

**PENERAPAN ALGORITMA SUPPORT VECTOR MACHINE DAN  
SCALE INVARIANT FEATURE TRANSFORM UNTUK  
PENGENALAN TULISAN TANGAN PADA KARAKTER  
HANACARAKA AKSARA JAWA**

Sebagai syarat untuk memperoleh gelar sarjana S-1 di Program Studi Informatika, Jurusan Informatika, Fakultas Teknik Industri, Universitas Pembangunan Nasional “Veteran”  
Yogyakarta



Disusun oleh :

Rama Tri Agung  
123180053

**PROGRAM STUDI INFORMATIKA  
JURUSAN INFORMATIKA  
FAKULTAS TEKNIK INDUSTRI  
UNIVERSITAS PEMBANGUNAN NASIONAL “VETERAN”  
YOGYAKARTA  
2022**

## **HALAMAN PENGESAHAN PEMBIMBING**

### **PENERAPAN ALGORITMA SUPPORT VECTOR MACHINE DAN SCALE INVARIANT FEATURE TRANSFORM UNTUK PENGENALAN TULISAN TANGAN PADA KARAKTER HANACARAKA AKSARA JAWA**

Disusun Oleh:  
Rama Tri Agung  
123180053

Telah diperiksa dan disetujui oleh pembimbing untuk diuji  
pada tanggal : .....

Menyetujui,  
Pembimbing I

Pembimbing II

Dessyanto Boedi Prasetyo, S.T., M.T.  
NIDN. 0505127501

Mangaras Yanu Florestiyanto, S.T., M.Eng.  
NIDN. 0521018201

Mengetahui,  
Koordinator Tugas Akhir

Mangaras Yanu Florestiyanto, S.T., M.Eng.  
NIDN. 0521018201

## DAFTAR ISI

HALAMAN PENGESAHAN PEMBIMBING .....	ii
DAFTAR ISI .....	iii
DAFTAR TABEL .....	v
DAFTAR GAMBAR.....	vi
DAFTAR LAMPIRAN .....	viii
BAB I PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah.....	2
1.3. Batasan Masalah .....	2
1.4. Tujuan Penelitian .....	2
1.5. Manfaat Penelitian .....	3
1.6. Tahapan Penelitian.....	3
1.7. Sistematika Penulisan .....	4
BAB II TINJAUAN LITERATUR .....	5
2.1. Hanacaraka Aksara Jawa .....	5
2.2. Pengenalan Karakter Tulisan Tangan .....	5
2.3. <i>Machine Learning</i> .....	5
2.4. Augmentasi Data Gambar.....	6
2.5. Pengolahan Citra.....	7
2.6. <i>Scale Invariant Feature Transform</i> .....	9
2.7. <i>K-Means Clustering</i> .....	12
2.8. <i>Bag of Words</i> .....	13
2.9. <i>Feature Scaling</i> .....	13
2.10. <i>Support Vector Machine</i> .....	14
2.11. Optimasi Parameter Model .....	16
2.12. Evaluasi.....	17
2.13. Penelitian Sebelumnya.....	17
BAB III METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM .....	27
3.1. <i>Business Understanding</i> .....	29
3.2. <i>Data Understanding</i> .....	29
3.3. <i>Data Preparation</i> .....	30
3.3.1. <i>Data Augmentation &amp; Image Preprocessing</i> .....	30
3.3.2. <i>Feature Extraction</i> .....	32
3.3.3. <i>Feature Preprocessing</i> .....	47
3.4. <i>Modelling</i> .....	48
3.4.1. Pelatihan Model SVM.....	48
3.4.2. Optimasi Model.....	53
3.5. <i>Evaluation</i> .....	54
3.6. <i>Deployment</i> .....	55
3.6.1. <i>Analysis and Quick Design</i> .....	55

3.6.2. <i>Prototype Cycles</i> .....	58
3.6.3. <i>Testing</i> .....	58
3.6.4. <i>Implementation</i> .....	59
<b>BAB IV HASIL DAN PEMBAHASAN.....</b>	<b>60</b>
4.1. Implementasi.....	60
4.1.1. <i>Data Understanding</i> .....	61
4.1.2. <i>Data Preparation</i> .....	63
4.1.3. <i>Modelling</i> .....	67
4.1.4. <i>Evaluation</i> .....	69
4.1.5. <i>Deployment</i> .....	71
4.2. Hasil .....	73
4.2.1. Pengujian Model .....	73
4.2.2. Pengujian Sampel Data .....	80
4.3. Pembahasan .....	82
<b>BAB V KESIMPULAN DAN SARAN .....</b>	<b>85</b>
5.1. Kesimpulan .....	85
5.2. Saran .....	86
<b>DAFTAR PUSTAKA.....</b>	<b>87</b>
<b>LAMPIRAN .....</b>	<b>91</b>

## DAFTAR TABEL

Tabel 2.1 <i>State of The Art</i> .....	23
Tabel 2.2 <i>State of The Art</i> Lanjutan.....	24
Tabel 2.3 <i>State of The Art</i> Lanjutan.....	25
Tabel 3.1 Kernel Matriks dari <i>Gaussian Filter</i> .....	34
Tabel 3.2 Contoh Ilustrasi Konvolusi Matriks .....	34
Tabel 3.3 Hasil Matriks Setelah Dikonvolusi.....	35
Tabel 3.4 Matriks Blur Skala 0.....	36
Tabel 3.5 Matriks Blur Skala 1.....	36
Tabel 3.6 Matriks Hasil Perhitungan DoG .....	36
Tabel 3.7 Sampel Matriks <i>Keypoint</i> .....	39
Tabel 3.8 Matrik Hasil Perhitungan Magnitudo .....	39
Tabel 3.9 Matriks Hasil Perhitungan Arah Gradien .....	40
Tabel 3.10 Matriks Hasil Perhitungan Arah Gradien Tiap Blok.....	41
Tabel 3.11 Matriks Hasil Kernel RBF.....	49
Tabel 3.12 Hasil Perhitungan Matriks Hessian .....	50
Tabel 3.13 Hasil Perhitungan <i>Error</i> .....	50
Tabel 3.14 Hasil Perhitungan Delta Alfa .....	51
Tabel 3.15 Hasil Perhitungan Pembaruan Alfa .....	51
Tabel 3.16 Hasil Perhitungan Pembaruan Alfa Iterasi Kedua.....	51
Tabel 3.17 Hasil Akhir Perhitungan Pembaruan Alfa .....	52
Tabel 3.18 Hasil Perhitungan Kernel Data <i>Testing</i> .....	53
Tabel 3.19 Parameter Optimasi Model.....	54
Tabel 4.1 Hasil Pengujian Optimasi Parameter SVM-SIFT .....	73
Tabel 4.2 Rata-rata Akurasi Terhadap Parameter <i>Size</i> .....	74
Tabel 4.3 Rata-rata Akurasi Terhadap Parameter K.....	74
Tabel 4.4 Hasil Pelatihan SVM-SIFT Skenario 14 .....	75
Tabel 4.5 Hasil Pelatihan SVM tanpa SIFT .....	77
Tabel 4.6 Hasil Pelatihan SVM tanpa SIFT Lanjutan .....	78
Tabel 4.7 Detail Hasil Klasifikasi Sampel Data .....	81
Tabel 4.8 Detail Hasil Klasifikasi Sampel Data Lanjutan.....	82
Tabel 4.9 Perbandingan Akurasi dari Penelitian Lain .....	84

## DAFTAR GAMBAR

Gambar 1.1 <i>CRISP-DM Data Science Process</i> .....	3
Gambar 1.2 Metode Pengembangan Perangkat Lunak RAD .....	4
Gambar 2.1 Karakter Hanacaraka .....	5
Gambar 2.2 Perbandingan Histogram Asli dengan Hasil Perataan Histogram .....	9
Gambar 2.3 <i>Scale Space</i> dan DoG.....	10
Gambar 2.4 Proses pencarian kandidat <i>keypoint</i> pada tiap skala .....	10
Gambar 2.5 Hasil identifikasi <i>interest keypoints</i> .....	11
Gambar 2.6 Fitur SIFT .....	12
Gambar 2.7 Contoh klastering pada model BoW .....	13
Gambar 3.1 Alur Metodologi Penelitian .....	28
Gambar 3.2 Flowchart Proses Utama .....	29
Gambar 3.3 Contoh Karakter ‘ha’ .....	30
Gambar 3.4 <i>Flowchart Image Augmentation &amp; Preprocessing Process</i> .....	30
Gambar 3.5. Hasil Augmentasi dan <i>Preprocessing</i> pada Gambar Karakter ‘ba’ .....	32
Gambar 3.6 <i>Flowchart</i> Ekstraksi SIFT .....	33
Gambar 3.7 Penerapan <i>Gaussian Filter</i> pada gambar .....	35
Gambar 3.8 Gambar DoG.....	37
Gambar 3.9 Hasil identifikasi interest keypoints.....	38
Gambar 3.10 Potongan gambar salah satu keypoint .....	38
Gambar 3.11 Histogram dari Besaran Arah Gradien .....	40
Gambar 3.12 Hasil gambar keypoint setelah dilakukan rotasi .....	41
Gambar 3.13 Histogram deskriptor blok pertama .....	42
Gambar 3.14 <i>Flowchart Create BoW</i> .....	43
Gambar 3.15 <i>Flowchart Create Feature from BoW</i> .....	45
Gambar 3.16 <i>Flowchart Feature Preprocessing</i> .....	47
Gambar 3.17 <i>Flowchart Modelling SVM</i> .....	49
Gambar 3.18 Perancangan Arsitektur Sistem.....	56
Gambar 3.19 Proses DFD Level 0.....	56
Gambar 3.20 Proses DFD Level 1 .....	57
Gambar 3.21 Kerangka Antarmuka Sebelum Proses Klasifikasi .....	57
Gambar 3.22 Kerangka Antarmuka Setelah Proses Klasifikasi .....	58
Gambar 4.1. Halaman Aplikasi Hasil Prediksi.....	60
Gambar 4.2 Hasil Keluaran Modul Program 4.7 .....	63
Gambar 4.3 Hasil Augmentasi dan <i>Preprocessing</i> .....	65
Gambar 4.4 Hasil Ekstraksi Fitur SIFT .....	65
Gambar 4.5 Proses Pelatihan dan Optimasi Parameter .....	68
Gambar 4.6 Tampilan Hasil Pelatihan dan Optimasi .....	69
Gambar 4.7 Hasil Plot <i>Confussion Matrix</i> .....	70
Gambar 4.8 Hasil <i>Classification Report</i> .....	71
Gambar 4.9 Hasil Perhitungan Akurasi .....	71

Gambar 4.10 Proses <i>Build/Deploy Heroku</i> .....	72
Gambar 4.11 Hasil <i>Confusion Matrix</i> SVM-SIFT Skenario 14 .....	76
Gambar 4.12 Hasil <i>Classification Report</i> SVM-SIFT Skenario 14 .....	76
Gambar 4.13 Hasil Confusion Matrix SVM tanpa SIFT .....	78
Gambar 4.14 Hasil <i>Classification Report</i> SVM tanpa SIFT .....	79
Gambar 4.15 Perbandingan Akurasi SVM-SIFT dan SVM tanpa SIFT .....	79
Gambar 4.16 Contoh Variasi Sampel Data Karakter ‘Ba’ .....	80
Gambar 4.17 Hasil <i>Classification Report</i> SVM-SIFT pada Sampel Data .....	81

## **DAFTAR LAMPIRAN**

LAMPIRAN A. Pengujian Parameter Skenario 1 .....	92
LAMPIRAN B. Pengujian Parameter Skenario 2 .....	94
LAMPIRAN C. Pengujian Parameter Skenario 3 .....	96
LAMPIRAN D. Pengujian Parameter Skenario 4 .....	98
LAMPIRAN E. Pengujian Parameter Skenario 5 .....	100
LAMPIRAN F. Pengujian Parameter Skenario 6 .....	102
LAMPIRAN G. Pengujian Parameter Skenario 7 .....	104
LAMPIRAN H. Pengujian Parameter Skenario 8 .....	106
LAMPIRAN I. Pengujian Parameter Skenario 9 .....	108
LAMPIRAN J. Pengujian Parameter Skenario 10 .....	110
LAMPIRAN K. Pengujian Parameter Skenario 11 .....	112
LAMPIRAN L. Pengujian Parameter Skenario 12 .....	114
LAMPIRAN M. Pengujian Parameter Skenario 13 .....	116
LAMPIRAN N. Pengujian Parameter Skenario 14 .....	118
LAMPIRAN O. Pengujian Parameter Skenario 15 .....	120
LAMPIRAN P. Pengujian Parameter Skenario 16 .....	122
LAMPIRAN Q. Pengujian Parameter Skenario 17 .....	124
LAMPIRAN R. Pengujian Parameter Skenario 18 .....	126
LAMPIRAN S. Pengujian Parameter Skenario 19 .....	128
LAMPIRAN T. Pengujian Parameter Skenario 20 .....	130

## **BAB I**

### **PENDAHULUAN**

#### **1.1. Latar Belakang**

Aksara Jawa "ha-na-ca-ra-ka" merupakan salah satu warisan leluhur bangsa Indonesia (Sari et al., 2018). Aksara Jawa juga bagian dari bahasa Jawa yang melekat dalam budaya Jawa. Dengan perkembangan teknologi dan komunikasi global, kondisi budaya semakin terkikis. Ratusan bahasa daerah di Indonesia terancam punah. Salah satu bahasa daerah yang terancam adalah bahasa Jawa (Lorentius et al., 2019). Pengguna bahasa Jawa ini semakin berkurang jumlahnya dan hanya sedikit remaja yang mengenal aksara Jawa dengan jelas (Setiawan et al., 2019), karena minat masyarakat terhadap aksara jawa juga sangat memprihatinkan (Lorentius et al., 2019).

Melihat kondisi tersebut, bagaimana pentingnya nilai dan eksistensi budaya tersebut, maka perlu sebuah sistem yang dapat mengenali huruf-huruf Hanacaraka Aksara Jawa (Lorentius et al., 2019). Sistem tersebut dapat diimplementasikan dalam bidang pengenalan tulisan tangan (*Handwriting recognition*) yang merupakan kemampuan komputer dalam menerima dan memproses input tulisan tangan manusia yang dapat dipahami dari sumber seperti dokumen kertas, foto, dan lain-lain. Pengenalan tulisan tangan ini berguna untuk menunjang kelestarian Aksara Jawa sebagai alat atau perangkat lunak yang memiliki kemampuan untuk mengenali tulisan tangan karakter Aksara Jawa secara otomatis (Dewa et al., 2018).

Pengenalan tulisan tangan pada karakter hanacaraka aksara jawa telah diusulkan oleh beberapa penelitian sebelumnya dengan menggunakan metode yaitu *Convolutional Neural Network* (CNN) (Dewa et al., 2018) (Rismiyati et al., 2017) (Wibowo et al., 2018), *K-Nearest Neighbor* (KNN) (Sari et al., 2018), dan *Support Vector Machine* (SVM) (Rismiyati et al., 2018). Pada metode CNN memiliki performa klasifikasi yang sangat baik dalam bidang ini dengan tingkat akurasi yang dapat mencapai 94.57% (Wibowo et al., 2018), CNN juga mahir dalam menangani inputan yang bersifat *noisy* (Rajesh et al., 2016), namun akurasi yang tinggi pada CNN membutuhkan jumlah data training yang banyak (Wibowo et al., 2018) dan dengan metode yang kompleks ini maka arsitekturnya akan cukup sulit dibangun serta dapat terjadinya *overfitting* (Rajesh et al., 2016). KNN adalah metode yang sederhana, efektif, mudah diterapkan, tidak parametrik dan memberikan tingkat kesalahan yang rendah dalam proses pelatihan (Thamilselvana & Sathiaseelan, 2015), metode ini tidak membutuhkan jumlah dataset yang banyak (Sari et al., 2018), tapi kekurangannya pada metode KNN relatif memiliki hasil performa yang kurang baik daripada metode lainnya (Naufal et al., 2021) dan sulit menemukan nilai optimal (Thamilselvana & Sathiaseelan, 2015). Kemudian metode SVM merupakan metode paling efektif dalam klasifikasi, terutama populer dalam klasifikasi teks, memiliki akurasi yang cukup tinggi (Thamilselvana & Sathiaseelan, 2015), tidak memiliki masalah dalam *overfitting* (Rajesh et al., 2016), dan tidak membutuhkan jumlah dataset yang sangat besar (Rismiyati et al., 2018). Namun sayangnya,

metode ini cukup sulit untuk mencari model parameter yang cocok maupun optimal dalam klasifikasi (Thamilselvana & Sathiaseelan, 2015).

Beberapa penelitian diatas, metode SVM dapat digunakan secara fleksibel tanpa membutuhkan dataset yang besar dan memiliki performa yang cukup baik, namun penelitian sebelumnya yang menggunakan metode tersebut belum dapat menyaingi akurasi dari metode CNN yang diatas 90% (Rismiyati et al., 2018) (Sari et al., 2018). Untuk dapat meningkatkan akurasi dibutuhkan bantuan fitur ekstraksi, pada penelitian pengenalan tulisan tangan karakter lainnya (Thailand, Bangla dan Latin) telah mengusulkan metode yaitu menggunakan *Scale Invariant Feature Transform Descriptor* (SIFT Descriptor) yang berpengaruh dalam peningkatan akurasi klasifikasi secara signifikan menjadi diatas 95% dan mengungguli performa fitur ekstraksi *Histograms of Oriented Gradients* (HOG) (Surinta et al., 2015).

Sehingga pada penelitian ini pengenalan tulisan tangan pada karakter hanacaraka aksara jawa akan menerapkan metode SVM dengan bantuan SIFT dalam meningkatkan performa akurasi. Dari metode tersebut akan dibandingkan dan menemukan bagaimana pengaruh terhadap akurasi jika metode SIFT diterapkan dalam ekstraksi fitur.

## 1.2. Rumusan Masalah

Berdasarkan dari latar belakang, rumusan masalah pada penelitian ini yaitu:

1. Bagaimana menerapkan algoritma SVM dalam klasifikasi pengenalan tulisan tangan hanacaraka aksara jawa?
2. Bagaimana menerapkan algoritma SIFT sebagai ekstraksi fitur dalam membantu meningkatkan akurasi klasifikasi?
3. Bagaimana penerapan evaluasi performa akurasi dalam klasifikasi?

## 1.3. Batasan Masalah

Batasan masalah dari penelitian ini yaitu :

1. Klasifikasi dilakukan hanya pada 20 karakter Hanacaraka Aksara Jawa.
2. Sumber data berasal dari dataset yang disediakan di [www.kaggle.com](http://www.kaggle.com) oleh Phiard.
3. Data diaugmentasi dengan tujuh variasi.
4. Data yang digunakan dalam format gambar.
5. Analisis dilakukan untuk melihat performa algoritma menggunakan akurasi klasifikasi.

## 1.4. Tujuan Penelitian

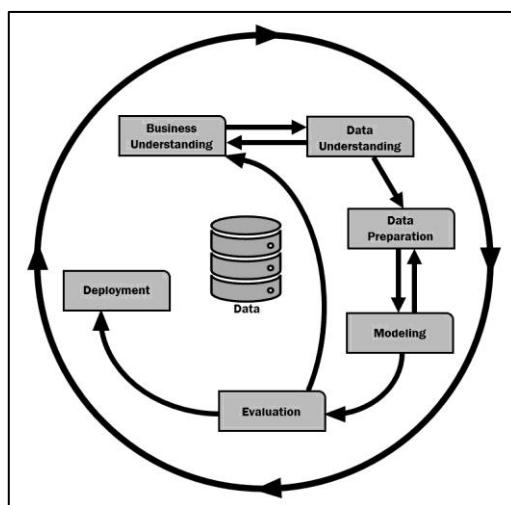
Tujuan dari penelitian ini yaitu menerapkan algoritma SVM dengan bantuan SIFT sebagai ekstraksi fitur dalam melakukan klasifikasi tulisan tangan aksara jawa dan mengidentifikasi performa akurasi algoritma yang terbaik dalam melakukan klasifikasi.

## 1.5. Manfaat Penelitian

Hasil penelitian dapat dimanfaatkan dalam bantuan pembelajaran baik pada siswa di instansi pendidikan maupun orang lain secara individu dan membangun kembali budaya bahasa khas jawa dengan mengenal lebih mudah terhadap karakter-karakternya.

## 1.6. Tahapan Penelitian

Tahapan-tahapan pada penelitian ini menerapkan metode proses *Cross Industry Standard Process for Data Mining* (CRISP-DM) merujuk pada penelitian oleh Rasyidi, et al. (2021), Dowa, et al. (2018), dan Schröera, et al. (2021) yang telah dimodifikasi untuk menyesuaikan kebutuhan penelitian. Metode ini adalah suatu standarisasi pemrosesan *data mining* yang telah dikembangkan dimana data yang ada akan melewati setiap fase terstruktur dan terdefinisi dengan jelas dan efisien (Hasanah et al., 2021). Berikut adalah proses CRISP-DM yang diterapkan pada penelitian ini



Gambar 1.1 *CRISP-DM Data Science Process*

### 1. *Business Understanding*

Melakukan studi literatur yang berkaitan dengan penelitian ini yaitu pengenalan tulisan tangan, karakter aksara jawa, metode SVM dan SIFT.

### 2. *Data Understanding*

Mengumpulkan dan mengeksplorasi *dataset* yang perlu digunakan dalam penelitian ini dan memahami pola serta struktur yang penting dari data gambar tersebut.

### 3. *Data Preparation*

Melakukan persiapan data sebelum dilatih terhadap model. Beberapa tahap yang dilakukan disini yaitu augmentasi data gambar, *preprocessing* data gambar, fitur ekstraksi SIFT, dan normalisasi fitur serta *splitting* data menjadi *training* dan *testing*.

