	0.84	0.56
46 Cecak	144	903	0.958	0.784	0.795	0.467
47 Layar	144	266	0.972	0.82	0.826	0.686

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	144	264	0.991	0.841	0.847	0.77
49 Pengkol	144	65	0.956	0.723	0.737	0.677
5 Ka	144	905	0.978	0.825	0.831	0.752
50 Wigyan	144	323	0.983	0.799	0.798	0.726
51 Cakra	144	163	0.965	0.865	0.869	0.707
52 Pa Cerek	144	124	0.953	0.823	0.876	0.739
53 Nga Lelet	144	71	0.909	0.831	0.877	0.751
54 Pada Lingsa	144	977	0.975	0.83	0.84	0.595
55 Pada Madya	144	111	0.977	0.847	0.846	0.797
56 Purwa Pada	144	1	1	0	0	0
57 Murda Na	144	39	0.964	0.688	0.831	0.689
58 Murda Ka	144	1	1	0	0.00075	0.0006
59 Murda Ta	144	3	1	0	0.683	0.565
6 Da	144	368	0.951	0.823	0.826	0.721
60 Murda Sa	144	6	0.763	0.167	0.253	0.224
61 Murda Pa	144	1	1	0	0.00060 3	0.000603
63 Murda Ga	144	71	0.953	0.831	0.835	0.72
67 Pasangan Murda Ga	144	11	1	0	0.398	0.314
7 Ta	144	620	0.949	0.835	0.84	0.755
71 Pasangan Murda Ta	144	76	0.931	0.829	0.826	0.731
8 Sa	144	753	0.928	0.814	0.819	0.722
9 Wa	144	465	0.938	0.84	0.847	0.739
95 Mahaprana Sha	144	56	0.901	0.651	0.695	0.603

Skenario 9 (S9)

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
all	179	19199	0.961	0.821	0.867	0.741
1 Ha	179	638	0.967	0.904	0.907	0.81
10 La	179	532	0.983	0.921	0.929	0.82
11 Pa	179	457	0.941	0.838	0.864	0.746
12 Dha	179	57	1	0.255	0.778	0.675
13 Ja	179	137	0.972	0.876	0.875	0.75
14 Ya	179	283	0.983	0.936	0.936	0.814
15 Nya	179	58	0.984	0.879	0.954	0.813
16 Ma	179	608	0.979	0.913	0.917	0.79
17 Ga	179	339	0.95	0.926	0.934	0.818
18 Ba	179	220	0.96	0.959	0.955	0.83
19 Tha	179	33	0.981	0.909	0.905	0.782
2 Na	179	1523	0.981	0.929	0.937	0.828
20 Nga	179	298	0.971	0.866	0.889	0.764
21 Pasangan Ha	179	179	0.975	0.916	0.925	0.815
22 Pasangan Na	179	229	0.983	0.921	0.926	0.814
23 Pasangan Ca	179	29	0.894	0.966	0.895	0.797
24 Pasangan Ra	179	11	1	0	0.194	0.167
25 Pasangan Ka	179	130	0.935	0.881	0.925	0.756
26 Pasangan Da	179	115	0.97	0.887	0.911	0.805
27 Pasangan Ta	179	196	0.897	0.893	0.894	0.726
28 Pasangan Sa	179	160	0.934	0.906	0.906	0.76
29 Pasangan Wa	179	75	0.987	0.947	0.976	0.866
3 Ca	179	74	0.849	0.838	0.84	0.741
30 Pasangan La	179	105	0.956	0.829	0.907	0.739
31 Pasangan Pa	179	142	0.92	0.831	0.871	0.744
32 Pasangan Dha	179	33	0.986	1	0.995	0.892
33 Pasangan Ja	179	18	1	0.902	0.99	0.881
34 Pasangan Ya	179	8	1	0	0.0453	0.038
35 Pasangan Nya	179	38	0.98	0.895	0.895	0.84
36 Pasangan Ma	179	94	0.959	0.904	0.938	0.835
37 Pasangan Ga	179	31	0.929	0.849	0.927	0.755
38 Pasangan Ba	179	63	0.948	0.841	0.845	0.741
39 Pasangan Tha	179	22	0.796	0.889	0.887	0.802
4 Ra	179	487	0.962	0.899	0.902	0.774
40 Pasangan Nga	179	33	1	0	0.557	0.458
41 Wulu	179	1644	0.975	0.906	0.917	0.727
42 Pepet	179	564	0.985	0.917	0.921	0.795
43 Suku	179	1181	0.982	0.91	0.916	0.754
44 Taling	179	1202	0.982	0.921	0.927	0.835
45 Taling Tarung	179	402	0.966	0.91	0.92	0.611
46 Cecak	179	966	0.966	0.842	0.864	0.485
47 Layar	179	263	0.939	0.878	0.898	0.729

Class	Images	Instances	Precision	Recall	mAP0,5	mAP0,5-0,95
48 Pangkon	179	301	0.995	0.93	0.936	0.835
49 Pengkol	179	56	0.968	1	0.99	0.902
5 Ka	179	925	0.978	0.911	0.918	0.82
50 Wignyan	179	332	0.986	0.904	0.906	0.814
51 Cakra	179	146	0.992	0.932	0.935	0.75
52 Pa Cerek	179	121	0.962	0.846	0.864	0.732
53 Nga Lelet	179	65	0.928	0.892	0.904	0.781
54 Pada Lingsa	179	1008	0.983	0.909	0.915	0.605
55 Pada Madya	179	109	0.982	0.89	0.886	0.823
56 Purwa Pada	179	3	0.89	1	0.995	0.84
57 Murda Na	179	34	0.971	0.983	0.988	0.855
6 Da	179	387	0.935	0.935	0.947	0.826
60 Murda Sa	179	3	1	0	0.354	0.319
63 Murda Ga	179	75	0.902	0.92	0.911	0.794
67 Pasangan Murda Ga	179	13	1	0	0.388	0.322
7 Ta	179	612	0.965	0.933	0.936	0.831
71 Pasangan Murda Ta	179	81	0.991	0.926	0.925	0.786
8 Sa	179	724	0.949	0.919	0.917	0.805
9 Wa	179	468	0.958	0.927	0.933	0.801
95 Mahaprana Sha	179	59	0.957	0.76	0.888	0.763

Halaman ini sengaja dikosongkan

BIODATA PENULIS



Penulis dilahirkan di Gresik, 24 Januari 2001, merupakan anak tunggal. Penulis telah menempuh pendidikan formal yaitu di TK Al Falah Sembayat Manyar Gresik, MI Al Falah Sembayat Manyar Gresik, MTs. Maarif NU Assa'adah I Sampurnan Bungah dan SMAN 1 Gresik. Setelah lulus dari SMAN tahun 2019, Penulis mengikuti SNMPTN dan diterima di Departemen Teknik Informatika FTEIC - ITS pada tahun 2019 dan terdaftar dengan NRP 05111940000060.

Selama masa kuliah, penulis sempat aktif di beberapa organisasi seperti UKM Rebana-ITS, LKKI Informatika KMI, Jama'ah Masjid Manarul Ilmi ITS (JMMI-ITS), serta aktif dalam berbagai kepanitiaan seperti Mentoring Agama Islam ITS, Pelatihan Spiritual Kebangsaan ITS, serta berbagai kepanitiaan lainnya. Selain itu, selama kuliah penulis juga mengabdi dan menimba ilmu di Pondok Pesantren Darussalam, Keputih.