### 4. *Modelling*

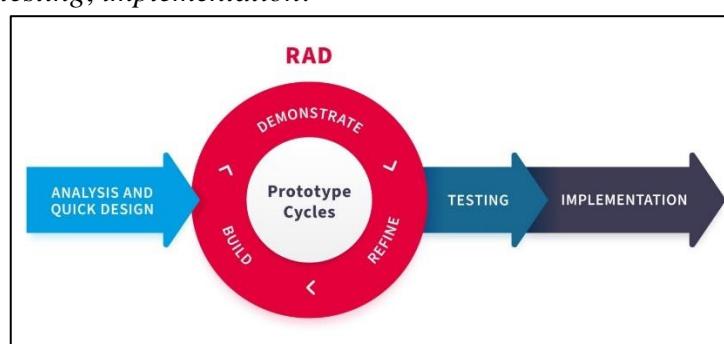
Melakukan pelatihan data fitur yang telah disiapkan menggunakan model *Support Vector Machine*, serta melakukan *tuning parameter* untuk mencari parameter terbaik dari model SVM tersebut.

##### 5. *Evaluation*

Melakukan evaluasi performa akurasi terhadap model yang telah dibangun dengan beberapa macam skenario pengujian.

##### 6. *Deployment*

Melakukan pengembangan sistem aplikasi berbasis *website* dari hasil model yang telah selesai dilatih dan diuji dengan baik. *Deployment* ini fase terakhir yang berguna untuk menyediakan interaksi pengguna terhadap aplikasi sehingga dapat dioperasikan secara luas atau global. Tahap ini juga disebut dengan proses pengembangan perangkat lunak. Proses pengembangan ini akan menggunakan metode *Rapid Application Development* (RAD) yang merupakan salah satu metode pengembangan suatu sistem informasi dengan waktu yang relatif singkat (Noertjahyana, 2004). RAD dapat dijadikan acuan untuk mengembangkan suatu sistem informasi yang unggul dalam hal kecepatan, ketepatan dan biaya yang lebih rendah (Hidayat & Hati, 2021). Beberapa proses yang ada di dalam metode RAD diantaranya yaitu *analysis and quick design, prototype cycles (build, demonstrate, refine), testing, implementation*.



Gambar 1.2 Metode Pengembangan Perangkat Lunak RAD

## 1.7. Sistematika Penulisan

Pada penelitian ini memiliki sistematika penulisan yang terurut yaitu pertama pada Bab I (Pendahuluan) yang menjelaskan latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, tahapan penelitian, dan sistematika penulisan. Bab I ini bertujuan untuk memudahkan pembaca memahami maksud dan tujuan penelitian ini. Kemudian pada Bab II (Tinjauan Pustaka) menjelaskan mengenai pembahasan dari penelitian terkait dengan penelitian sebelumnya yang akan digunakan sebagai referensi di penelitian ini. Selanjutnya pada Bab III (Metodologi Penelitian) yaitu menjelaskan mengenai metode-metode penyelesaian yang akan dilakukan pada penelitian ini untuk menyelesaikan masalah yang diangkat mulai dari perancangan hingga pengujian. Setelah itu pada Bab IV (Hasil dan Pembahasan) yaitu menjelaskan mengenai analisis dan pembahasan dari hasil yang didapatkan pada sistem yang telah dibangun berdasarkan metode yang digunakan. Terakhir pada Bab V (Kesimpulan dan Saran) yaitu menjelaskan mengenai kesimpulan yang didapatkan dari hasil penelitian ini dan memberikan saran yang dapat digunakan sebagai bekal pada pengembangan penelitian ini maupun penelitian selanjutnya.

## BAB II

### TINJAUAN LITERATUR

#### 2.1. Hanacaraka Aksara Jawa

Aksara jawa atau lebih dikenal dengan Hanacaraka adalah turunan aksara brahmani (berasal dari Hindustan) yang digunakan untuk penulisan berbahasa Jawa, bahasa Makasar, bahasa Madura, bahasa Melayu, bahasa Sunda, bahasa Bali dan bahasa Sasak (Hidayat & Shofa, 2016). Banyak orang yang fasih berbahasa jawa, namun sangat sedikit orang yang bisa membaca aksara Hanacaraka. Penggunaan aksara Hanacaraka secara umum pun juga sangat terbatas, misalnya pada papan penunjuk jalan, papan nama, dan beberapa artikel yang ada pada koran serta majalah. Itupun hanya untuk sekedar mempertahankan keberadaan aksara Hanacaraka agar tidak hilang sama sekali (Setiawan & Sulaiman, 2015). Hanacaraka aksara jawa terdiri dari 20 karakter huruf dasar yaitu seperti pada Gambar 2.1 yang akan digunakan sebagai data penelitian ini.



Gambar 2.1 Karakter Hanacaraka

#### 2.2. Pengenalan Karakter Tulisan Tangan

Pengenalan gambar adalah prosedur penting untuk pemrosesan gambar, bagaimana sebuah gambar dipersepsi sebagai manusia yang mempersepsi gambar tersebut. Pengenalan tulisan tangan adalah salah satu masalah yang paling dicari dan dipertimbangkan, karena tulisan tangan dapat memungkinkan orang untuk melakukan beberapa pekerjaan, misalnya, pasca-presentasi, pemeriksaan cek bank, dan penanganan tertulis secara manual pada struktur. Juga dapat berkontribusi besar dalam pengembangan proses otomasi dan dapat mengembangkan interaksi antara manusia dan mesin dalam berbagai aplikasi seperti otomasi perkantoran dan aplikasi entri data (Ali et al., 2019). Sistem pengenalan berusaha mengenali teks yang diubah menjadi format yang dapat dieksekusi mesin (format digital) yang dapat diproses oleh perangkat lunak pengolah karakter baik secara *online* maupun *offline*.

#### 2.3. Machine Learning

*Machine Learning* merupakan cara untuk memungkinkan suatu mesin memecahkan sebuah masalah yang baru dengan mempelajari pola terhadap cara memecahkan masalah yang telah dipecahkan sebelumnya (Mohaiminul & Sultana, 2018). *Machine Learning* ini

merupakan salah satu bidang *Artificial Intelligent* yang memungkinkan sebuah sistem untuk dapat beradaptasi atau belajar melalui data training yang disediakan. Kemampuan kinerja dari *Machine Learning* sangat dipengaruhi oleh algoritma yang digunakan untuk menyelesaikan sebuah masalah (Liu et al., 2017).

Dalam *Machine Learning* ada dua metode yang umum digunakan yaitu *Supervised Learning* dan *Unsupervised Learning*. *Supervised learning* adalah metode di mana data yang digunakan sebelumnya diberikan sebuah label dengan data yang sesuai untuk setiap kelasnya (Saputra et al., 2019). Kemudian *Unsupervised Learning* merupakan metode di mana data pelatihan yang digunakan tidak diberikan label sebagai kelas data, sehingga *Machine Learning* menganalisis data dengan menganalisis kesamaan atau kedekatan pola pada data yang ada. (Chen X. et al., 2021).

## 2.4. Augmentasi Data Gambar

Augmentasi data adalah strategi yang memungkinkan praktisi untuk secara signifikan meningkatkan keragaman data yang tersedia untuk model pelatihan, tanpa benar-benar mengumpulkan data baru (Sanjaya & Ayub, 2020). Augmentasi gambar, secara sederhana, merupakan tindakan mereplika gambar yang ada dengan berbagai penyesuaian untuk memperbanyak data latih (Hadiprakoso & Qomariasih, 2022). Augmentasi data meningkatkan variasi gambar dengan memanipulasi transformasi dimensi gambar, maka pada dasarnya metode yang digunakan dalam augmentasi data ini menggunakan perlakuan metode pengolahan citra. Memperkecil ukuran gambar, rotasi gambar, dan *shear* gambar merupakan teknik yang digunakan pada penelitian ini untuk meningkatkan jumlah data latih. Model terlatih akan lebih realistik dari kondisi dunia nyata dan akan mampu beradaptasi dengan berbagai perubahan kondisi yang ada (Hadiprakoso & Oomariasih, 2022).

### a. Rotasi Gambar

Secara sederhana perhitungan untuk melakukan rotasi gambar dapat menggunakan rumus matriks berikut ini.

Pada persamaan diatas akan menghasilkan matriks gambar yang berputar belawan arah jarum jam sesuai derajat  $\theta$  yang ditentukan, dengan  $x_0$  dan  $y_0$  sebagai sumbu putar dari sudut putaran yang dilakukan. Pada kasus penelitian ini sumbu putar akan diperoleh dari titik tengah imbang sebuah gambar.

b. Memperkecil Gambar

Proses ini hanya memperkecil gambar tanpa mengubah ukuran piksel dari gambar tersebut. Adapun cara yang dilakukan yaitu dengan menerapkan metode *resizing* gambar menjadi ukuran yang diinginkan kemudian hasil *resizing* gambar tersebut dimasukkan kembali dalam matriks ukuran piksel awal sehingga beberapa piksel yang ada di pojok gambar akan berwarna hitam karena nilai piksel tersebut tidak ada atau nol. Kedua proses tersebut akan menghasilkan gambar yang lebih kecil namun tidak mengurangi ukuran piksel yang ada. Metode augmentasi ini cukup berguna dalam

menciptakan variasi gambar baru bahkan dapat digabungkan dengan metode lainnya seperti rotasi gambar agar matriks tepi gambar tidak banyak terpotong ketika dilakukan dirotasi yang ekstrim.

### c. Shear Gambar

Konsep ini hampir sama seperti saat melakukan rotasi gambar, karena hal yang dilakukan yaitu mengkonvolusikan sebuah matriks khusus dengan indeks matriks gambar. Konvolusi yang dilakukan pada penelitian ini menggunakan ukuran matriks  $2 \times 2$ . Dalam melakukan transformasi ini terdapat dua sumbu yang akan berubah yaitu sumbu x dan sumbu y, gambar akan bergeser kearah horizontal ketika sumbu x ditransformasikan, sedangkan gambar akan bergeser kearah vertikal ketika sumbu y ditransformasikan.

Berikut adalah matriks untuk mentransformasikan sumbu x.

Dan berikut adalah matriks untuk mentransformasikan sumbu y.

Diatas merupakan matriks konvolusi untuk transformasi *shear* pada gambar. Nilai  $k$  merupakan indeks besaran dari seberapa tinggi pengaruh pergeseran gambar secara horizontal maupun vertikal.

## 2.5. Pengolahan Citra

Pengolahan citra (*Image Processing*) adalah sebuah disiplin ilmu yang mempelajari tentang teknik-teknik mengolah citra. Citra yang dimaksud disini adalah gambar diam (foto) maupun gambar bergerak (video) (Kusumanto & Tompunu, 2011). Pengolahan citra berkaitan dengan perbaikan kualitas terhadap suatu gambar (meningkatkan kontras, perubahan warna, restorasi citra), transformasi gambar (translasi, rotasi transformasi, skala, geometrik), melakukan pemilihan citra ciri (*feature images*) yang optimal untuk tujuan analisis, melakukan penyimpanan data yang sebelumnya dilakukan reduksi dan kompresi, transmisi data, dan waktu proses data (Munantri et al., 2019). Secara umum pengolahan citra digital dapat diartikan sebagai pemrosesan gambar dua dimensi dengan menggunakan komputer. Citra digital merupakan sebuah array (larik) yang berisikan nilai-nilai real maupun komplek yang dapat direpresentasikan dengan deretan bit tertentu (Munantri et al., 2019).

Pada penelitian ini membutuhkan proses *image data preprocessing*, sehingga banyak metode pengolahan citra yang diterapkan. Sebagian proses pengolahan citra yang dilakukan pada penelitian ini diterapkan pada proses augmentasi data yang berguna untuk mereplikasi data sehingga *dataset* yang digunakan dapat lebih banyak dan bervariasi. Sebagian lainnya proses pengolahan citra dilakukan diluar augmentasi data yang dilakukan, beberapa diantaranya yaitu melakukan proses *grayscale* gambar, *resizing* gambar, dan *histogram equalization* gambar.

*a. Resizing*

Proses *resizing* gambar pada penelitian ini lebih tertuju pada pengecilan ukuran piksel gambar. Penelitian ini menggunakan opencv-python dalam melakukan proses pengecilan gambar. Proses yang terjadi didalamnya menggunakan *interpolation* INTER\_AREA, proses ini menggunakan hubungan area piksel untuk pengambilan sampel ulang. Ini paling cocok untuk mengurangi ukuran gambar (mengecilkan gambar).

### *b. Grayscaleing*

Salah satu cara untuk mempercepat komputasi citra yaitu dengan mengubah warna citra gambar menjadi hitam putih (*grayscale*). Pada penelitian ini gambar yang digunakan hanya sekedar huruf hanacaraka sehingga tidak akan mengubah isi fitur didalamnya apabila mengubah warna citra gambar. Pada penelitian ini proses *grayscaleing* dilakukan dengan menggunakan metode pembobotan (*weighted*). Berikut adalah rumus untuk mengkonversi citra gambar berwarna menjadi citra gambar hitam putih.

$$\text{Grayscale} = 0.299R + 0.587G + 0.114B \dots \quad (2.4)$$

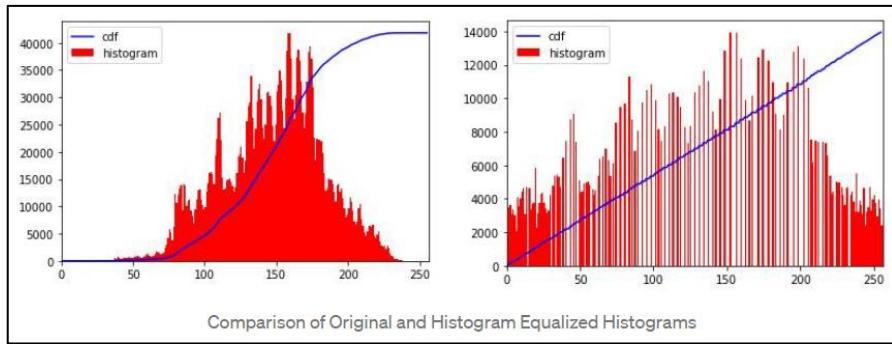
Rumus 2.4. menghitung nilai kombinasi RGB (*red*, *green*, *blue*) dengan bobot setiap masing-masing warna.

### c. Histogram equalization

*Histogram Equalization* atau perataan histogram adalah metode dalam pengolahan citra yang menggunakan histogram dari suatu citra untuk mengatur tingkat keceraham citra tersebut. *Histogram equalization* ini dilakukan untuk perataan histogram pada citra, dimana distribusi nilai derajat warna pada suatu citra dibuat merata. Dengan *histogram equalization* ini sebuah citra akan memiliki kontras yang seragam dan derajat atau tingkat warna yang merata. Yang dimaksud dengan perataan histogram ini adalah mengubah derajat keabuan suatu piksel ( $r$ ) dengan derajat keabuan yang baru ( $s$ ) dengan suatu fungsi transformasi  $T$ , yang dalam hal ini  $s = T(r)$ . Berikut fungsi histogramnya secara menerus.

Dan berikut fungsi histogramnya dalam bentuk diskrit.

Setelah menghitung matriks citra, hasil akhir yang diperoleh memungkinkan dalam bentuk desimal, angka desimal tidak akan bisa menginterpretasikan sebuah nilai warna sehingga perlu membulatkan nilai tersebut menjadi nilai angka bilangan bulat.



**Gambar 2.2** Perbandingan Histogram Asli dengan Hasil Perataan Histogram

## 2.6. Scale Invariant Feature Transform

SIFT (*Scale Invariant Feature Transform*) telah terbukti berkinerja lebih baik daripada deskriptor lokal lainnya (Mortensen et al. 2005). Diberikan titik fitur, deskriptor SIFT menghitung vektor gradien untuk setiap piksel di lingkungan titik fitur dan membangun histogram arah gradien yang dinormalisasi. Deskriptor SIFT menciptakan lingkungan 16x16 yang dipartisi menjadi 16 subwilayah masing-masing 4x4 piksel. Untuk setiap piksel dalam subwilayah, SIFT menambahkan vektor gradien piksel ke histogram arah gradien dengan mengkuantisasi setiap orientasi ke salah satu dari 8 arah dan memberi bobot kontribusi setiap vektor berdasarkan besarnya. Setiap arah gradien selanjutnya dibobot dengan skala Gaussian  $= n/2$  di mana  $n$  adalah ukuran lingkungan dan nilai-nilai didistribusikan ke bin tetangga menggunakan interpolasi trilinear untuk mengurangi efek batas saat sampel bergerak di antara posisi dan orientasi (Mortensen et al. 2005). Berikut beberapa proses penting dalam SIFT (Wang et al., 2013) (Hassan et al., 2018):

### a. Scale-space extrema detection

Proses pertama diawali dengan mendeteksi *points of interest*, atau yang disebut juga *keypoint* pada SIFT. Gambar asli diambil dan menghasilkan urutan gambar blur terus-menerus, kemudian mengubah ukuran gambar asli menjadi 50% dari ukurannya dan menghasilkan gambar blur berulang kali. Citra dikonvolusikan dengan filter *Gaussian* pada skala yang berbeda, kemudian diambil perbedaan dari citra *Gaussian* yang berurutan. *Keypoint* kemudian diambil sebagai maxima/minima dari *Difference of Gaussians* (DoG) yang terjadi pada beberapa skala. Berikut rumus DoG.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma) \dots \quad (2.7)$$

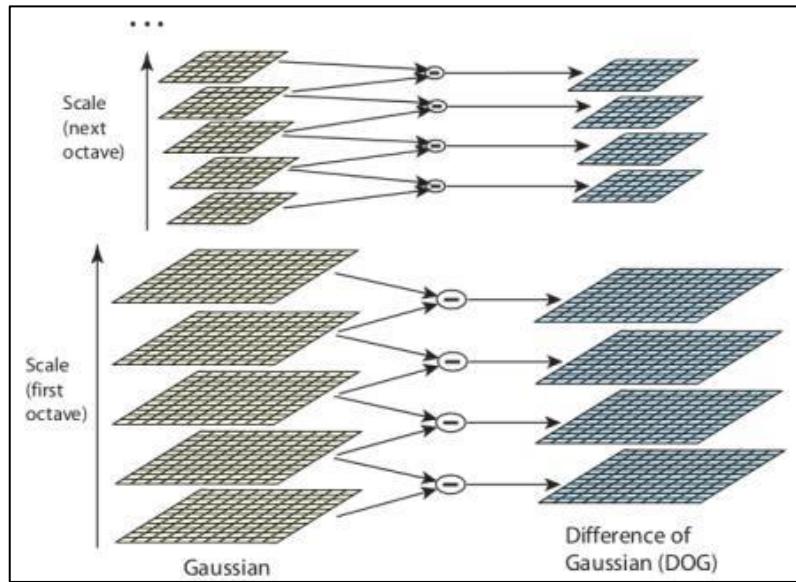
Dimana  $L(x, y, k\sigma)$  adalah konvolusi dari citra asli  $I(x, y)$  dengan *Gaussian Blur*  $G(x, y, k\sigma)$  dalam skala  $k\sigma$  yaitu.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \dots \quad (2.8)$$

Dan

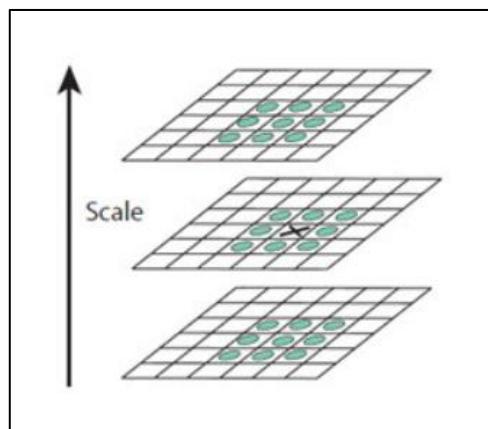
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \dots \quad (2.9)$$

Nilai  $k$  merupakan nilai kontinu dari beberapa nilai skala yang berbeda-beda untuk membandingkan beberapa hasil dari pengurangan konvolusi *Gaussian Blur*.



**Gambar 2.3 Scale Space dan DoG**

Setelah gambar DoG diperoleh, *keypoints* diidentifikasi sebagai *local minima/maxima* dari gambar DoG di seluruh skala. Hal ini dilakukan dengan membandingkan setiap piksel dalam gambar DoG dengan delapan tetangganya pada skala yang sama dan sembilan piksel tetangga yang sesuai di setiap skala tetangga. Jika nilai piksel adalah maksimum atau minimum di antara semua piksel yang dibandingkan, maka dipilih sebagai candidate *keypoint*.

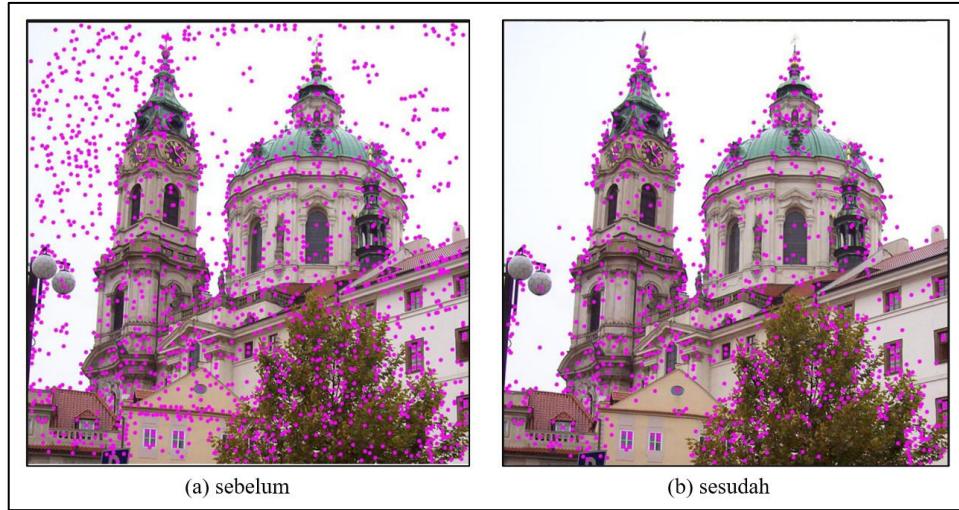


**Gambar 2.4 Proses pencarian kandidat *keypoint* pada tiap skala**

#### b. Keypoint localization

Menemukan *maxima/minima* dalam citra DoG dan menemukan sub piksel *maxima/minima* untuk mendapatkan *keypoints* pada *scale space extreme* di citra DoG. Melakukan pencocokan detail ke data terdekat untuk lokasi, skala, dan rasio kelengkungan utama yang akurat. Informasi ini memungkinkan untuk menolak titik yang kontrasnya rendah (dan karenanya sensitif terhadap *noise*) atau tidak terlokalisasi

di sepanjang tepi. Tepi dan area kontras rendah dianggap sebagai poin kunci yang buruk. Penghapusan poin-poin kunci tersebut meningkatkan efisiensi dan ketahanan algoritma. Pendekatan yang mirip dengan *Harris Corner Detector* diterapkan di sini.



**Gambar 2.5 Hasil identifikasi *interest keypoints***

#### c. *Orientation assignment*

Setiap *keypoint* diberikan satu atau lebih orientasi berdasarkan arah gradien gambar lokal. Ini adalah langkah kunci dalam mencapai invarian terhadap rotasi karena *deskriptor keypoint* dapat direpresentasikan relatif terhadap orientasi ini dan karenanya mencapai invarian terhadap rotasi gambar. Besaran gradien dan orientasi diperoleh dengan menggunakan persamaan. Besaran dan orientasi dihitung untuk semua piksel yang mengelilingi titik-titik kunci. Setelah itu, histogram dapat dibuat.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \dots \quad (2.10)$$

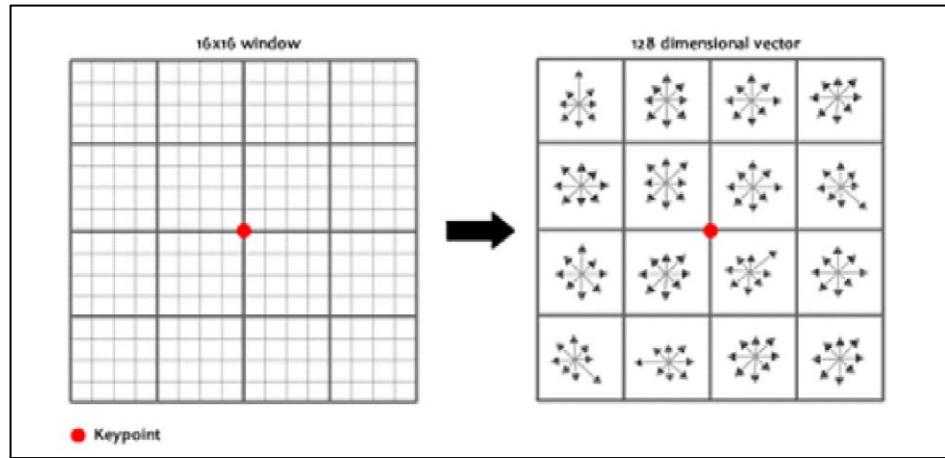
$$\theta(x, y) = \text{atan2}(L(x, y+1) - L(x, y-1), L(x+1, y) - L(x-1, y)) \dots \quad (2.11)$$

Citra  $L(x, y, \sigma)$  yang dihaluskan *Gaussian* pada skala *keypoint*  $\sigma$  diambil sehingga semua komputasi dilakukan dengan cara *scale-invariant*. Untuk sampel gambar  $L(x, y)$  pada skala  $\sigma$ , besaran gradien  $m(x, y)$ , dan orientasi  $\theta(x, y)$ , dihitung terlebih dahulu menggunakan rumus diatas.

#### d. *Keypoint descriptor*

Terakhir, menghitung vektor deskriptor untuk setiap *keypoints* sedemikian rupa sehingga deskriptor sangat unik/berbeda dan sebagian tidak berubah untuk variasi yang tersisa seperti iluminasi, sudut pandang 3D, dll. Dengan representasi ini, dimungkinkan dapat dengan mudah memperoleh fitur yang diperlukan. Untuk melakukannya, matriks  $16 \times 16$  yang disekitar *keypoints* diatur dan matriks ini dibagi menjadi 16 matriks ukuran  $4 \times 4$ . Di dalam setiap matriks  $4 \times 4$ , besaran gradien dan orientasi dapat diperoleh. Histogram ini dibagi menjadi delapan bin dan jumlah

orientasi yang ditambahkan ke bin tergantung pada besaran gradien. Sehingga, setiap titik kunci dideskripsikan oleh  $4 \times 4 \times 8 = 128$  dimensi vektor.



**Gambar 2.6 Fitur SIFT**

## 2.7. K-Means Clustering

*K-Means Clustering* adalah suatu metode penganalisaan data atau metode data mining yang melakukan proses pemodelan tanpa supervisi (*unsupervised*) dan merupakan salah satu metode yang melakukan pengelompokan data dengan sistem partisi (Yunita, 2018). Data-data yang memiliki karakteristik yang sama dikelompokkan dalam satu *cluster/kelompok* dan data yang memiliki karakteristik yang berbeda dikelompokkan dengan *cluster/kelompok* yang lain sehingga data yang berada dalam satu *cluster/kelompok* memiliki tingkat variasi yang kecil (Sari et al., 2018). Metode *K-Means* ini merupakan teknik dalam klaster data yang sangat terkenal karena kecepatannya dalam mengklasterkan data (Vhallah et al., 2018).

Menurut Pradnyana & Permana (2018) proses *K-Means* dimulai dengan mengidentifikasi data yang akan di-*cluster*,  $X = [x_1, x_2, \dots, x_n]$  dengan  $n$  adalah jumlah data yang akan dikelompokkan. Pada awal iterasi, pusat setiap *cluster* yaitu *centroid*, ditetapkan secara bebas (sembarang),  $C = [c_1, c_2, \dots, c_n]$ . Kemudian dihitung jarak antara setiap data dengan setiap pusat *cluster*. Untuk melakukan penghitungan jarak data ke-i ( $x_i$ ) pada pusat cluster ke-k ( $c_k$ ), diberi nama ( $d_{ik}$ ), dapat digunakan formula *Euclidean Distance*. Suatu data akan menjadi anggota dari *cluster* ke-k apabila jarak dengan pusat *cluster* tersebut merupakan jarak terpendek dibandingkan dengan pusat *cluster* lain, berikut rumus nya.

Selanjutnya, kelompokkan data-data yang menjadi anggota pada setiap *cluster*. Nilai pusat *cluster* yang baru dapat dihitung dengan cara mencari nilai rata-rata dari data-data yang menjadi anggota pada *cluster* tersebut menggunakan rumus berikut.

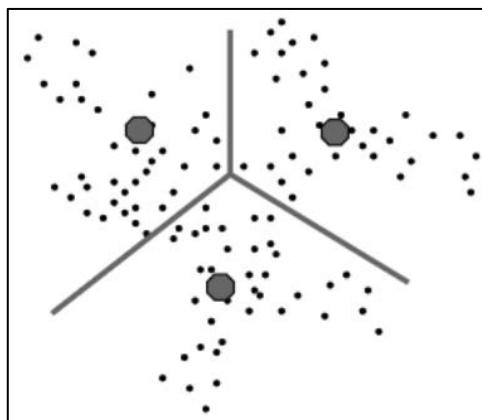
Dimana  $x_i \in cluster$  ke-k dan p adalah banyaknya anggota  $cluster$  ke-k

Secara ringkas algoritma dasar dalam *K-Means* adalah

- a. Tentukan jumlah cluster ( $k$ ), tetapkan pusat *cluster* (*centroid*) sembarang secara acak.
- b. Hitung jarak setiap data ke pusat *cluster*.
- c. Kelompokkan data ke dalam *cluster* yang dengan jarak yang paling pendek ke pusat *cluster*.
- d. Hitung pusat *cluster* yang baru dan Ulangi langkah b sampai dengan d hingga sudah tidak ada lagi data yang berpindah ke *cluster* yang lain.

### **2.8. Bag of Words**

*Bag of Words* atau dalam *computer vision* juga dikenal sebagai *Bag of Visual Words* merupakan sebuah model yang merepresentasikan objek secara global misalnya dalam penelitian ini berbagai fitur citra sebagai bag (*multiset*) fitur tanpa memperdulikan aturan fitur bahkan urutan fitur untuk menjaga keanekaragamannya (Mardiana & Nyoto, 2015). Definisi lain untuk BoW adalah sebuah model yang mempelajari sebuah kosakata/fitur dari seluruh dokumen/citra, lalu memodelkan tiap dokumen/citra dengan menghitung jumlah kemunculan setiap kata/fitur (Trisari et al., 2020). Dalam penelitian ini BoW diterapkan dalam rangkaian proses ekstraksi fitur setelah ekstraksi deskriptor dari SIFT dilakukan. BoW ini dibuat dengan melakukan proses klastering (pada penelitian ini yaitu K-Means) untuk menciptakan beberapa *centroid* yang akan menjadi beberapa fitur dalam sebuah *multiset* fitur. Dengan perhitungan pendekatan kesamaan jarak antara fitur-fitur citra hasil SIFT dengan *centroid* (fitur dalam *multiset*) maka akan terbentuk sebuah elemen list yang berisi jumlah kesamaan fitur dalam *multiset* dengan fitur dalam citra.



**Gambar 2.7 Contoh klastering pada model BoW**

### **2.9. Feature Scaling**

*Feature Scaling* adalah metode yang digunakan untuk menormalkan rentang variabel independen atau fitur data, ini merupakan teknik dalam mengubah nilai numerik dalam dataset ke skala umum, tanpa mendistorsi perbedaan dalam rentang nilai (Ambarwari et al., 2020). Dalam pemrosesan data, ini juga dikenal sebagai normalisasi data dan umumnya dilakukan selama langkah *preprocessing* data, hal ini merupakan salah satu langkah penting dalam *machine learning* sebelum membuat model karena dapat membantu mempercepat proses *training* (Ambarwari et al., 2020). Pada penelitian ini normalisasi tersebut dilakukan

dengan metode *MinMaxScaling* yang dimana menskalakan setiap variabel input secara terpisah ke rentang 0-1, yang merupakan rentang untuk nilai titik-mengambang yang paling presisi. Penggunaan metode ini akan tetap mempertahankan bentuk dataset (tidak ada distorsi). Berikut rumus yang digunakan dalam *MinMaxScaling* yaitu sebagai berikut.

## **2.10. Support Vector Machine**

*Support Vector Machine* (SVM), adalah salah satu algoritma pembelajaran mesin terbaik, yang diusulkan pada 1990-an dan sebagian besar digunakan untuk pengenalan pola (Pradhan, 2012). SVM sendiri merupakan metode learning machine yang bertujuan mendapatkan batas atau jarak terbaik yang dapat memisahkan antara 2 kelas, juga dapat disebut dengan istilah *hyperplane* (Satriyo, 2003). Ini juga telah diterapkan pada banyak masalah klasifikasi pola seperti pengenalan gambar, pengenalan ucapan, kategorisasi teks, deteksi wajah, dan deteksi kartu yang salah, dll. Pengenalan pola bertujuan untuk mengklasifikasikan data berdasarkan pengetahuan apriori atau informasi statistik yang diambil dari data mentah yang merupakan alat yang ampuh dalam pemisahan data di banyak disiplin ilmu. SVM adalah pembelajaran mesin jenis *supevised learning*. algoritma di mana diberikan satu set contoh pelatihan, masing-masing ditandai sebagai milik salah satu dari banyak kategori, algoritma pelatihan SVM membangun model yang memprediksi kategori contoh baru (Pradhan, 2012). SVM memiliki kemampuan yang lebih besar untuk menggeneralisasi masalah, yang merupakan tujuan pembelajaran statistik statistical.

SVM dimodifikasi dalam klasifikasi *non-linear* berdasarkan fungsi kernel. Kernel adalah parameter SVM yang dapat disesuaikan dengan kebutuhan. SVM non-linier digunakan untuk memecahkan permasalahan non-linier dengan menambahkan fungsi Kernel. Pada SVM non-Linier, nilai awal  $x$  yang dimasukkan pada fungsi  $f(x)$  menuju sebuah ruangan vector yang memiliki dimensi lebih tinggi. Pada kasus ini kernel yang digunakan adalah *Radial Basis Function* (RBF) (Kamble & Hegadi, 2017). Kernel RBF ini cocok digunakan dalam kelas yang banyak dan fitur yang banyak. Sehingga algoritma SVM yang dilakukan seperti berikut. Model yang optimal dari himpunan *hyperplanes* di *data training* dihitung dengan algoritma optimasi SVM (Surinta et al., 2015). Berikut rumus kernel RBF yang digunakan.

$$K(x, x') = \exp\left(-\frac{\|x, x'\|^2}{2\sigma^2}\right) \dots \quad (2.16)$$

Nilai parameter yang besar dari  $\gamma$  dapat menyebabkan *overfitting* karena bertambahnya jumlah *support vector*. Dalam kasus klasifikasi *multiclass*, perhitungan dilakukan sebanyak  $K$  kelas sehingga nilai prediksi akhir berbentuk vektor dengan panjang  $K$ . Dan nilai bobot akan sebanyak  $K*n$  fitur.

Algoritma pelatihan SVM ini menggunakan metode sequential learning yang sederhana dan tidak banyak memakan waktu (Harahap et al., 2018). Berikut langkah-langkah pelatihan yang dilakukan (Wijayanti et al., 2018).

1. Inisialisasi pada  $\sigma_1$  lalu parameter selanjutnya, seperti  $\lambda, \gamma, C, \varepsilon, \alpha, \sigma$ .

## Keterangan:

$\sigma$  = sigma.

$\alpha$  = alfa, berfungsi untuk inisialisasi awal support vector.

$\lambda$  = Lamda, konstanta skalar

$\gamma$  = gamma untuk mencari niali kecepatan.

C. = untuk memberi batas nilai toleransi.

$\Sigma$  = epsilon berfungsi menemukan nilai error.

2. Menemukan nilai matrik Hessian dengan menggunakan persamaan.

Dengan  $i, j : 1, \dots, n$ .

## Keterangan:

$x_i$  = data nilai ke-i.

$x_j$  = data nilai ke- $j$ .

$y_i$  = data nilai dari kelas ke- $i$ .

$y_j$  = data nilai dari kelas ke- $j$ .

$n$  = jumlah seluruh sampai data ke n.

$K(x_i, x_j)$  = kernel nilai yang dipakai.

3. Dimulai dari nilai data  $i$  sampai dengan data  $j$ , dijelaskan dalam persamaan dibawah ini:

## Keterangan:

$\alpha_i$  = nilai alfa ke – i.

$D_{ij}$  = Matrik Hessian.

$E_i$  = rata-rata Error.

$$b. \quad \delta\alpha_i = \min(\max[\gamma(1 - E_i), \alpha_i], C - \alpha_i) \dots \quad (2.19)$$

## Keterangan:

$\alpha_i$  = nilai alfa ke-i.

$\gamma$  = gamma untuk mencari niali kecepatan.

$E_i$  = rata-rata Error.

$C$  = untuk memberi batas nilai alfa.

$$c. \quad \alpha_i = \alpha_i + \delta\alpha_i \quad \dots \quad (2.20)$$

## Keterangan:

$\alpha_i$  = alfa nilai ke – i.

$\delta\alpha_i$  = delta alfa nilai ke - i.

4. Proses ke 3 diulang terus sampai mendapatkan nilai iterasi maksimal ( $\delta\alpha_i \leq \varepsilon$ ).
  5. Selanjutnya mendapatkan nilai *support vector*,  $SV = (\alpha_i > thresholdSV)$  misalkan  $threshold SV = 0$ .

Setelah *training* SVM dilakukan maka hasil parameter alfa akan dihasilkan untuk digunakan dalam proses klasifikasi, proses klasifikasi yang dilakukan membutuhkan nilai alfa dan nilai bias, berikut langkah-langkah yang dilakukan.

1. Langkah awal yang dilakukan yaitu mencari nilai bias.

### Keterangan:

$\alpha_i$  = alfa nilai ke-i.

$y_i$  = data nilai dari kelas ke- $i$ .

$m$  = data jumlah dari SV.

$K(x_i, x^-), K(x_i, x^+)$  = kernel nilai yang dipakai.

2. Kemudian dengan nilai alfa dan bias, maka klasifikasi dapat dilakukan dengan langkah berikut.

Dalam penelitian ini, SVM *multiclass* diterapkan dengan metode *one-against-rest* (*one-vs-all*). Metode ini melakukan perlakuan terhadap masing-masing kelas dengan membandingkan seluruh sisa kelassnya, apabila terdapat  $k$  kelas proses klasifikasi dilakukan sebanyak  $k$  kelas tersebut dengan membandingkan kelas pertama dengan seluruh kelas lainnya begitu pula perlakuan yang sama dilakukan pada kelas lainnya. Ketika salah satu kelas akan dilatih maka kelas lainnya akan digabung menjadi nilai negatif dari kelas yang dituju.

## 2.11. Optimasi Parameter Model

Optimasi pada penelitian ini dilakukan dengan menggunakan metode *Grid Search* terhadap parameter-parameter yang berpengaruh dalam performa model SVM, perlakuan ini juga sering disebut dengan *hyperparameter tuning*. *Hyperparameter tuning* adalah proses mencari nilai parameter optimal di mana mula-mula harus menentukan daftar parameter dan rentang pencarian untuk setiap parameter (Andini et al., 2022). *Hyperparameter tuning* ini berfungsi untuk membantu model menemukan nilai parameter yang tepat untuk tiap dataset agar mendapatkan hasil kinerja yang maksimal (Andini et al., 2022). Parameter-parameter tersebut diantaranya yaitu pada ukuran data gambar yang digunakan, jumlah *centroids* pada

*K-Means* klastering saat melakukan perhitungan ekstraksi fitur SIFT deskriptor, parameter C sebagai nilai parameter *regularization* model SVM dan parameter  $\gamma$  sebagai nilai koefisien dari persamaan *gaussian* kernel SVM.

Optimasi parameter ukuran gambar dan jumlah *centroids* dilakukan secara manual dengan mengganti nilai tersebut setiap melakukan proses *modelling* dari awal sampai selesai. Sedangkan optimasi parameter C dan  $\gamma$  dilakukan secara otomatis dengan menggunakan metode *Grid Search CV* yang dapat menelusuri satu persatu dan memeriksa pada kombinasi mana yang menghasilkan nilai terbaik. Grid search memberikan jaminan kepastian karena di setiap kombinasinya mempengaruhi setiap proses yang berjalan (Andini et al., 2022)

## 2.12. Evaluasi

Pada penelitian ini evaluasi dilakukan dengan mencari nilai performa akurasi yang terbaik. Selain itu, *classification report* juga dilakukan untuk melihat ketepatan prediksi model yang dibangun dengan nilai aktualnya. *Classification report* akan memberikan kesimpulan dari performa model pada seluruh kelas dengan mudah seperti diantaranya terdapat nilai *precision*, *recall*, *F1-score*, dan akurasi (Oktaviana & Azhar, 2021). Dalam perhitungannya performa *precision*, *recall*, *F1-score*, dan akurasi dapat dicari dengan menggunakan rumus berikut.

$$precision_k = \frac{TP_k}{TP_k + FP_k} \dots \quad (2.23)$$

$$recall_k = \frac{TP_k}{TP_k + FN_k} \dots \quad (2.24)$$

$$F1\_score_k = 2 \times \frac{recall_k \times precision_k}{recall_k + precision_k} \dots \quad (2.25)$$

$$accuracy = \frac{\text{total\_true\_predict}}{\text{total\_data}} \dots \quad (2.26)$$

## 2.13. Penelitian Sebelumnya

Beberapa penelitian yang telah dilakukan sebelumnya membahas tentang pengenalan tulisan tangan karakter hanacaraka aksara jawa dengan berbagai metode yang diimplementasikan dan penelitian lainnya juga telah membahas pengenalan tulisan tangan dengan objek karakter yang berbeda menggunakan metode SIFT Deskriptor.

Penelitian yang telah diusulkan (Wibowo et al., 2018) dengan menggunakan metode yaitu *Convolutional Neural Network* (CNN) oleh widowo dkk. penelitian ini menguji algoritma CNN yang merupakan salah satu jenis model *discriminative deep-learning* yang sangat luas digunakan untuk klasifikasi berdasarkan *supervised learning*. Dalam melakukan klasifikasi tulisan tangan penelitian ini menggunakan dataset yang sangat besar berjumlah sebanyak 11500 total karakter dengan 575 karakter tiap jenis karakter hanacaraka yang berjumlah 20 buah. *Dataset* tersebut diolah dengan membuat dua *modelling extraction* yaitu Model 1 memiliki 3 2D *Convolution Layer*, 3 *pooling layer*, 1 *fully connected layer* dan Model 2 hanya memiliki 1 *fully connected layer*. Kemudian untuk mengurangi *error* dan mengatur parameter seperti *momentum*, *learning rate*, *regularization method*, dan *activation function* menggunakan *Stochastic Gradient Descent algorithm*. Hasil penelitian

menunjukkan angka yang memuaskan dengan tingkat akurasi sebesar 94.57%, presisi 94.75%, recall 94.57%, dan F1 score 94.66%.

Penelitian lainnya (Dewa et al., 2018) diusulkan juga menggunakan CNN dan MLP. Pada penelitian ini perangkat lunak yang dikembangkan memanfaatkan deteksi kontur dan deteksi tepi *Canny* menggunakan pustaka OpenCV terhadap citra karakter Aksara Jawa untuk proses segmentasi. Data yang digunakan sebanyak 2000 karakter hanacaraka dan Modul CNN selanjutnya melakukan proses klasifikasi terhadap citra yang telah disegmentasi ke dalam 20 kelas berukuran 28x28 piksel. Untuk evaluasi, kinerja CNN dibandingkan dengan kinerja dari model *Multilayer Perceptron* (MLP) dari sisi akurasi klasifikasi dan waktu latih. Hasil pengujian menunjukkan akurasi dari model CNN mampu mengungguli akurasi dari model MLP yaitu sebesar 89% meskipun CNN membutuhkan waktu latih yang lebih lama dibandingkan dengan MLP.

Rismiyati, et al. (2017) juga melakukan penelitiannya menggunakan CNN dan DNN. Klasifikasi dilakukan dengan menggunakan 2470 gambar dataset karakter hanacaraka dengan input gambar berdimensi 32x32 piksel, selanjutnya dilakukan pengujian model dengan 10 *fold cross validation* menghasilkan performa akurasi yang cukup baik sebesar 70.22% untuk model CNN dan 64.41% untuk model DNN.

Selain menggunakan CNN terdapat penelitian dari (Sari et al., 2018) dengan objek yang sama pada penelitian sebelumnya namun menggunakan metode KNN dengan *Feature Extraction Roundness* dan *Eccentricity*. Meskipun metode KNN yang dikenal sebagai metode klasifikasi dengan akurasi lebih kecil dari CNN namun pada penelitian kekurangan dari algoritma KNN dapat ditutupi dengan menggunakan fitur ekstraksi *Roundness* dan *Eccentricity*. Sejumlah 240 data training yang digunakan pada penelitian dengan beberapa tahapan preprocessing yaitu proses *cropping*, *binary*, *converting* kedalam *negative image*, *filtering* dengan *median filter* dan terakhir *dilation*. Dengan pengukuran jarak antar data menggunakan KNN dengan nilai K sebesar 3 menghasilkan akurasi yang cukup optimal yaitu sebesar 87.5%.

Penelitian pada objek hanacaraka aksara jawa berikutnya diusulkan oleh (Rismiyati et al., 2018) menggunakan algoritma *Support Vector Machine* dengan HOG dan *Zone Base Features*. Data yang digunakan pada penelitian ini sebanyak 2459 dan kemudian melakukan *preprocessing* dan *skeletonization* untuk menghilangkan ruang putih di sekitar gambar, mengonversi menjadi gambar biner, dan mendapatkan kerangka objek biner. Performa dari fitur HOG dan *Zoning* tersebut akan dibandingkan untuk pengklasifikasian karakter hanacaraka. Dua jenis input akan digunakan untuk masing-masing ekstraktor fitur, biner dan kerangka citra karakter. Hasil percobaan menunjukkan bahwa fitur HOG mampu menunjukkan akurasi yang lebih tinggi dibandingkan dengan fitur berbasis zona sederhana (88,45%). Akurasi terbaik untuk HOG dicapai dengan menggunakan input biner. Di sisi lain, meskipun sederhana, fitur berbasis zona mampu mencapai akurasi 81,98% dengan menggunakan input kerangka.

Widiarti & Wastu (2009) mengusulkan penelitian mengenai pengenalan tulisan tangan aksara jawa dengan menggunakan metode *Hidden Markov Model* (HMM). Penelitian

menggunakan dataset yang cukup kecil yaitu sejumlah 1000 *record* (800 *data training* dan 200 *data testing*). Beberapa tahapan yang dilakukan yaitu *preprocessing* yang meliputi *filtering background noise*, transformasi gambar menjadi *binary image*, dan *resize image* menjadi 72x72 *pixel*; kemudian fitur ekstraksi dilakukan dengan menggunakan ekstraksi *Horizontal & Vertical Vector* dengan memecahkan menjadi empat percobaan yaitu membagi karakter ke dalam 1 *horizontal vector* (1H), 2 *horizontal vector* (2H), 1 *vertical vector* (1V), dan 2 *vertical vector* (2V); selanjutnya *modelling* data menggunakan HMM dengan jumlah *state* yang bervariasi 15-22 *state*; terakhir pada *testing* yang dilakukan dengan menggunakan *k-fold cross validation* untuk mencari akurasi yang paling optimal. Akurasi optimal yang didapatkan pada penelitian ini sebesar 85.7% dengan menggunakan 16 *state* dan fitur ekstraksi 1V.

Rasyidi et al. (2021) juga telah mengusulkan pengenalan tulisan tangan aksara jawa dengan menggunakan metode *Random Forest*. Pada penelitian ini data yang diambil sebesar 6000 gambar dan dibagi menjadi dua yaitu 70% *data training* dan 30% *data testing*, kemudian data tersebut dilakukan proses *data augmentation* dengan kombinasi *rotation image* dan *shear image* sehingga menghasilkan data sebesar 21000 data training dan 9000 data testing. Kemudian pada *preprocessing* dilakukan *binarization images*, *cropping*, dan *resizing* menjadi 32x32 piksel. Selain tiga hal tersebut terdapat proses tambahan yaitu kombinasi terhadap *thinning* dan HOG. Pada *training model* dengan menggunakan *random forest* beberapa parameter yang digunakan akan diuji dengan metode *grid search* dan *3-fold cross validation* pada parameter *Impurity Measure* (gini, entropy) dan jumlah tree (bervariasi 200-2000). Terakhir tahap uji dilakukan menggunakan *data testing* menghasilkan nilai yang optimal yaitu *accuracy* 97%, *precision* 97%, dan *recall* 97% dengan kombinasi tanpa menggunakan proses *thinning* dan HOG serta nilai parameter *impurity measure* yaitu gini dan jumlah *tree* yaitu 1800.

Metode KNN dan HOG juga dapat dilakukan dalam pengenalan tulisan tangan aksara jawa (Susanto et al., 2021). Penelitian ini menggunakan 1000 data citra karakter aksara jawa yang dibagi menjadi 20 kelas. Proses *preprocessing* yang dilakukan yaitu *grayscale*, *tresholding*, *median filter*, dan *size normalization*, selanjutnya fitur ekstraksi diterapkan menggunakan HOG. Proses *training model* menggunakan metode KNN dengan nilai K yang paling optimal adalah 1 serta rasio dari pemisahan *dataset training* dan *testing* yaitu 80:20, sehingga menghasilkan akurasi terbaik sebesar 98,5%. Tahap *testing* dilakukan dengan melakukan kombinasi nilai K dan rasio pemisahan *dataset*. Perbandingan hasil model KNN-*Median Filter-HOG* yang diperoleh meningkat sekitar 4% daripada model yang tidak menggunakan fitur ekstraksi dan *median filter*.

Kemudian terdapat penelitian lainnya yang sedikit berbeda menggunakan *Optical Character Recognition* dengan bantuan *Tesseract Tools* (Robby et al., 2019). Penelitian ini mengambil dua tipe data yang akan digunakan yaitu 5880 data tulisan tangan dan 260 data digital, data tersebut terdiri dari 120 kelas dari karakter aksara jawa yang dilengkapi dengan masing-masing tanda baca ‘a-i-u-e-o-é’ (sandangan). Kedua asal data tersebut dikombinasikan sebagai proses *training* yang dilakukan. Beberapa proses *preprocessing* diterapkan dalam penelitian ini yaitu rotasi citra, pengisian label, menghilangkan *noise*, dan

*sharpening*. Kemudian proses *training* dilakukan menggunakan teknik *Neural-Network API* dari *Tesseract OCR Tool*. Akurasi tertinggi (97,50%) yang dicapai oleh model dicapai dengan menggabungkan *single boundary box* untuk seluruh bagian karakter dan *separate boundary boxes* di karakter dan bagian sandangannya.

Penilitian berikutnya dengan objek yang berbeda yaitu pada aksara sasak (tidak berbeda jauh dengan aksara jawa) diusulkan oleh Yulianti et al. (2019) menggunakan SVM dan *Moment Invariant* sebagai ekstraksi fitur. Penelitian ini menggunakan *dataset* sebesar 2700 data yang dibagi menjadi 1800 data latih dan 900 data uji. Dalam *preprocessing image* beberapa hal yang dilakukan yaitu *greyscaling*, *binarization*, *cropping*, *resizing* menjadi 28x28 piksel, dan *thinning*. Kemudian Moment Invariant melakukan fungsi *non-linear* yang *invariant* terhadap rotasi, translasi dan skala dalam *moment* geometri citra yang menghasilkan tujuh nilai *moment invariant*. Modelling data dilakukan dengan menggunakan metode SVM *Tree Structure* untuk mengenali 18 kelas aksara sasak. Terakhir pada pengujian dilakukan dengan menggunakan *k-fold cross validation* dengan hasil akurasi yang optimal sebesar 92,52% dengan menggunakan 112 fitur dari *moment invariant*.

Selain pada penelitian menggunakan objek karakter hanacaraka, terdapat penelitian lainnya dengan objek berbeda yang dilakukan oleh (Surinta et al., 2015) yang menggunakan metode KNN dan SVM dengan menggunakan *Local Gradient Feature* pada beberapa objek karakter Thai, Bangla, dan Latin. *Local Gradient Feature* terdiri dari dua fitur yaitu HOG dan SiftD. *Dataset* yang digunakan pada penelitian ini berupa 68 kelas karakter Thai 13130 *data training* dan 1360 *data testing*, 10 kelas digit Thai 8055 *data training* dan 1500 *data testing*, 45 kelas karakter Bangla 4627 *data training* dan 900 *data testing*, 10 kelas digit Bangla 9161 *data training* dan 1500 *data testing*, 25 kelas karakter Latin 26392 *data training* dan 11287 *data testing*, 10 kelas digit Latin 1637 *data training* dan 880 *data testing*. Sebelum melakukan klasifikasi dataset dilakukan proses *preprocessing* berupa gambar tulisan tangan diubah menjadi skala *greyscale* dan dinormalisasi menjadi gambar ukuran tetap. Ada dua resolusi ruang piksel yang digunakan dalam penelitian ini yaitu ruang piksel 28x28 dan 36x36. Hasilnya menunjukkan bahwa deskriptor fitur gradien lokal secara signifikan mengungguli secara langsung menggunakan intensitas piksel dari gambar. Fitur HOG dan siftD memberikan performa yang sangat baik dan signifikan dengan akurasi mendekati 100%. Ketika deskriptor fitur yang diusulkan digabungkan dengan SVM, akurasi yang sangat tinggi diperoleh pada kumpulan data tulisan tangan Thailand (karakter dan angka), kumpulan data tulisan tangan Latin (karakter dan angka), dan kumpulan data angka tulisan tangan Bangla.

Penelitian lainnya yang berbeda dilakukan oleh Narang et al. (2020) dengan melakukan pengenalan tulisan tangan pada karakter Devanagari menggunakan SIFT dan *Gabor Filter* sebagai teknik fitur ekstraksi serta SVM sebagai teknik klasifikasinya. Pada penelitian ini menggunakan dataset karakter Devanagari sejumlah 5484 sampel dengan 33 kelas karakter. Dari data karakter tersebut dinormalisasi ke ukuran karakter 64x64 piksel dengan menggunakan metode *nearest neighbourhood interpolation* untuk pemrosesan dan pengenalan lebih lanjut. Selanjutnya fitur ekstraksi dilakukan terpisah oleh SIFT dan Gabor untuk dibandingkan, terakhir klasifikasi dilakukan menggunakan SVM dengan kernel yang

beragam (linier, poly, RBF, dan sigmoid). Hasil penelitian tersebut dilakukan dengan beberapa testing dari variasi parameter yang dimasukkan juga menggunakan *k-fold cross validation* dan menunjukkan bahwa nilai performa akurasi terbaik diperoleh pada kombinasi metode dari penggunaan *5-fold cross validation*, Gabor, dan poly-SVM dengan nilai akurasi sebesar 91.39%. Disisi lain penggunaan SIFT mendapatkan akurasi awal sebesar 65.97% menggunakan RBF-SVM.

Kemudian, penelitian berbeda mengenai pengenalan kata tulisan tangan bahasa Arab menggunakan SIFT dan SVM dilakukan oleh Hassan et al. (2019). Penelitian tersebut menggunakan data kata-kata dalam bahasa Arab sebanyak 2072 sampel data latih dan 868 sampel data uji. Proses penelitian ini diawali dengan *preprocessing data* yaitu melakukan konversi warna RGB pada citra menjadi warna abu-abu, kemudian fitur ekstraksi dilakukan dengan menggunakan metode pendekatan SIFT, selanjutnya sebelum klasifikasi dilakukan metode *K-Means* dan FINN akan diterapkan sebagai *feature selection* pada data input sehingga fitur yang digunakan menjadi lebih baik dari yang sebelumnya, tahap terakhir dilakukan klasifikasi data dengan menggunakan SVM. Hasil penelitian menunjukkan bahwa dengan menggunakan SIFT dan SVM serta sedikit bantuan *feature selection* memperoleh nilai akurasi yang tinggi yaitu sebesar 99.08%.

Dengan metode KNN-SIFT juga dapat diusulkan dalam penelitian mengenai *Document Script Identification* (Rajput & Ummapure, 2017). Penelitian ini bertujuan untuk mengenali bahasa dalam sebuah dokumen tulisan tangan diantaranya bahasa Inggris, Kanada, dan Devanagari (Hindi). Terdapat 240 data dokumen Kanada, 240 Hindi, dan 240 Inggris digunakan dalam melakukan *training* model KNN. Sebelumnya data gambar tersebut dilakukan *preprocessing* yaitu segmentasi baris/kata pada kalimat dalam dokumen, menghilangkan *noise* dengan *median filter*, normalisasi ukurang gambar menjadi 256x256 piksel, dan *binarization*. Pada tahap testing dilakukan tiga tipe yaitu pada 210 teks 2 kata, 210 teks 3 kata, dan 300 teks lebih dari 3 kata, hasil pengujian menunjukkan model yang paling optimal yaitu KNN dengan nilai K adalah 1 memiliki akurasi sebesar 97,65% klasifikasi 2 bahasa dan 96,71% klasifikasi 3 bahasa. Rata-rata nilai akurasi mencapai 97,18%.

Sriwathsan et al. (2020) mengusulkan penelitiannya tentang *Signature Recognition* menggunakan metode SVM dan fitur SIFT/SURF. Penelitian ini menggunakan 1600 *signature* data dengan pemisahan 800 *data training* dan 800 *data testing*. *Preprocessing* yaitu diterapkan pada gambar data yaitu *binarization*, *noise removal*, dan *boundary extraction*. Kemudian ekstraksi fitur dicoba pada SIFT dan SURF untuk melihat perbandingan hasil yang diperoleh, tidak hanya itu *K-Means*, *codebook generation*, dan *quantization* data juga dilakukan setelahnya. Terakhir tahapan *modelling* dilakukan dengan SVM. *Testing* dilakukan dengan menggunakan *10-fold cross validation* dan melakukan *tuning parameter* dari SVM serta *K-Means*. Hasil terbaik diperoleh pada SIFT mendapat akurasi sebesar 68% sedangkan SURF memperoleh akurasi sebesar 96,87%, hasil tersebut diperoleh dari beberapa percobaan yang dilakukan.

Berikutnya oleh Wang et al. (2015) mengusulkan penelitian tentang *License Plate Recognition* hanya menggunakan SIFT. Penelitian ini mengambil 700 gambar *data training* dan 100 gambar *data testing*. Selain menggunakan SIFT terdapat teknik lainnya yaitu *candidate filtration*, *tilt correction*, *character segmentation*, dan *character recognition*. Semua teknik tersebut akan dibungkus dengan metode fitur Sift. Hasil penilitian yang dilakukan memperoleh performa akurasi pada *chinese character recognition* 95,4%, *noise region recognition* 100%, dan total keseluruhan pengenalan 96,0%.

Mirip dengan sebelumnya penelitian yang diusulkan oleh Doush & AL-Btoush (2017) menggunakan metode SIFT sebagai metode klasifikasi dari *Currency Recognition*. Penelitian ini menggunakan 100 *data training* dan 400 *data testing* yang diambil dari kamera *handphone* sebanyak 20 kelas uang kertas maupun logam. Data citra dilakukan kompresi gambar dan *crop background* sebelum *training*. Penelitian ini menguji dua kasus yaitu menggunakan pengenalan pada gambar yang berwarna dan pengenalan pada gambar yang hitam putih. Hanya dengan menggunakan SIFT hasil performa akurasi penelitian menunjukkan pada gambar berwarna yaitu 71% uang kertas, 25% uang logam dan pada gambar hitam putih yaitu 53% uang kertas, 20% uang logam.

*Face Recognition* menggunakan SIFT & SURF diusulkan oleh Gupta et al. (2020). Penelitian ini menggunakan beberapa *dataset* yang berasal dari Yale2B, Face 94, M2VTS, ORL, dan FERET dengan rasio pembagian *data training* dan *data testing* sebesar 80%:20%. Beberapa *preprocessing* dilakukan yaitu *face edge detection*, *segmentation*, dan *localization*. Selanjutnya fitur ekstraksi oleh SIFT dan SURF dengan beberapa kombinasi percobaan pada ukuran dimensi vektor yaitu 32, 64. Terakhir klasifikasi hasil gambar menggunakan metode *Decision Tree & Random Forest*. Pengujian mengambil performa pada nilai *accuracy*, *true-positive rate*, *false-positive rate*, dan *area under the curve*. Hasil akurasi terbaik yang diperoleh yaitu 99,7% pada kombinasi metode SIFT(64) + SURF(32) *Random Forest* dan *dataset* Yale2B.

Penelitian lainnya mengenai pengenalan tulisan tangan dengan objek karakter Marathi diusulkan oleh Kamble & Hegadi (2017). Penelitian ini menggunakan data yang besar yaitu 17271 data angka yang berasal dari U. Bhattacharya dan B.B. Chaudhuri kemudian 31320 jumlah data miliki sendiri, dengan kombinasi dari 4800 huruf vokal, 6400 huruf konsonan, 20120 angka. *Preprocessing* dan ekstraksi fitur dilakukan dengan menghitung *area*, *perimeter*, *eccentricity*, *orientation*, dan nilai *Euler* berdasarkan komponen piksel citra. Selanjutnya klasifikasi dilakukan oleh metode SVM dan KNN dengan variasi *5-fold validation*. Hasil performa akurasi yang diperoleh menggunakan data milik sendiri sebesar 88,52% SVM dan 80,25% KNN.

Penelitian terakhir yang diusulkan mengenai pengenalan huruf dan angka menggunakan *Hybrid DWT-DCT* dengan metode klasifikasi KNN dan SVM (Ghadekar et al., 2018). Penelitian ini dilakukan secara terpisah dimana pertama klasifikasi dilakukan pada angka dengan jumlah data 60000 *training* dan 10000 *testing*, sedangkan pada klasifikasi huruf memiliki jumlah data 128000 *training* dan 20800 *testing*. Fitur ekstraksi dilakukan dengan menggabungkan dua metode DWT untuk mendapatkan nilai

*approximation matrix* dan DCT untuk mendapatkan nilai koefisien kemudian hasil fitur disimpan dan digunakan dalam klasifikasi. Klasifikasi menggunakan metode KNN dengan nilai K yaitu 5 dan SVM. Hasil klasifikasi pertama pada digit memperoleh akurasi 97,33% KNN dan 97,74% SVM kemudian klasifikasi kedua pada huruf memperoleh akurasi 88,56% KNN dan 89,51% SVM.

Dari penelitian-penelitian yang telah dijabarkan sebelumnya, maka dapat diringkas menjadi suatu tabel state of the art yang dapat dilihat pada tabel *State of The Art* berikut ini.

**Tabel 2.1 State of The Art**

No	Penulis	Judul	Metode	Hasil
1	Wibowo et al. (2018)	<i>Handwritten Javanese Character Recognition using Descriiminative Deep Learning Technique</i>	CNN	Hasil penelitian menggunakan metode CNN menunjukkan angka yang memuaskan dengan tingkat akurasi sebesar 94.57%, presisi 94.75%, <i>recall</i> 94.57%, dan <i>F1 score</i> 94.66%.
2	Dewa et al. (2018)	<i>Convolutional Neural Networks for Handwritten Javanese Character Recognition</i>	CNN dan MLP	Hasil pengujian menunjukkan akurasi dari model CNN mampu mengungguli akurasi dari model MLP yaitu sebesar 89% meskipun CNN membutuhkan waktu latih yang lebih lama dibandingkan dengan MLP.
3	Sari et al. (2018)	<i>Roundness and Eccentricity Feature Extraction for Javanese Handwritten Character Recognition based on K-Nearest Neighbor</i>	KNN dengan <i>Feature Extraction Roundness</i> dan <i>Eccentricity</i>	Dengan menggunakan dataset yang cukup kecil sebesar 240 data klasifikasi KNN yang dibantu oleh <i>Feature Extraction Roundness</i> dan <i>Eccentricity</i> dengan nilai K sebesar 3 menghasilkan akurasi yang cukup optimal yaitu sebesar 87.5%.
4	Rismiyati et al. (2018)	<i>HOG and Zone Base Features for Handwritten Javanese Character Classification</i>	SVM using HOG or Zone Base Features	Hasil percobaan menunjukkan bahwa fitur HOG mampu menunjukkan akurasi yang lebih tinggi dibandingkan dengan fitur berbasis zona sederhana (88,45%). Di sisi lain, meskipun sederhana, fitur berbasis zona mampu mencapai akurasi 81,98%.
5	Widiarti & Wastu (2009)	<i>Javanese Character Recognition Using Hidden Markov Model</i>	HMM dengan fitur ekstraksi <i>Horizontal &amp; Vertical Vector</i>	Hasil dari penelitian menunjukkan bahwa akurasi yang optimal didapatkan dari jumlah <i>state</i> yaitu 16 dan fitur ekstraksi pada 1 <i>vertical vector</i> (1V) sehingga meraih akurasi sebesar 85,7%.
6	Yulianti et al. (2019)	Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode <i>Moment Invariant</i> dan <i>Support Vector Machine</i>	SVM dengan <i>Moment Invariant</i>	Hasil penelitian dengan melakukan <i>preprocessing</i> ( <i>greyscaling</i> , <i>binarization</i> , <i>cropping</i> , <i>resizing</i> 28x28, dan <i>thinning</i> ), <i>moment invariant</i> sebanyak 112 fitur, dan SVM mendapatkan hasil akurasi optimal sebesar 92,52%.
7	Rasyidi et al. (2021)	<i>Classification of handwritten Javanese script using random forest algorithm</i>	<i>Random Forest</i>	Hasil penelitian menunjukkan bahwa dengan menggunakan data yang banyak sebesar 21000 <i>data training</i> dan 9000 <i>data testing</i> menghasilkan nilai akurasi, presisi, dan <i>recall</i> yang tinggi sebesar 97,7% tanpa menggunakan proses <i>thinning</i> dan HOG serta parameter <i>impurity measure</i> yaitu <i>gini</i> dan jumlah <i>tree</i> yaitu 1800.

**Tabel 2.2 State of The Art Lanjutan**

No	Penulis	Judul	Metode	Hasil
8	Rismiyati et al. (2017)	<i>Deep Learning for Handwritten Javanese Character Recognition</i>	CNN & DNN	Klasifikasi berhasil dilakukan dengan menggunakan 2470 dataset ukuran 32x32 piksel memperoleh akurasi 70,22% model CNN dan 64.41% model DNN.
9	Susanto et al. (2021)	<i>Histogram of Gradient in K-Nearest Neighbor for Javanese Alphabet Classification</i>	KNN & HOG	Peningkatan akurasi terjadi sekitar 4% ketika menggunakan model KNN-median filter-HOG pada 1000 data karakter aksara jawa. Akurasi tertinggi mencapai 98,5% dengan nilai K adalah 1 dan rasio pembagian dataset 80:20.
10	Robby et al. (2019)	<i>Implementation of Optical Character Recognition using Tesseract with the Javanese Script Target in Android Application</i>	Neural Network API Tesseract OCR	OCR dengan menggunakan data tulisan tangan dan data digital pada aksara jawa menerapkan API Tesseract OCR menghasilkan nilai akurasi tertinggi 97,5%. Beberapa preprocessing yang dilakukan yaitu rotasi, <i>labelling</i> , <i>noise removal</i> , dan <i>sharpening</i> .
11	Surinta et al. (2015)	<i>Recognition of handwritten characters using local gradient feature descriptors</i>	KNN dan SVM menggunakan Local Gradient Feature	Hasil menunjukkan bahwa deskriptor fitur gradien lokal secara signifikan mengungguli secara langsung menggunakan intensitas piksel dari gambar. Fitur HOG dan SIFT memberikan performa yang sangat baik dan signifikan ketika digabungkan dengan metode SVM dengan akurasi mendekati 100%.
12	Narang et al. (2020)	<i>On the recognition of Devanagari ancient handwritten characters using SIFT and Gabor features</i>	SVM dengan SIFT dan Gabor	Dengan dataset sebesar 5484 sampel untuk 33 kelas karakter Devanagari, preprocessing hanya normalisasi ukuran gambar, fitur ekstraksi SIFT dan Gabor, serta klasifikasi SVM memperoleh nilai akurasi sebesar 91,39% untuk Gabor & Poly-SVM dan 65,97% untuk SIFT & RBF-SVM.
13	Hassan et al. (2019)	<i>Arabic Handwriting Word Recognition Based on Scale Invariant Feature Transform and Support Vector Machine</i>	SVM dan SIFT	Hasil penelitian yang dilakukan dengan menggunakan 2072 data latih dan beberapa tahapan yaitu <i>preprocessing</i> untuk konversi warna, fitur ekstraksi menggunakan SIFT, bantuan <i>feature selection</i> K-Means & FINN, dan klasifikasi menggunakan SVM menghasilkan akurasi tinggi sebesar 99.08%.
14	Rajput & Ummapure (2017)	<i>Script Identification from Handwritten Documents using SIFT Method</i>	KNN & SIFT	Topik <i>Script Identification</i> berhasil dilakukan dengan menggunakan KNN-SIFT dengan 3 bahasa sebagai target (Inggris, Kanada, Devanagari). Dengan parameter K adalah 1 memperoleh akurasi sebesar 97,65% untuk 2 bahasa, 96,71% untuk 3 bahasa, dan 97,18% rata-rata semuanya.
15	Sriwathsan et al. (2020)	<i>Offline Handwritten Signature Recognition Based on SIFT and SURF Features Using SVMs</i>	SVM & SIFT/ SURF	1600 signature data berhasil dilakukan pengenalan dengan metode SVM dan SIFT/SURF sebagai ekstraksi fitur. Pada SIFT memperoleh akurasi 68% sedangkan SURF 96,87% dengan <i>10-fold cv</i> dan <i>tuning parameter</i> . Sebelumnya juga dilakukan <i>preprocessing binarization</i> , <i>noise removal</i> , <i>boundary extraction</i> .

**Tabel 2.3 State of The Art Lanjutan**

No	Penulis	Judul	Metode	Hasil
16	Wang et al. (2015)	<i>License plate recognition based on SIFT feature</i>	SIFT	Hasil penelitian mengenai <i>License Plate Recognition</i> daerah Cina dengan hanya menggunakan metode SIFT dan 800 data memperoleh akurasi pada <i>chinese character recognition</i> 95,4%, <i>noise region recognition</i> 100%, dan total keseluruhannya 96,0%.
17	Doush & AL-Btoush (2017)	<i>Currency recognition using a smartphone: Comparison between color SIFT and gray scale SIFT algorithms</i>	SIFT	Hanya dengan menggunakan SIFT pengenalan uang berhasil dilakukan dengan tingkat akurasi pada gambar berwarna yaitu 71% uang kertas, 25% uang logam dan pada gambar hitam putih yaitu 53% uang kertas, 20% uang logam.
18	Gupta et al. (2020)	<i>2D-human face recognition using SIFT and SURF descriptors of face's feature regions</i>	<i>Decision Tree &amp; Random Forest, SIFT &amp; SURF</i>	Hasil penelitian <i>face recognition</i> memperoleh akurasi terbaik 99,7% pada <i>dataset</i> Yale2B dengan kombinasi SIFT(64) + SURF(32) dan <i>Random Forest. Preprocessing</i> dilakukan yaitu <i>face edge detection, segmentation, and localization</i> .
19	Kamble & Hegadi (2017)	<i>Comparative Study of Handwritten Marathi Characters Recognition Based on KNN and SVM Classifier</i>	SVM & KNN	Penelitian pada pengenalan karakter Marathi dengan jumlah data 31320 gabungan dari seluruh huruf memperoleh hasil akurasi sebesar 88,52% SVM dan 80,25% KNN. Ekstraksi fitur juga dilakukan yaitu menghitung <i>eccentricity, orientation</i> dan <i>area</i> .
20	Ghadekar et al. (2018)	<i>Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM Classifier</i>	SVM & KNN, Hybrid DWT-DCT	Hasil penelitian klasifikasi angka dan huruf berhasil dengan data yang sangat besar, dengan bantuan fitur ekstraksi dari <i>Hybrid DWT-DCT</i> metode klasifikasi KNN dan SVM dapat berjalan dengan baik, sehingga memperoleh nilai akurasi pada klasifikasi pertama (digit) 97,33% KNN dan 97,74% SVM, sedangkan pada klasifikasi kedua (huruf) 88,56% KNN dan 89,51% SVM.

Berdasarkan tinjauan penelitian-penelitian sebelumnya, maka penelitian ini akan memiliki kesamaan dalam hal tema penelitian yaitu pengenalan tulisan tangan dan dengan objek yang sama yaitu pada karakter hanacaraka aksara jawa. Penelitian ini akan memiliki perbedaan dari sisi metode yang digunakan, serta hasil dari penelitian.

Beberapa perbedaan dengan penelitian sebelumnya terbagi menjadi dua yaitu pertama dengan metode yang berbeda yang digunakan seperti CNN, HMM, KNN, SVM, dan lainnya dengan beberapa fitur ekstraksi tersendiri sendiri (Dewa et al., 2018) (Rasyidi et al., 2021) (Rismiyati et al., 2018) (Sari et al., 2018) (Widiarti & Wastu, 2009) (Susanto et al., 2021), dan yang kedua menggunakan metode fitur ekstraksi yang sama namun berbeda dalam objek dan metode klasifikasi yang digunakan seperti dari (Surinta et al., 2015) dengan objek Thai, Bangla, Latin metode KNN & SVM, lalu dari (Hassan et al., 2019) dengan objek Arabic metode SVM, kemudian dari (Narang et al., 2020) dengan objek Devanagari metode SVM, dan terakhir dari (Sriwathsan et al., 2020) dengan objek tanda tangan metode SVM.

Dari beberapa perbandingan penelitian sebelumnya, terdapat permasalahan pada pengenalan tulisan tangan karakter hanacara terhadap jumlah *data training* yang digunakan, tingkat akurasi yang diperoleh dan tingkat kompleksitas suatu model, sehingga penelitian ini

menggunakan metode SVM yang merupakan metode paling efektif dalam klasifikasi, memiliki akurasi yang cukup tinggi (Thamilselvana & Sathiaseelan, 2015), tidak memiliki masalah dalam *overfitting* (Rajesh et al., 2016), dan tidak membutuhkan jumlah *dataset* yang sangat besar (Rismiyati et al., 2018).

Namun hanya dengan metode SVM sulit untuk meraih akurasi klasifikasi yang tinggi diatas 90%, oleh sebab itu penelitian ini menggunakan metode fitur ekstraksi tambahan yaitu SIFT untuk membantu meningkatkan akurasi klasifikasi, metode SIFT ini telah dilakukan oleh (Surinta et al., 2015) (Hassan et al., 2019) (Narang et al., 2020) dan terbukti memiliki akurasi yang cukup tinggi.

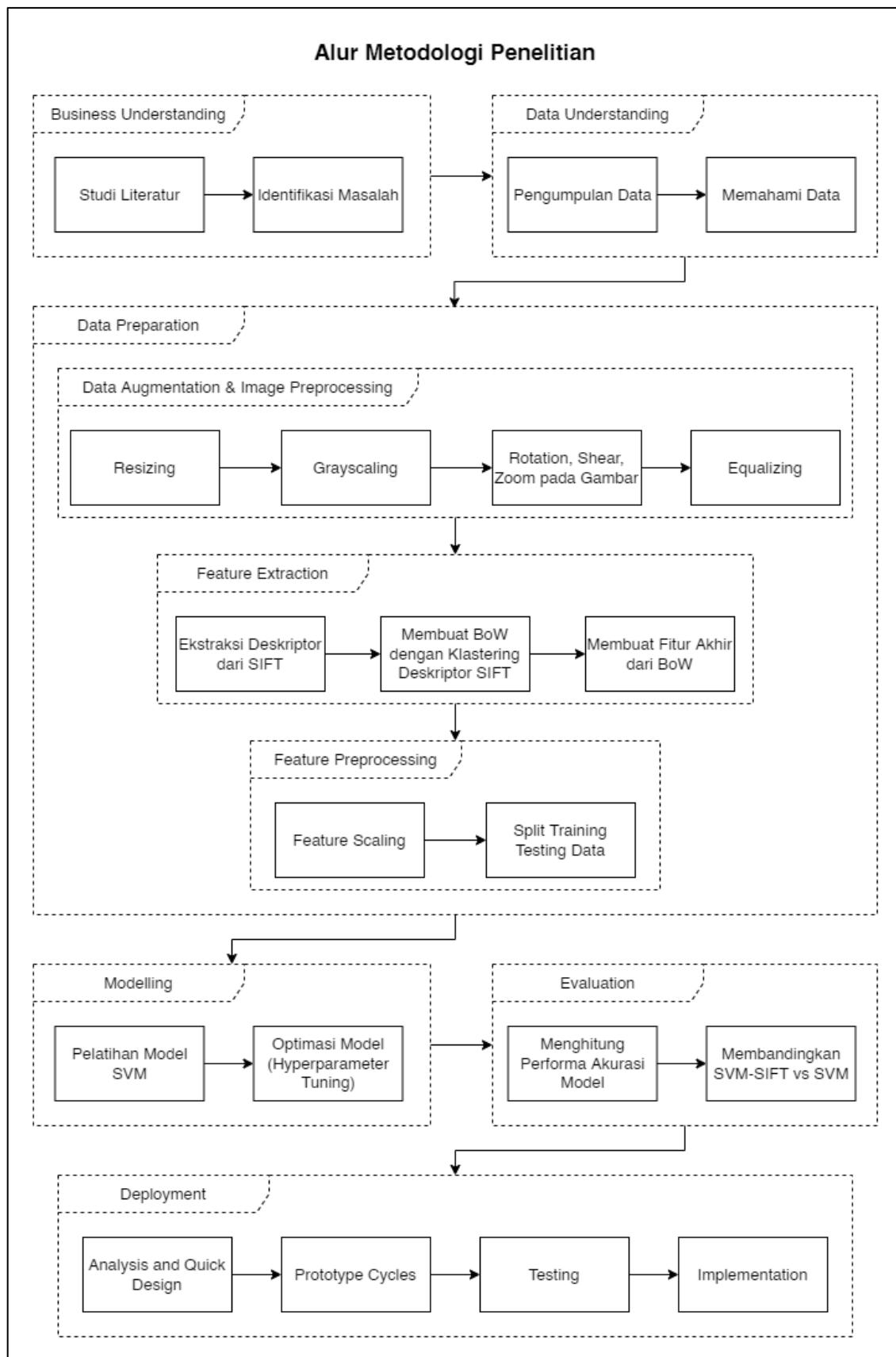
Sehingga pada penelitian ini pengenalan tulisan tangan pada karakter hanacaraka aksara jawa akan menerapkan metode SVM dengan bantuan SIFT dalam meningkatkan performa akurasi. Dari metode tersebut akan dibandingkan dan menemukan bagaimana pengaruh terhadap akurasi jika metode SIFT diterapkan dalam ekstraksi fitur.

## **BAB III**

### **METODOLOGI PENELITIAN DAN PENGEMBANGAN SISTEM**

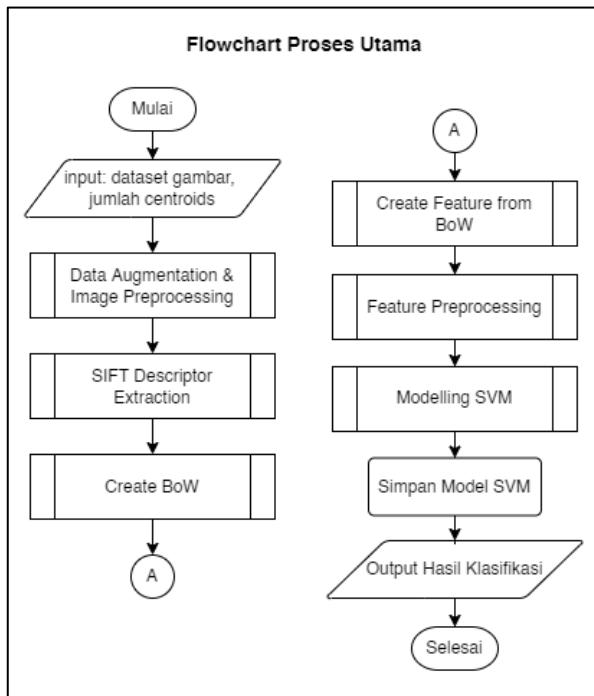
Pada penelitian ini terdapat metodologi penelitian dengan beberapa tahapan secara sistematis dengan menerapkan metode CRISP-DM merujuk pada penelitian oleh Rasyidi, et al. (2021), Dawa, et al. (2018), dan Schröera, et al. (2021) yang konten dari tahapan itu sendiri telah disesuaikan dengan kebutuhan dari penelitian ini. Adapun tahapan tersebut dipecah menjadi enam tahapan.

Berawal dari *business understanding* yaitu sebagai studi literatur berkaitan dengan penelitian ini dalam cakupan yang sama atau setidaknya mirip dengan penelitian ini. Kemudian *data understanding* sebagai proses pengumpulan data dan memahami informasi data tersebut seperti melihat distribusi penyebaran data, melakukan pemeriksaan terhadap data yang akan digunakan apakah data tersebut valid atau tidak secara umum, dan bagaimana langkah-langkah dalam memperbaiki data-data tersebut agar dapat diolah dan dihitung dalam model *machine learning*. Selanjutnya tahapan *data preparation* yang merupakan beberapa proses persiapan data seperti augmentasi data untuk melakukan pembuatan dan pemproduksi data gambar, *image preprocessing* dilakukan untuk membuat data gambar menjadi lebih rapi dan bagus sebelum diolah lebih lanjut, fitur ekstraksi oleh SIFT dilakukan untuk mengambil fitur-fitur penting dalam sebuah gambar tersebut menjadi data input dalam sebuah model *machine learning*, *feature scaling* yang dilakukan dengan metode standarisasi panjang nilai sebuah data dalam fitur *dataset* yang berguna untuk meningkatkan performa pelatihan model, dan terakhir dalam tahapan ini yaitu *splitting data* menjadi data latih dan data uji dengan rasio tertentu. Setelah data sudah siap digunakan tahapan berikutnya yaitu *modelling* sebagai proses latihan data dengan menggunakan metode SVM, agar dapat menemukan performa terbaik dari model SVM, penelitian ini juga akan melakukan proses *hyperparameter tuning* yang bertujuan untuk mencari parameter-parameter terbaik yang akan digunakan, *hyperparameter tuning* ini dilakukan dengan menggunakan metode GridSearchCV yang membuat beberapa skenario uji parameter dengan tingkat evaluasi dari *data validation* yang diambil dari data latih. Setelah mendapatkan model yang terbaik tahapan selanjutnya yaitu *evaluation* sebagai tahapan pengujian model untuk melihat berapa besar nilai performa model SVM tersebut dengan mengambil nilai akurasi model tersebut, kemudian dengan hasil nilai evaluasi performa tersebut maka model SVM-SIFT tersebut dapat dibandingkan dengan model SVM tanpa ekstraksi fitur SIFT, perbandingan model ini akan menemukan kesimpulan apakah model SVM-SIFT dapat meningkatkan performa akurasi yang baik. Tahapan terakhir penelitian ini adalah *deployment* sebagai proses peluncuran model yang telah dibangun sebelumnya ke dalam aplikasi untuk digunakan secara bebas oleh pengguna dalam *platform website* menggunakan layanan Heroku dan Github, proses pengembangan aplikasi pada tahapan ini menggunakan metode RAD. Berikut secara singkat alur metodologi penelitian ini pada Gambar 3.1.



**Gambar 3.1 Alur Metodologi Penelitian**

Adapun proses lengkap dari tahapan ini dalam bentuk flowchart dapat dilihat seperti berikut ini.



Gambar 3.2 Flowchart Proses Utama

### 3.1. Business Understanding

Beberapa hal yang dilakukan pada *business understanding* yaitu mengkaji studi literatur terhadap permasalahan dan solusi yang ada. Tahapan ini menganalisis beberapa permasalahan yang diangkat dengan membaca literatur seperti jurnal-jurnal atau prosiding maupun buku serta artikel-artikel resmi yang diakui kebenarannya. Berdasarkan hasil studi literatur bahwa pada penelitian-penelitian sebelumnya mengenai pengenalan tulisan tangan karakter hanacaraka aksara jawa terdapat permasalahan terhadap penggunaan ukuran *dataset* yang dibutuhkan untuk menggapai performa akurasi yang optimal. Beberapa penelitian lain juga telah menerapkan penggunaan *dataset* yang sedikit dalam klasifikasi hanacaraka namun tidak mendapatkan hasil akurasi yang optimal. Solusi dari hal tersebut yaitu pada penelitian ini menggunakan metode yang kuat dalam melakukan klasifikasi terhadap fitur dan kelas yang banyak yaitu *Support Vector Machine* (SVM), kemudian agar memperoleh performa akurasi yang optimal maka metode ini perlu dibantu dengan ekstraksi fitur dari *Scale Invariant Feature Transform* (SIFT) yang sangat kuat pada masalah skala gambar sehingga dapat membantu meningkatkan performa akurasi yang cukup optimal dengan dataset yang tidak banyak.

### 3.2. Data Understanding

Pada tahapan *data understanding* hal yang dilakukan adalah mengumpulkan data dan memahami data untuk ditindaklanjuti pada tahapan selanjutnya. Pada penelitian ini *dataset* karakter hanacaraka aksara jawa diambil secara *online* dari website Kaggle oleh Phiard yang berjumlah sebanyak 2632 karakter dengan 20 jenis karakter didalamnya. Namun pada *dataset* tersebut telah dilakukan augmentasi sejumlah 6 varian setiap gambarnya termasuk

varian normal. Sayangnya augmentasi yang dilakukan oleh Phiard tidak signifikan terlihat berbeda dengan gambar aslinya sehingga pada penelitian ini hanya mengambil gambar yang normal sebanyak 420 karakter dengan 20 jenis karakter tanpa augmentasi oleh Phiard.

Data gambar yang telah dikumpulkan tersebut merupakan gambar yang berwarna hitam putih dengan ukuran 224 x 224 piksel, contoh karakter dapat dilihat pada Gambar 3.2 berikut.



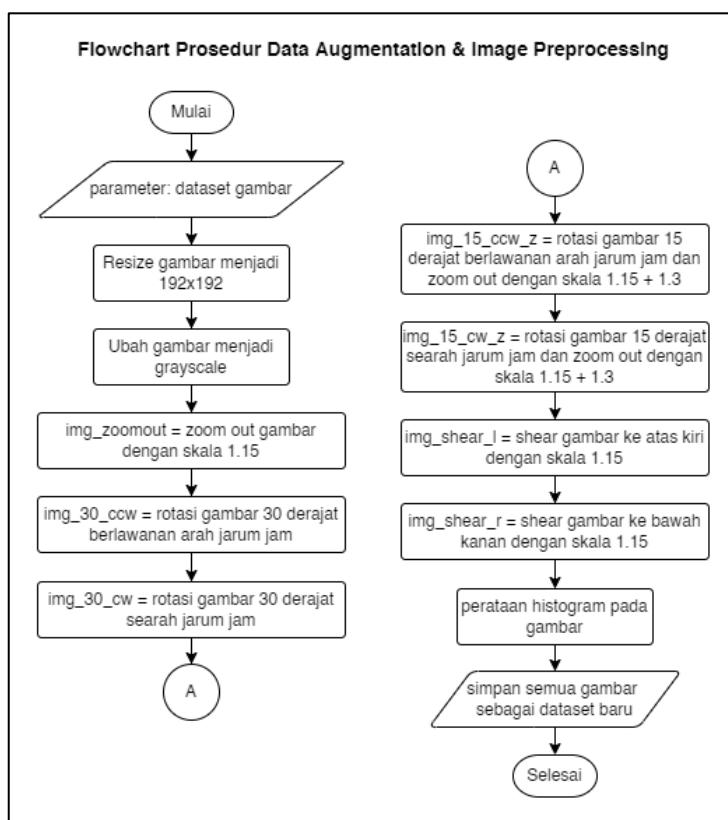
**Gambar 3.3 Contoh Karakter ‘ha’**

### 3.3. Data Preparation

Setelah mengetahui kebutuhan data yang digunakan, pada tahap *data preparation* yaitu data akan diolah terlebih dahulu sebelum masuk pada tahapan *modelling*, sehingga tahap ini akan menghasilkan keluaran sebuah fitur yang sudah rapi dan siap dilatih. Tahapan ini akan dibagi menjadi 3 tahapan penting yaitu *Data Augmentation & Image Preprocessing*, *Feature Extraction*, dan *Feature Preprocessing*.

#### 3.3.1. Data Augmentation & Image Preprocessing

Proses pada tahapan ini secara ringkas dapat dilihat pada gambar *flowchart* dibawah berikut ini.



**Gambar 3.4 Flowchart Image Augmentation & Preprocessing Process**

Penggunaan *dataset* yang sedikit memiliki tantangan tersendiri, seperti yang telah dijelaskan diatas pada penelitian ini menggunakan jumlah *dataset* yang tidak banyak. *Dataset* diambil dari sumber berjumlah 420 data gambar dengan 20 kelas. Karena *dataset* yang diperoleh sangat sedikit maka augmentasi data perlu dilakukan untuk menghasilkan *dataset* yang baru dengan jumlah yang lebih banyak dan variasi yang berbeda-beda.

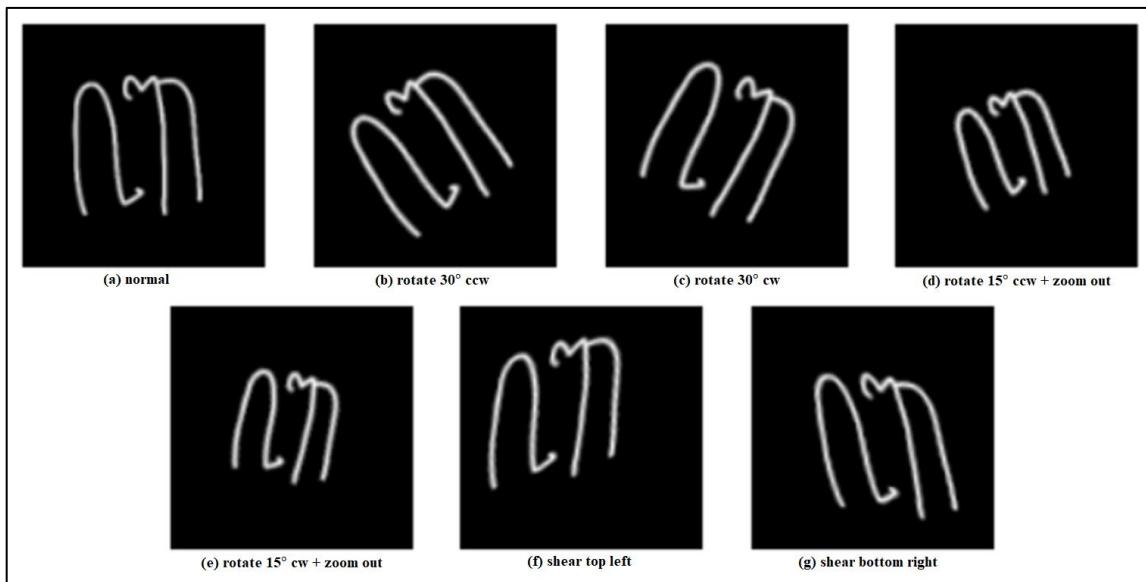
Terdapat banyak metode augmentasi yang dapat dilakukan, namun pada penelitian ini menggunakan tiga metode augmentasi gambar yaitu rotasi gambar, pengecilan gambar, dan *shear* gambar. Pada rotasi gambar dilakukan dengan menggunakan Rumus 2.1., kemudian untuk pengecilan gambar digunakan proses kombinasi antara *cropping* gambar dan *resizing* gambar, dan terakhir pada *shear* gambar dilakukan dengan menggunakan Rumus 2.2. atau Rumus 2.3.

Metode tersebut digunakan karena hasil yang diperoleh tetap menjaga kualitas gambar yang ada dan tidak membuat gambar menjadi rusak ataupun sulit dilihat sehingga variasi dataset tidak terlalu buruk dan masih layak untuk digunakan. Setelah augmentasi dilakukan maka jumlah *dataset* menjadi 2940 dengan tujuh variasi gambar hasil augmentasi yaitu:

- a. Gambar normal
- b. Rotasi 30 derajat berlawanan jarum jam
- c. Rotasi 30 derajat searah arah jarum jam
- d. Rotasi 15 derajat berlawanan jarum jam dan pengecilan gambar
- e. Rotasi 15 derajat searah arah jarum jam dan pengecilan gambar
- f. *Shear* gambar kearah atas kiri
- g. *Shear* gambar kearah bawah kanan

Selain itu data gambar perlu dilakukan *preprocessing* untuk memperbaiki gambar dan menyediakan data gambar yang terbaik. Beberapa hal yang dilakukan pada tahapan ini yaitu mengganti ukuran gambar, mengganti warna gambar menjadi hitam putih dan melakukan perataan histogram pada gambar. Proses *preprocessing* tersebut dilakukan bersamaan ketika sedang melakukan augmentasi terhadap *dataset*.

Pertimbangan terhadap ukuran suatu gambar telah dilakukan uji coba terlebih dahulu dalam mencari ukuran yang optimal untuk menyediakan *dataset* dalam proses pemodelan. Ukuran gambar yang akan dilakukan proses *resizing* gambar yaitu menjadi 192x192 piksel. Selanjutnya gambar akan diubah menjadi hitam putih dengan menggunakan metode *weighted grayscaling* seperti pada Rumus 2.4. Setelah itu proses terakhir dilakukan *equalizing histogram* yaitu dengan meratakan histogram pada sebuah matriks gambar agar membuat gambar menjadi lebih tajam, penerapan *equalizing histogram* dihitung dengan cara menggunakan Rumus 2.6. Berikut Gambar 3.4. merupakan hasil dari semua proses pengolahan gambar beserta hasil augmentasinya.



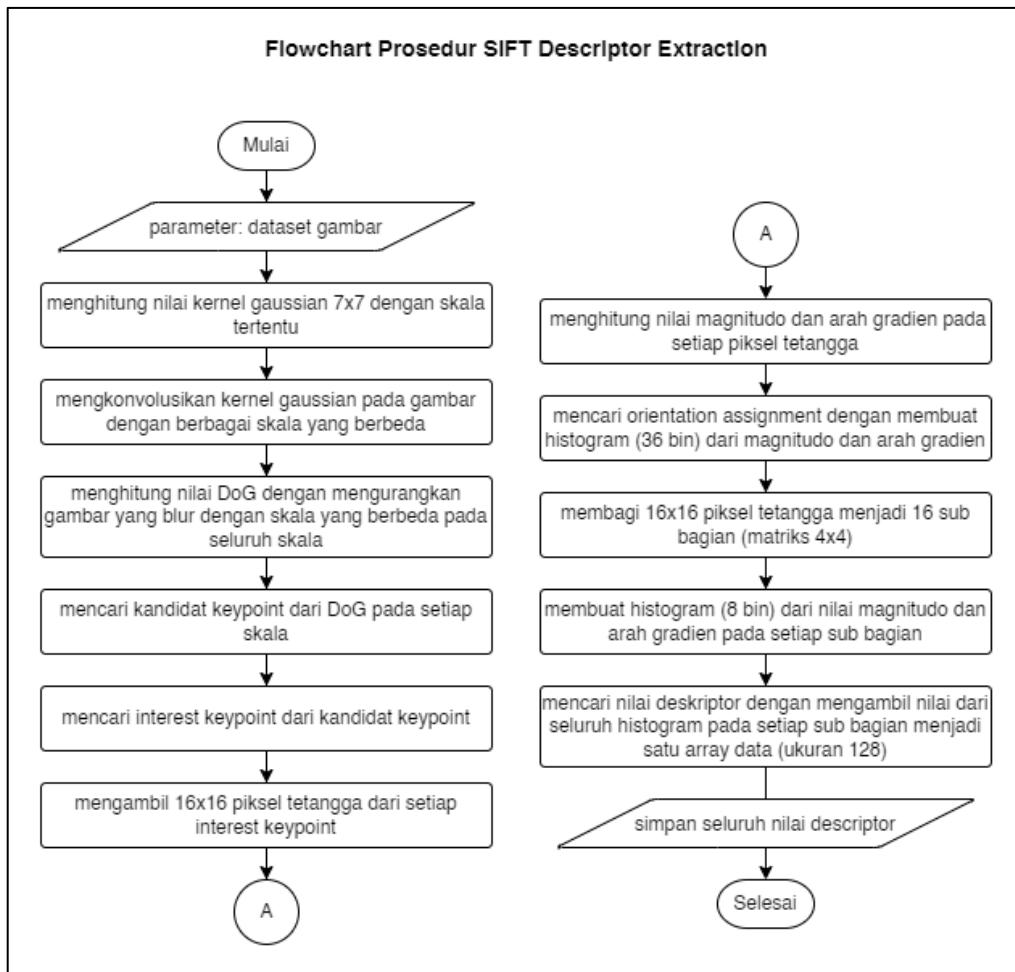
Gambar 3.5. Hasil Augmentasi dan *Preprocessing* pada Gambar Karakter ‘ba’

### 3.3.2. Feature Extraction

Dalam penelitian ini ekstraksi fitur menggunakan *Scale Invariant Feature Transform* (SIFT). SIFT mengambil nilai dari besaran sudut-sudut dari sebuah gambar menjadi sebuah fitur yang akan digunakan. Namun pada penerapannya dalam penelitian ini, hasil fitur yang diekstrak oleh SIFT pada gambar tidak dapat digunakan langsung kedalam model *machine learning*, karena input yang diharapkan oleh model berbeda dengan output yang dikeluarkan oleh SIFT, hasil ekstraksi SIFT akan berukuran dinamis sesuai dengan banyaknya fitur *keypoint* terhadap gambar sedangkan model *machine learning* membutuhkan input fitur yang statis. Sehingga dalam penerapannya hasil fitur dari SIFT akan diolah kembali hingga menjadi bentuk *Bag of Feature* (BoF) atau *Bag of Words* (BoW) dengan bantuan metode K-Means dalam klasterisasi seluruh fitur dari semua data gambar yang telah diekstrak. Kemudian fitur akhir akan dihasilkan dengan cara mencari nilai klaster menggunakan K-Means terhadap BoW yang telah dibuat dengan jumlah fitur sebanyak jumlah klaster yang ditentukan. Fitur akhir itulah yang akan menjadi input model yang dibangun. Maka dari itu proses ekstraksi fitur ini akan dibagi menjadi tiga proses utama, yaitu ekstraksi deskriptor dari SIFT, membuat BoW dari deskriptor menggunakan klustering, dan membuat fitur akhir dari BoW.

#### a. Ekstraksi Deskriptor dari SIFT

Ada empat langkah utama yang terlibat dalam algoritma ekstraksi fitur SIFT yaitu *scale-space peak selection* yang dilakukan dalam mencari lokasi potensial untuk menemukan fitur, kemudian *keypoint localization* dilakukan untuk menemukan titik kunci fitur secara akurat, selanjutnya proses *orientation assignment* dilakukan untuk menetapkan orientasi ke poin-poin utama, dan terakhir yaitu *keypoint descriptor* dilakukan untuk menggambarkan keypoint sebagai vektor berdimensi tinggi. Berikut ini adalah proses SIFT secara ringkas dalam bentuk *flowchart*.



**Gambar 3.6 Flowchart Ekstraksi SIFT**

Pertama *scale-space peak selection* dilakukan dengan menyiapkan *Gaussian Blur Filter* yang memiliki nilai sigma dinamis untuk setiap skala yang digunakan, dari *gaussian* matriks yang diperoleh akan dikonvolusikan terhadap citra agar menghasilkan citra yang blur, dalam perhitungannya sebagai berikut untuk nilai sigma yaitu  $\sigma = 1,6$  dan perbedaan pada tiap skala akan mengalikan nilai sigma dengan nilai  $k = \sqrt{2}$ , sehingga nilai sigma pada tiap skala yaitu  $\sigma_s = [k^0\sigma, k^1\sigma, k^2\sigma, \dots, k^n\sigma]$ . Sebagai contoh, pertama dilakukan yaitu membuat *gaussian filter* dengan kernel 7x7, karena *gaussian* merupakan nilai yang terpusat maka nilai yang ditengah matriks kernel adalah nilai yang paling besar sedangkan nilai yang diujung kernel adalah nilai yang paling kecil, sehingga pembuatan matriks akan seperti berikut, *nilai baris* =  $[-3, -2, -1, 0, 1, 2, 3]$  dan *nilai kolom* =  $[-3, -2, -1, 0, 1, 2, 3]$  semakin dekat nilai ke angka 0 maka nilai *gaussian* akan semakin besar. Selanjutnya terapkan Rumus 2.9.

$$G_{[0]} M_{[i,j]}(x_i, y_j, \sigma_0) = \frac{1}{2\pi\sigma_s^2} e^{-\frac{x_i^2+y_j^2}{2\sigma_s^2}}$$

$$G_{[0]} M_{[0,0]}(x_0, y_0, k^0 \sigma) = \frac{1}{2\pi(k^0 \sigma)^2} e^{-\frac{x_0^2+y_0^2}{2(k^0 \sigma)^2}}$$

$$G_{[0]} M_{[0,0]}(x_0, y_0, k^0 \sigma) = \frac{1}{2\pi(\sqrt{2}^0 \cdot 1,6)^2} e^{-\frac{-3^2+(-3)^2}{2(\sqrt{2}^0 \cdot 1,6)^2}}$$

$$G_{[0]} M_{[0,0]}(x_0, y_0, k^0 \sigma) = \frac{1}{16.08} e^{-3.515625}$$

$$G_{[0]} M_{[0,0]}(x_0, y_0, k^0 \sigma) = 0.00184826$$

Perhitungan terus dilakukan hingga menjadi matriks 7x7 seperti berikut.

**Tabel 3.1 Kernel Matriks dari Gaussian Filter**

0.0018483	0.0049077	0.0088176	0.0107194	0.0088176	0.0049077	0.0018483
0.0049077	0.0130315	0.0234134	0.0284635	0.0234134	0.0130315	0.0049077
0.0088176	0.0234134	0.0420663	0.0511396	0.0420663	0.0234134	0.0088176
0.0107194	0.0284635	0.0511396	0.0621699	0.0511396	0.0284635	0.0107194
0.0088176	0.0234134	0.0420663	0.0511396	0.0420663	0.0234134	0.0088176
0.0049077	0.0130315	0.0234134	0.0284635	0.0234134	0.0130315	0.0049077
0.0018483	0.0049077	0.0088176	0.0107194	0.0088176	0.0049077	0.0018483

Setelah mendapatkan kernel matriks selanjutnya dilakukan konvolusi terhadap citra gambar berdasarkan ukuran kernel yang dibuat. Berikut perhitungan konvolusi yang dilakukan mengacu pada Rumus 2.8., sebagai contoh mengambil beberapa matriks.

**Tabel 3.2 Contoh Ilustrasi Konvolusi Matriks**

...											
255	255	255	255	255	255	255	103	0	0	...	...
255	255	206	255	255	255	255	255	7	0	...	...
255	11	0	57	255	255	255	255	255	0	...	...
255	0	0	0	55	255	255	255	255	239	...	...
255	0	0	0	0	234	255	255	255	255	...	...
153	0	0	0	0	0	255	255	255	255	...	...
28	0	0	0	0	0	2	255	255	255	...	...
0	0	0	0	0	0	0	113	255	255	...	...
0	0	0	0	0	0	0	0	251	255	...	...
0	0	0	0	0	0	0	0	0	255	...	...

Dari tabel diatas dapat dilihat bahwa konvolusi dilakukan secara satu per satu terhadap ukuran kernel dengan matriks yang dihitung secara bergantian.

Perhitungan konvolusi:

$$\text{Nilai matriks}_{ij} = \sum \text{kernel} \times \text{matriks}_{ij}$$

$$\text{Nilai matriks}_{ij} = 0.001848 \times 255 + 0.0049077 \times 255 + \dots + 0.0018483 \times 2$$

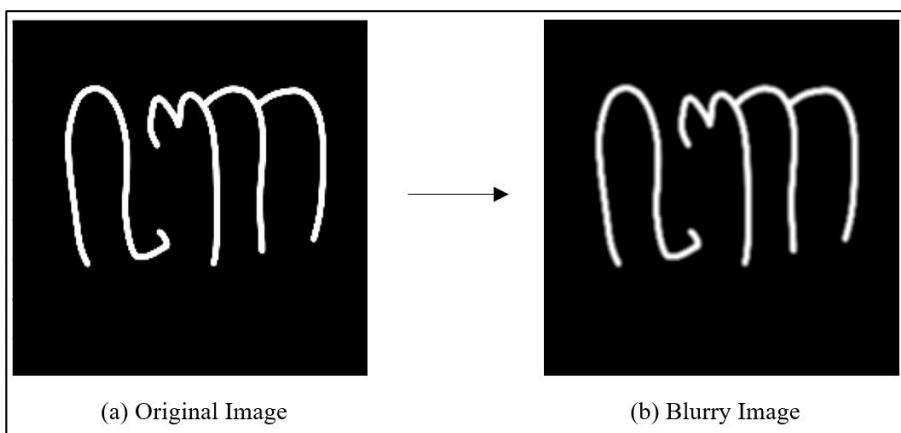
Nilai matriks<sub>ij</sub> = 38

Berikut hasil perhitungan konvolusinya

**Tabel 3.3 Hasil Matriks Setelah Dikonvolusi**

	...										
	255	250	249	250	255	242	196	111	38	1	
...	231	196	175	201	236	255	240	183	96	26	...
	202	131	82	118	183	235	255	231	170	83	
	176	81	7	38	121	204	253	255	228	153	
	163	70	0	5	59	148	228	255	253	210	...
	134	46	0	0	23	82	169	230	255	240	...
	102	18	0	0	0	25	96	184	241	255	
	83	3	0	0	0	0	38	125	213	255	
	80	0	0	0	0	0	11	68	156	230	
	79	0	0	0	0	0	0	25	93	180	
	...										

Pada Tabel 3.3. merupakan matriks hasil proses konvolusi yang dilakukan dengan kernel dari gaussian dengan ukuran 7x7. Berikut hasil gambar yang telah dikonvolusi.



**Gambar 3.7 Penerapan Gaussian Filter pada gambar**

Kemudian lakukan perhitungan yang berulang-ulang hingga mendapatkan beberapa gambar yang blur dengan tingkatan nilai skala sigma yang berbeda dalam tiap *space*. Setelah itu kurangkan matriks citra dari skala terendah dengan satu skala diatasnya sehingga akan menghasilkan nilai matriks citra dari *Difference of Gaussian* (DoG), berikut perhitungannya yang mengacu pada Rumus 2.7.

$$D(x, y, \sigma) = L(x, y, k^0\sigma) - L(x, y, k^1\sigma)$$

Dengan mengambil contoh matriks hasil konvolusi gaussian dengan skala 0 yaitu seperti matriks berikut ini.

**Tabel 3.4 Matriks Blur Skala 0**

												...
...	255	250	249	250	255	242	196	111	38	1		...
	231	196	175	201	236	255	240	183	96	26		
	202	131	82	118	183	235	255	231	170	83		
	176	81	7	38	121	204	253	255	228	153		
	163	70	0	5	59	148	228	255	253	210		
	134	46	0	0	23	82	169	230	255	240		
	102	18	0	0	0	25	96	184	241	255		
	83	3	0	0	0	0	38	125	213	255		
	80	0	0	0	0	0	11	68	156	230		
	79	0	0	0	0	0	0	25	93	180		
...												

Kemudian dari matriks pada Tabel 3.4. dilakukan perhitungan pengurangan matriks dengan matriks hasil konvolusi dengan skala 1, seperti matriks berikut.

**Tabel 3.5 Matriks Blur Skala 1**

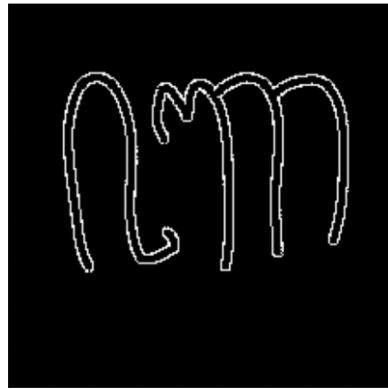
												...
...	255	250	249	250	255	240	195	111	39	1		...
	229	195	173	200	234	255	239	183	97	28		
	200	135	85	119	183	234	255	229	169	84		
	173	84	7	40	122	203	253	255	226	152		
	161	72	0	6	60	146	226	255	253	209		
	134	47	0	0	25	83	168	228	255	239		
	104	19	0	0	0	27	97	183	240	255		
	86	3	0	0	0	0	40	125	212	255		
	83	0	0	0	0	0	12	68	155	228		
	82	0	0	0	0	0	0	26	94	180		
...												

Untuk mencari nilai DoG, kedua matriks tersebut dilakukan perhitungan pengurangan sehingga akan menghasilkan matriks berikut.

**Tabel 3.6 Matriks Hasil Perhitungan DoG**

												...
...	0	0	0	0	0	2	1	0	255	0		...
	2	1	2	1	2	0	1	0	255	254		
	2	252	253	255	0	1	0	2	1	255		
	3	253	0	254	255	1	0	0	2	1		
	2	254	0	255	255	2	2	0	0	1		
	0	255	0	0	254	255	1	2	0	1		
	254	255	0	0	0	254	255	1	1	0		
	253	0	0	0	0	0	254	0	1	0		
	253	0	0	0	0	0	255	0	1	2		
	253	0	0	0	0	0	0	255	255	0		
...												

Dan berikut ini adalah hasil gambar perhitungan DoG yang dilakukan.



**Gambar 3.8 Gambar DoG**

Sama seperti sebelumnya, lakukan perhitungan DoG berulang kali pada setiap skala hingga menghasilkan banyak gambar DoG.

Setiap DoG yang dihasilkan, *keypoint* akan diidentifikasi berdasarkan lokal maxima/minima pada setiap skala. Ini dilakukan dengan cara membandingkan setiap piksel yang ada di dalam citra DoG terhadap 8 piksel tetangganya di skala yang sama dan 9 piksel tetangga lainnya pada setiap skala tetangganya. Jika nilai piksel tersebut merupakan maxima/minima dari perbandingan seluruh piksel tetangganya (26 piksel), maka nilai piksel tersebut menjadi sebagai kandidat *keypoint* pada skala tersebut.

Setelah melakukan pencarian kandidat *keypoint*, maka akan banyak sekali *keypoint* yang terdeteksi dalam sebuah citra yang diproses, dan beberapa diantaranya tidak stabil. *Keypoint* ini akan tersebar disetiap adanya extrema, sehingga tidak dipungkiri bahwa banyak *keypoint* yang terbentuk tidak akurat atau tidak tepat karena tidak adanya batasan ataupun *threshold* dalam pencarian sebelumnya. Pada tahapan selanjutnya *keypoint localization* dilakukan untuk pencocokan detail fitur ke data terdekat untuk lokasi, skala, dan rasio kelengkungan utama yang akurat. Informasi ini memungkinkan untuk menolak titik yang kontrasnya rendah (dan karenanya sensitif terhadap noise) atau tidak terlokalisasi di sepanjang tepi.

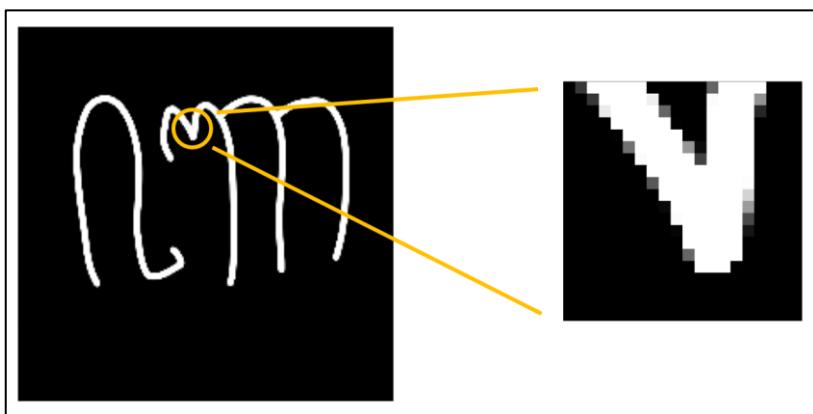
Pertama, untuk setiap titik kunci kandidat, interpolasi data terdekat digunakan untuk menentukan posisinya secara akurat. Pendekatan baru menghitung lokasi interpolasi ekstrem, yang secara substansial meningkatkan pencocokan dan stabilitas. Interpolasi ini dilakukan menggunakan *Quadratic Taylor Expansion* dari fungsi skala DoG, dengan kandidat *keypoint* sebagai titik asal. Kedua, untuk membuang *keypoint* yang memiliki kontras rendah, maka nilai *Taylor Expansion* orde kedua dihitung pada suatu *offset* tertentu. Jika nilai tersebut kurang dari 0,03 maka kandidat *keypoint* akan dibuang. Terakhir, fungsi dari DoG akan memiliki respons yang banyak di sepanjang tepi objek, sehingga muncul beberapa kandidat *keypoint* yang kurang tepat terhadap sejumlah *noise* yang kecil, sehingga perlu menghilangkan *keypoint* tersebut. Pada setiap tepi objek terdapat nilai kelengkungan utama untuk menjadi *keypoint* yang

stabil, untuk menemukan kelengkungan utama pada tepi objek dapat dilakukan dengan memecahkan nilai *eigenvalues* pada matriks *Hessian* orde kedua.



Gambar 3.9 Hasil identifikasi interest keypoints

Hal yang dilakukan berikutnya adalah menetapkan orientasi ke setiap *keypoint* untuk menjadikannya invariant rotasi. Tahapan ini disebut sebagai *Orientation Assignment*. Pada langkah ini setiap *keypoint* diberikan satu atau lebih orientasi berdasarkan arah gradien gambar lokal. Pertama, gambar yang dikonvolusi oleh Gaussian  $L(x, y, \sigma)$  pada skala sigma semuanya diambil. Perhitungan magnitudo dan arah untuk gradien dilakukan untuk setiap piksel di wilayah tetangga di sekitar *keypoint* pada citra blur dari Gaussian  $L(x, y, \sigma)$  dilakukan dengan mengacu pada Rumus 2.10. dan Rumus 2.11. Kemudian membuat sebuah orientasi histogram dengan 36 pembagian dan setiap pembagian tersebut meliputi 10 derajat orientasi. Setiap sampel pada piksel tetangga ditambahkan kedalam bagian orientasi histogram dengan bobot dari perhitungan magnitudo pada gradien dan pembobotan dari lingkaran jendela dari citra Gaussian  $L$  dengan sigma yang nilainya 1,5 kali dari skala *keypoint*-nya. Setelah melakukan ini pada semua piksel di sekitar *keypoint*, maka histogram akan memiliki puncak di beberapa titiknya. Lalu puncak tertinggi tersebut akan diambil menjadi titik orientasi dan setiap puncak di atas 80% juga dipertimbangkan untuk menghitung orientasi *keypoint* tersebut. Sebagai contoh berikut adalah salah satu potongan dari gambar yang teridentifikasi sebagai *keypoint*.



Gambar 3.10 Potongan gambar salah satu keypoint

Dan berikut adalah matriks dari *keypoint* gambar tersebut.

**Tabel 3.7 Sampel Matriks Keypoint**

255	255	255	255	255	7	0	0	0	0	0	0	255	255	255	255	0	0
57	255	255	255	255	255	0	0	0	0	0	100	255	255	255	255	0	0
0	55	255	255	255	255	239	0	0	0	0	251	255	255	255	148	0	0
0	0	234	255	255	255	255	101	0	0	0	255	255	255	255	38	0	0
0	0	0	255	255	255	255	255	2	0	0	255	255	255	255	0	0	0
0	0	0	2	255	255	255	255	255	0	0	255	255	255	255	0	0	0
0	0	0	0	113	255	255	255	255	151	12	255	255	255	255	0	0	0
0	0	0	0	0	251	255	255	255	255	66	255	255	255	255	0	0	0
0	0	0	0	0	0	255	255	255	255	255	255	255	255	255	0	0	0
0	0	0	0	0	0	87	255	255	255	255	255	255	255	253	0	0	0
0	0	0	0	0	0	0	255	255	255	255	255	255	255	212	0	0	0
0	0	0	0	0	0	0	9	255	255	255	255	255	255	155	0	0	0
0	0	0	0	0	0	0	0	252	255	255	255	255	255	73	0	0	0
0	0	0	0	0	0	0	0	5	255	255	255	255	255	21	0	0	0
0	0	0	0	0	0	0	0	0	255	255	255	255	255	0	0	0	0
0	0	0	0	0	0	0	0	0	102	255	255	255	255	0	0	0	0
0	0	0	0	0	0	0	0	0	0	255	255	255	255	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Kemudian dari potongan gambar tersebut diambil 16x16 piksel tetangga yang akan dimasukkan dalam perhitungan magnitudo dan arah gradien pada tiap piksel.

$$m(0,0) = \sqrt{(255 - 57)^2 + (55 - 255)^2}$$

$$m(0,0) = \sqrt{79204} = 281.4$$

$$\theta(0,0) = \text{atan2}(55 - 255, 255 - 57)$$

$$\theta(0,0) = \text{atan2}(-200, 198) = -45^\circ \text{ atau } 315^\circ$$

Perhitungan dilakukan berulang hingga semua piksel tetangga memiliki nilai magnitudo dan arah gradien, berikut hasil perhitungannya.

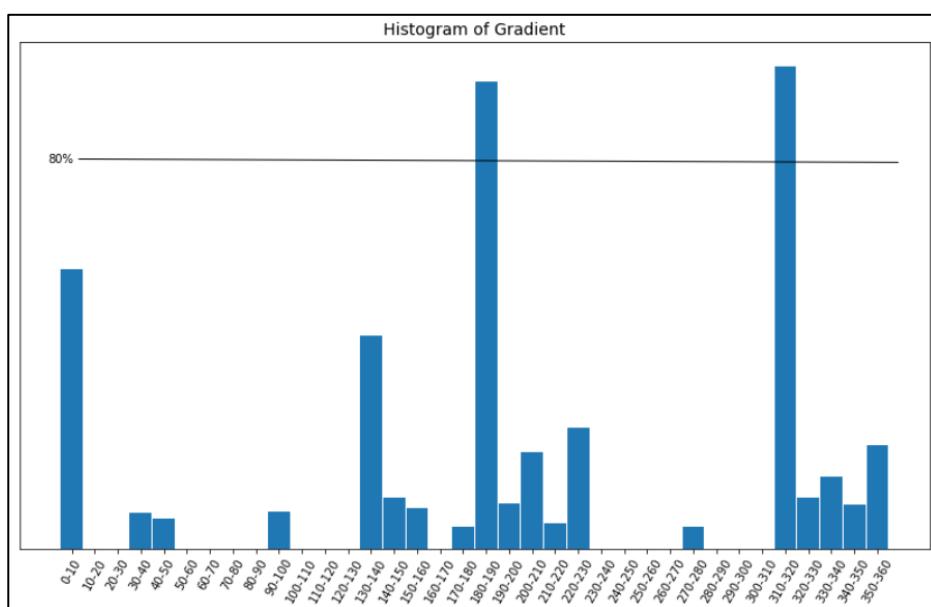
**Tabel 3.8 Matrik Hasil Perhitungan Magnitudo**

281	0	0	0	356	350	0	0	0	100	358	155	0	0	277	255	
361	201	0	0	16	361	260	0	0	251	298	4	0	107	335	148	
240	361	21	0	0	155	361	101	0	255	255	0	0	217	295	38	
0	346	359	0	0	0	296	361	2	255	255	0	0	255	258	0	
0	2	361	290	0	0	0	359	296	255	255	0	0	255	255	0	
0	0	113	361	142	0	0	104	352	123	243	0	0	255	255	0	
0	0	0	275	361	4	0	0	216	243	189	0	0	255	255	0	
0	0	0	0	358	305	0	0	0	189	0	0	0	255	255	0	
0	0	0	0	87	361	168	0	0	0	0	0	2	259	253	0	
0	0	0	0	0	269	354	0	0	0	0	0	43	273	212	0	
0	0	0	0	0	9	361	246	0	0	0	0	0	100	290	155	0
0	0	0	0	0	0	0	252	357	3	0	0	0	182	288	73	0
0	0	0	0	0	0	5	359	250	0	0	0	0	234	265	21	0
0	0	0	0	0	0	0	255	297	0	0	0	0	255	256	0	0
0	0	0	0	0	0	0	102	361	153	0	0	0	360	255	0	0
0	0	0	0	0	0	0	0	275	361	255	360	361	1	0	0	0

**Tabel 3.9 Matriks Hasil Perhitungan Arah Gradien**

315°	0°	0°	0°	136°	137°	0°	0°	0°	0°	45°	0°	0°	0°	203°	180°
315°	354°	0°	0°	180°	135°	157°	0°	0°	0°	31°	0°	0°	180°	220°	180°
347°	315°	0°	0°	0°	174°	135°	179°	0°	0°	1°	0°	0°	180°	210°	180°
0°	317°	315°	0°	0°	0°	149°	135°	180°	0°	0°	0°	0°	180°	188°	0°
0°	0°	315°	331°	0°	0°	0°	135°	149°	3°	0°	0°	0°	180°	180°	0°
0°	0°	359°	315°	358°	0°	0°	180°	134°	32°	0°	0°	0°	180°	180°	0°
0°	0°	0°	336°	315°	0°	0°	0°	151°	90°	0°	0°	0°	180°	180°	0°
0°	0°	0°	0°	315°	327°	0°	0°	0°	90°	0°	0°	0°	180°	180°	0°
0°	0°	0°	0°	0°	315°	0°	0°	0°	0°	0°	0°	0°	180°	190°	180°
0°	0°	0°	0°	0°	341°	316°	0°	0°	0°	0°	0°	0°	180°	201°	180°
0°	0°	0°	0°	0°	0°	315°	359°	0°	0°	0°	0°	0°	180°	209°	180°
0°	0°	0°	0°	0°	0°	358°	316°	0°	0°	0°	0°	0°	180°	208°	180°
0°	0°	0°	0°	0°	0°	0°	315°	0°	0°	0°	0°	0°	180°	196°	180°
0°	0°	0°	0°	0°	0°	0°	359°	329°	0°	0°	0°	0°	180°	185°	0°
0°	0°	0°	0°	0°	0°	0°	0°	315°	0°	0°	0°	0°	225°	180°	0°
0°	0°	0°	0°	0°	0°	0°	0°	338°	315°	270°	225°	225°	180°	0°	0°

Selanjutnya pembuahan gradien histogram dengan jumlah sudut  $360^{\circ}$  yang dipecah menjadi 36 bin atau batang, sehingga tiap batang berisi rentang  $10^{\circ}$ . Nilai dari batang histogram diisi dengan penambahan bobot pada tiap nilai magnitudo. Sehingga akan menghasilkan histogram seperti berikut.



**Gambar 3.11 Histogram dari Besaran Arah Gradien**

Nilai tertinggi akan menjadi arah gradien yang utama dan nilai yang diatas 80% dari nilai tertinggi juga akan menjadi kandidat arah gradien. Dari histogram tersebut bahwa nilai tertinggi terdapat pada sudut  $310^{\circ}$ . Sehingga gambar yang digunakan dalam ekstraksi fitur harus dirotasi sebesar  $310^{\circ}$  yang menjadi sumbu utama pada *keypoint* tersebut. Berikut hasil gambar yang telah dilakukan rotasi.



**Gambar 3.12 Hasil gambar keypoint setelah dilakukan rotasi**

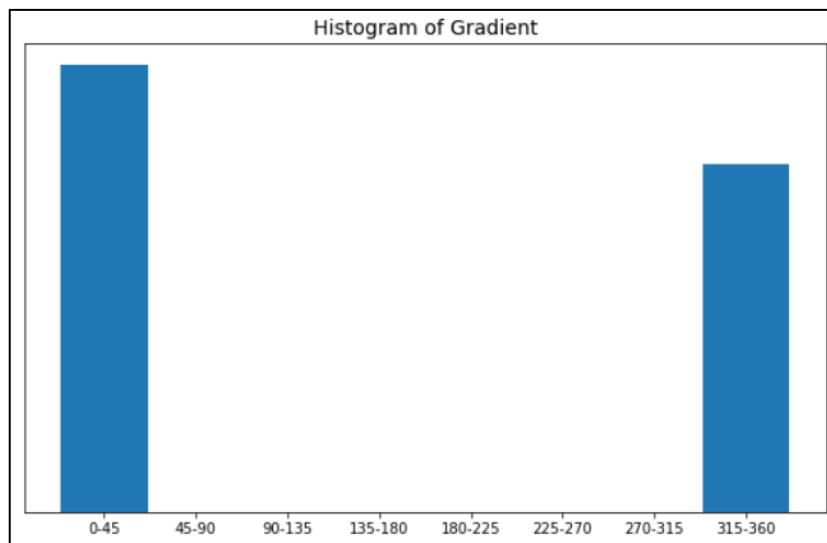
Pada tahap ini sekarang *keypoint* telah memiliki lokasi, skala, dan orientasi. Selanjutnya yang terakhir yaitu mencari nilai deskriptor atau fitur pada *keypoint* tersebut. Untuk melakukan itu piksel 16x16 tetangga *keypoint* yang diambil akan dipecahkan menjadi 16 sub blok dengan ukuran piksel 4x4. Sama seperti sebelumnya setiap piksel yang ada di blok tersebut akan dihitung nilai arah gradiennya, kemudian dari arah gradien tersebut dibuat histogram dengan 8 bin, setiap bin tersebut memiliki rentang derajat sebesar 45°.

Sama seperti sebelumnya setiap piksel akan dihitung untuk mencari nilai arah gradien menggunakan Rumus 2.11., berikut ini adalah hasil perhitungannya.

**Tabel 3.10 Matriks Hasil Perhitungan Arah Gradien Tiap Blok**

0	344	321	319	322	0	157	144	133	180	0	0	0	0	0	0	0
0	0	323	323	359	0	0	131	136	130	90	0	0	0	0	0	0
0	0	0	339	319	279	0	0	124	131	131	117	0	0	0	0	0
0	0	0	319	313	311	293	0	0	117	131	131	126	0	0	0	0
0	90	90	0	297	311	312	306	0	0	92	132	140	137	0	0	0
89	84	101	99	90	87	340	326	342	0	180	179	137	145	156	0	0
86	102	108	88	95	98	103	35	0	0	0	153	158	140	143	173	0
90	90	90	90	90	94	95	87	68	0	0	0	90	143	139	131	0
0	270	270	0	270	0	90	90	0	0	0	0	0	0	131	135	0
267	269	277	272	270	270	270	271	270	0	270	0	0	0	0	150	0
253	278	290	287	276	264	289	285	280	282	270	271	0	270	0	200	0
270	270	270	272	270	277	275	269	289	295	286	295	295	270	235	223	0
0	0	0	0	270	0	270	270	270	271	270	296	289	290	254	227	0
0	0	0	0	0	0	0	0	0	0	270	270	270	271	270	266	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	270	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Setelah itu setiap blok tersebut akan dikonversikan menjadi data histogram seperti sebelumnya dari arah gradien, berikut ini adalah contoh hasil perhitungan untuk blok pertama.



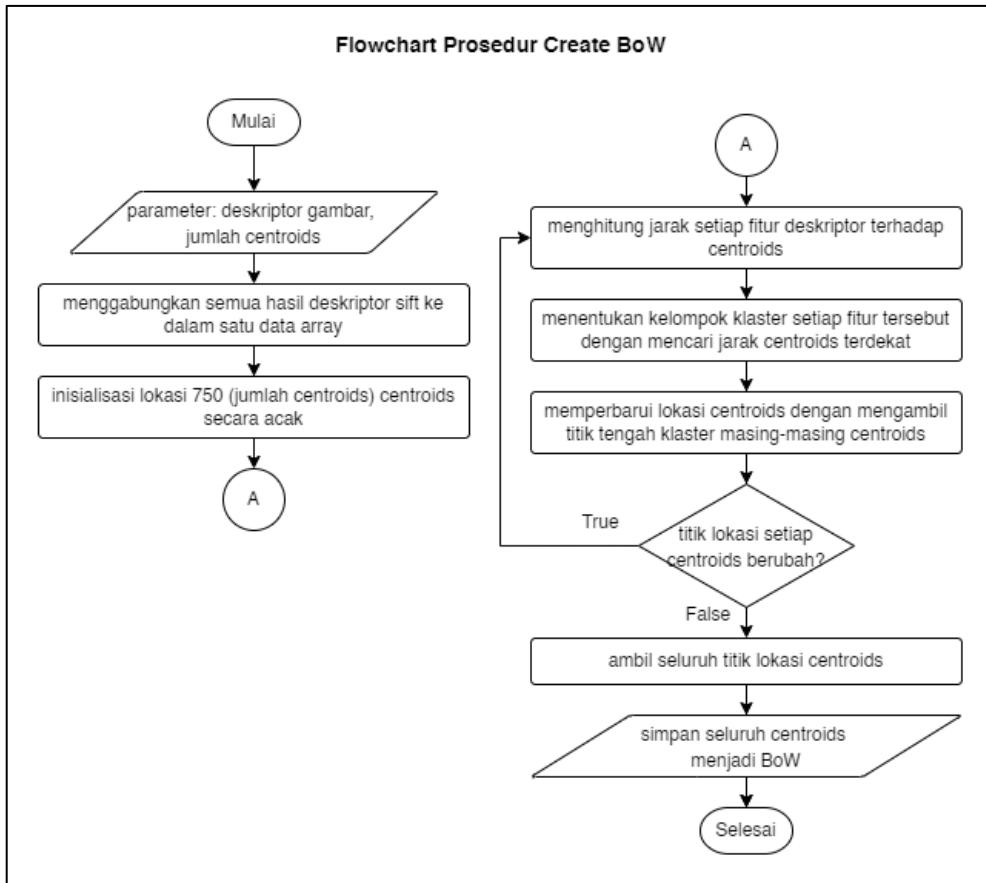
**Gambar 3.13 Histogram deskriptor blok pertama**

Hasil nilai pada sumbu y histogram tersebut akan merepresentasikan sebuah nilai deskriptor pada setiap blok dan pada sumbu x akan menjadi indeks dari nilai array deskriptor, satu blok memiliki 8 nilai karena memiliki 8 bin dari historgram sehingga total nilai deskriptor ini adalah  $8 \times 16 = 128$  fitur. Berikut salah satu contoh nilai deskriptor pada blok pertama.

$$\text{descriptor}_0 = [9, 0, 0, 0, 0, 0, 0, 7]$$

#### b. Membuat BoW dari Deskriptor Menggunakan Klastering

Proses pengambilan fitur deskriptor telah selesai dilakukan oleh SIFT, pada tahapan ini seluruh fitur deskriptor dari seluruh gambar dikumpulkan dalam satu *dataset* yang digunakan untuk pembuatan *Bag of Words* (BOW) dengan menggunakan metode *K-Means Clustering*. Hal ini dilakukan karena setiap citra memiliki jumlah fitur yang berbeda-beda sehingga untuk membuatnya sama rata maka model BoW perlu dilakukan seperti hal nya ketika ingin melakukan klasifikasi pada analisis berbasis teks/kalimat. BoW ini akan menjadi kamus dari seluruh fitur yang telah di ekstrak sebelumnya. Dataset ini berukuran  $N \times 128$  dimana  $N$  adalah jumlah gabungan fitur seluruh citra dan 128 adalah ukuran hasil deskriptor pada setiap fitur. Penggunaan metode K-Means ini dilakukan untuk menghitung kedekatan antara satu fitur dengan fitur lainnya sehingga beberapa fitur yang saling berdekatan akan menjadi satu kelompok/klaster. Pada setiap kelompok pada *K-Means* terdapat *centroids* yang merupakan titik tengah dari kelompok tersebut. Seluruh nilai titik *centroids* ini akan menjadi fitur yang disimpan dalam BoW tersebut. Berikut adalah *flowchart* proses membuat BoW secara ringkas.



**Gambar 3.14 Flowchart Create BoW**

Langkah pertama yang dilakukan pada *K-Means* adalah menetapkan besaran  $k$  atau jumlah *centroids* yang akan digunakan sebagai jumlah BoW, dalam penelitian ini besaran nilai  $k$  yaitu 750. Kemudian seluruh *centroids* tersebut akan diinisialisasikan secara acak berdampingan dengan seluruh deskriptor. Selanjutnya menghitung jarak setiap fitur deskriptor  $X$  terhadap *centroids*  $C$  untuk mencari kelompok klaster  $X_i$  dengan menggunakan Rumus 2.12. dimana  $X = [x_1, x_2, \dots, x_N]$  dan  $C = [c_1, c_2, \dots, c_k]$ . Sebagai contoh diambil 1 sampel dengan 3 *centroids*:

$$x_1 = \begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ 0 \end{bmatrix}, c_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \dots \\ 1 \end{bmatrix}, c_2 = \begin{bmatrix} 0 \\ 0 \\ 2 \\ \dots \\ 4 \end{bmatrix}, c_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 2 \end{bmatrix}$$

Jarak  $x_1$  ke  $c_1$

$$\min \left( \sum_{k=1}^k d_{11} \right) = \left\| \begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ 0 \end{bmatrix} - \begin{bmatrix} 1 \\ 2 \\ 3 \\ \dots \\ 1 \end{bmatrix} \right\|^2$$

$$\min\left(\sum_{k=1}^k d_{11}\right) = \left\| \begin{bmatrix} 2 \\ 0 \\ 1 \\ \dots \\ -1 \end{bmatrix} \right\|^2 = 398$$

Jarak  $x_1$  ke  $c_2$

$$\min\left(\sum_{k=1}^k d_{12}\right) = \left\| \begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 2 \\ \dots \\ 4 \end{bmatrix} \right\|^2$$

$$\min\left(\sum_{k=1}^k d_{12}\right) = \left\| \begin{bmatrix} 3 \\ 2 \\ 2 \\ \dots \\ -4 \end{bmatrix} \right\|^2 = 436$$

Jarak  $x_1$  ke  $c_3$

$$\min\left(\sum_{k=1}^k d_{13}\right) = \left\| \begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 2 \end{bmatrix} \right\|^2$$

$$\min\left(\sum_{k=1}^k d_{13}\right) = \left\| \begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ -2 \end{bmatrix} \right\|^2 = 580$$

Setelah melakukan perhitungan terlihat bahwa jarak  $x_1$  paling dekat dengan  $c_1$  sehingga  $x_1$  akan dikelompokkan kedalam klaster  $c_2$ , begitu juga perhitungan dilakukan pada seluruh  $x_i$  untuk mencari kelompok klasternya. Selanjutnya titik *centroids* akan diperbarui menjadi titik tengah diantara seluruh  $x_i$  yang berada di klaster  $c_k$ . Untuk melakukan hal tersebut dapat menggunakan perhitungan rata-rata pada setiap deskriptor tersebut seperti pada Rumus 2.13. Sebagai contoh terdapat 3 sampel deskriptor  $x_1, x_2, x_3$  ( $p = 3$ ) yang berada di dalam klaster  $c_1$ .

$$x_1 = \begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ 0 \end{bmatrix}, x_2 = \begin{bmatrix} 2 \\ 2 \\ 3 \\ \dots \\ 2 \end{bmatrix}, x_3 = \begin{bmatrix} 1 \\ 2 \\ 5 \\ \dots \\ 1 \end{bmatrix}, c_1 = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \dots \\ 1 \end{bmatrix}$$

$$c_1 = \frac{\begin{bmatrix} 3 \\ 2 \\ 4 \\ \dots \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \\ 3 \\ \dots \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 5 \\ \dots \\ 1 \end{bmatrix}}{3}$$

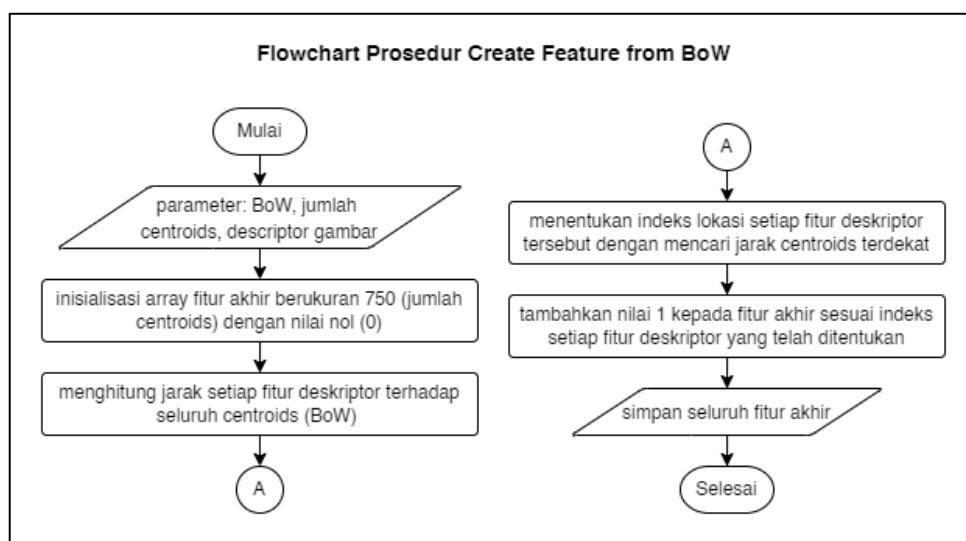
$$c_1 = \frac{\begin{bmatrix} 6 \\ 6 \\ 12 \\ \dots \\ 3 \end{bmatrix}}{3}$$

$$c_1 = \begin{bmatrix} 2 \\ 2 \\ 4 \\ \dots \\ 1 \end{bmatrix}$$

Kedua proses tersebut dilakukan berulang terus menerus hingga posisi dari seluruh *centroids* tidak ada yang berubah lagi. Terakhir seluruh posisi pada 750 titik *centroids* tersebut akan disimpan dalam kamus fitur sebagai *bag* untuk digunakan saat pembentukan fitur akhir dari BoW.

#### c. Membuat Fitur Akhir dari BoW

Setelah semua fitur deskriptor dikelompokkan dan disimpan ke dalam BoW selanjutnya dapat membentuk fitur akhir pada setiap citra. Pembentukan fitur ini dilakukan mirip seperti proses sebelumnya yaitu melakukan ekstraksi pada sebuah citra menggunakan SIFT untuk mendapatkan beberapa deskriptor unik citra tersebut, kemudian dari fitur deskriptor tersebut dilakukan klastering pada BoW untuk mendapatkan fitur akhir dari citra tersebut. Fitur dari citra tersebut sama dengan ukuran klaster yang sebelumnya dilakukan yaitu berukuran 750 fitur, isinya merupakan jumlah deskriptor pada klaster-klaster terdekat diantara 750 *centroids* (BoW). Proses ringkas dari pembuatan fitur akhir ini dapat dilihat pada flowchart dibawah.



Gambar 3.15 Flowchart Create Feature from BoW

Berikut adalah contoh pada salah satu citra, dengan mengambil sampel 1 fitur deskriptor dan 3 *centroids* secara acak. Sebelum melakukan perhitungan, persiapkan

terlebih dahulu fitur akhir dari citra tersebut dengan ukuran 750 fitur dan inisialisasikan nilainya dengan 0.

$F = [0,0,0,\dots,0]$  dengan  $\text{panjang}(F) = 750$ .

$$x_1 = \begin{bmatrix} 5 \\ 1 \\ 3 \\ \dots \\ 1 \end{bmatrix}, c_1 = \begin{bmatrix} 4 \\ 2 \\ 3 \\ \dots \\ 2 \end{bmatrix}, c_2 = \begin{bmatrix} 0 \\ 3 \\ 1 \\ \dots \\ 8 \end{bmatrix}, c_3 = \begin{bmatrix} 3 \\ 3 \\ 2 \\ \dots \\ 0 \end{bmatrix}$$

Jarak  $x_1$  ke  $c_1$

$$\min\left(\sum_{k=1}^k d_{11}\right) = \left\| \begin{bmatrix} 5 \\ 1 \\ 3 \\ \dots \\ 1 \end{bmatrix} - \begin{bmatrix} 4 \\ 2 \\ 3 \\ \dots \\ 2 \end{bmatrix} \right\|^2$$

$$\min\left(\sum_{k=1}^k d_{11}\right) = \left\| \begin{bmatrix} 1 \\ -1 \\ 0 \\ \dots \\ -1 \end{bmatrix} \right\|^2 = 244$$

Jarak  $x_1$  ke  $c_2$

$$\min\left(\sum_{k=1}^k d_{12}\right) = \left\| \begin{bmatrix} 5 \\ 1 \\ 3 \\ \dots \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 3 \\ 1 \\ \dots \\ 8 \end{bmatrix} \right\|^2$$

$$\min\left(\sum_{k=1}^k d_{12}\right) = \left\| \begin{bmatrix} 5 \\ -2 \\ 2 \\ \dots \\ -7 \end{bmatrix} \right\|^2 = 691$$

Jarak  $x_1$  ke  $c_3$

$$\min\left(\sum_{k=1}^k d_{13}\right) = \left\| \begin{bmatrix} 5 \\ 1 \\ 3 \\ \dots \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 3 \\ 2 \\ \dots \\ 0 \end{bmatrix} \right\|^2$$

$$\min\left(\sum_{k=1}^k d_{13}\right) = \left\| \begin{bmatrix} 2 \\ -2 \\ 1 \\ \dots \\ 1 \end{bmatrix} \right\|^2 = 403$$

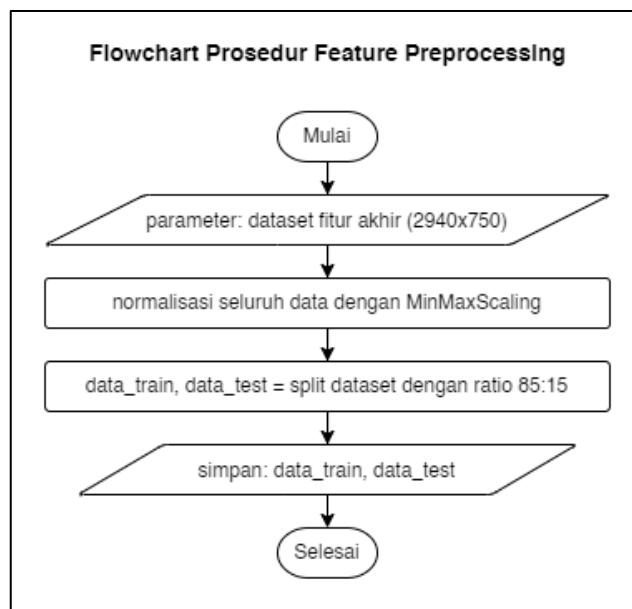
Setelah melakukan perhitungan terlihat bahwa jarak  $x_1$  paling dekat dengan  $c_1$  sehingga  $x_1$  akan dikelompokkan kedalam klaster  $c_1$ , kemudian pada fitur deskriptor

$x_1$  tersebut yaitu terjadi penambahan nilai menjadi  $F = [1,0,0,\dots,0]$ , begitu juga perhitungan dilakukan pada seluruh  $x_i$  untuk menambahkan nilai fitur  $F$ .

Terakhir apabila seluruh deskriptor citra tersebut telah dilakukan perhitungan, sebagai contoh menghasilkan fitur  $F = [5,2,4,\dots,0]$ . Selanjutnya fitur tersebut akan menjadi fitur akhir pada citra yang akan digunakan pada klasifikasi, begitu pula dilakukan pada citra yang lainnya.

### 3.3.3. Feature Preprocessing

Pada tahap ini *feature scaling* dan *split dataset* akan dilakukan untuk menyediakan data dengan distribusi yang baik, sehingga perlakuan pada *training* model dapat diterapkan dengan lebih baik. Sebelumnya fitur telah diekstrak dengan menggunakan SIFT dan menghasilkan 750 dimensi fitur pada setiap citra dengan nilai kontinu berisi jumlah BoW dengan jarak yang terdekat. Secara ringkas proses *feature preprocessing* dapat dilihat pada *flowchart* berikut ini.



Gambar 3.16 Flowchart Feature Preprocessing

#### a. Feature Scaling

Pada penelitian ini *feature scaling* menggunakan metode normalisasi *MinMaxScaling* yang mengubah nilai menjadi rentang 0-1. Perhitungan tersebut menggunakan Rumus 2.14. Sebagai contoh diambil sampel  $x = [7,3,2,6,7,2,1,0,8]$ .

$$x_{min} = 0$$

$$x_{max} = 8$$

$$x'_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

$$x'_1 = \frac{7 - 0}{8} = 0.875$$

$$x'_2 = \frac{3 - 0}{8} = 0,375$$

...

$$x'_9 = \frac{8 - 0}{8} = 1$$

Setelah semuanya dilakukan perhitungan normalisasi maka berikut hasilnya  $x' = [0.875, 0.375, 0.25, 0.75, 0.875, 0.25, 0.125, 0, 1]$ .

#### b. Split Dataset

Langkah *preprocessing* yang terakhir dilakukan adalah *split dataset* atau pembagian *dataset*. Pembagian *dataset* ini digunakan untuk mengevaluasi performa model *machine learning*. Metode evaluasi model ini membagi *dataset* menjadi dua bagian yakni bagian yang digunakan untuk data *training* dan untuk data *testing* dengan proporsi tertentu. Data *training* digunakan untuk *fit* model *machine learning*, sedangkan data *testing* digunakan untuk mengevaluasi hasil *fit* model tersebut. Pembagian *dataset* pada penelitian ini dilakukan dengan rasio perbandingan 85% untuk data *training* dan 15% untuk data *testing*. Sehingga dengan jumlah total *dataset* awal sebesar 2940 data gambar akan terpecah menjadi 2499 data gambar untuk *training* dan 441 data gambar untuk *testing*.

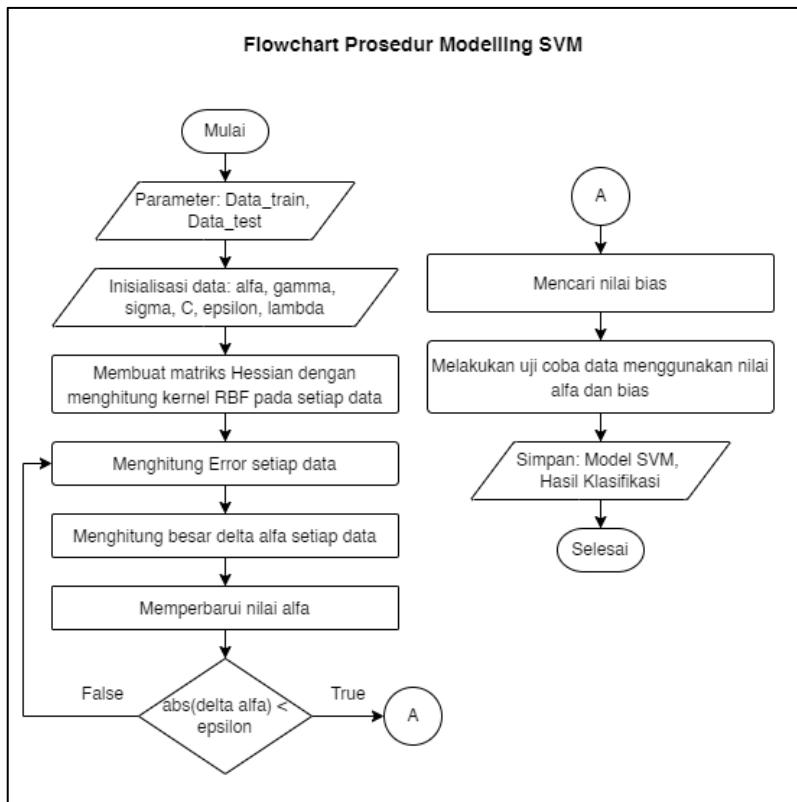
### 3.4. Modelling

Tahapan *modelling* merupakan tahapan dalam pembuatan model *machine learning* dengan cara melakukan pelatihan terhadap semua data yang telah disediakan dan melakukan optimasi model untuk mencari model dengan performa yang terbaik dengan cara menguji beberapa parameter penting dalam pelatihan model tersebut.

#### 3.4.1. Pelatihan Model SVM

Penelitian ini menggunakan SVM dalam pembuatan model *machine learning*. Proses pelatihan terhadap model SVM menggunakan dataset yang sudah disiapkan yaitu berupa data gambar dari 20 karakter hanacaraka aksara jawa. Data tersebut telah dilakukan ekstraksi fitur SIFT sehingga dengan keluaran sebesar 750 fitur pada setiap gambar. Dan hasil dari ekstraksi data tersebut juga telah dilakukan normalisasi menggunakan metode *standardization* sehingga rentang nilai setiap fitur memiliki distribusi yang seimbang.

Proses perhitungan *sequential training* SVM dilakukan untuk mengklasifikasi data dari beberapa objek. Penelitian ini menerapkan metode SVM *multiclass* dengan pendekatan *one against rest* dimana tiap kelas akan dibandingkan 1 dengan semua kelas sehingga pada akhir pelatihan yang dilakukan akan menghasilkan matriks nilai bobot alfa sebesar  $n \times K$  dan nilai bias sebesar  $1 \times K$  dengan  $n$  sebagai jumlah data dan  $K$  sebagai jumlah kelas. Sebagai contoh berikut proses perhitungan *training* pada metode SVM dengan mengambil sampel mengklasifikasikan karakter ‘ba’ sehingga data karakter selain ‘ba’ akan digabungkan dan dianggap sebagai klasifikasi negatif atau salah (kelas ‘ba’ vs selain kelas ‘ba’), kelas ‘ba’ akan bernilai 1 dan selain itu akan bernilai -1. Berikut *flowchart* ringkasan proses pemodelan menggunakan SVM.



**Gambar 3.17 Flowchart Modelling SVM**

Dengan mengambil 5 sampel ['ba', 'ba', 'nya', 'ma', 'ma'] dengan nilai label kelas menjadi [1, 1, -1, -1, -1]. Langkah pertama yang dilakukan adalah menginisialisasikan beberapa variabel parameter yang dibutuhkan dalam perhitungan,  $\lambda = 0.5$ ,  $\gamma = 0.001$ ,  $C = 1$ ,  $\epsilon = 0.0005$ ,  $\alpha = 0$ ,  $\sigma = 15.81$ . Kemudian menghitung nilai RBF kernel pada setiap data dengan menggunakan Rumus 2.16.

$$K(x_1, x_1) = \exp\left(-\frac{\|x_1, x_1\|^2}{2\sigma^2}\right)$$

$$K(x_1, x_1) = \exp\left(-\frac{0}{2(15.81)^2}\right)$$

$$K(x_1, x_1) = \exp(0)$$

$$K(x_1, x_1) = 1$$

Lakukan hal yang berulang hingga data yang terakhir, sehingga akan menghasilkan data seperti berikut.

**Tabel 3.11 Matriks Hasil Kernel RBF**

D	1	2	3	4	5
1	1	0.33070886	0.02627371	0.0303213	0.03641921
2	0.33070886	1	0.02452981	0.03978491	0.04552518
3	0.02627371	0.02452981	1	0.00481004	0.00508454
4	0.0303213	0.03978491	0.00481004	1	0.0541797
5	0.03641921	0.04552518	0.00508454	0.0541797	1

Selanjutnya melakukan perhitungan nilai matrik Hessian dari seluruh data latih dengan menggunakan Rumus 2.17.

$$D_{11} = y_1 y_1 (K(x_1, x_1) + \lambda^2)$$

$$D_{11} = (1)(1)(1 + 0.5^2)$$

$$D_{11} = 1.25$$

Lakukan perhitungan berulang hingga data yang terakhir, sehingga akan menghasilkan data seperti berikut.

**Tabel 3.12 Hasil Perhitungan Matriks Hessian**

D	1	2	3	4	5
1	1.25	0.58070886	-0.27627371	-0.2803213	-0.28641921
2	0.58070886	1.25	-0.27452981	-0.28978491	-0.29552518
3	-0.27627371	-0.27452981	1.25	0.25481004	0.25508454
4	-0.2803213	-0.28978491	0.25481004	1.25	0.3041797
5	-0.28641921	-0.29552518	0.25508454	0.3041797	1.25

Langkah selanjutnya yaitu melatih parameter bobot atau alfa dengan cara menghitung nilai *error* terlebih dahulu menggunakan Rumus 2.18. seperti berikut.

$$E_1 = \sum_{j=1}^n \alpha_1 D_{1j}$$

$$E_1 = (0 \times 1.25) + (0 \times 0.58070886) + (0 \times -0.27627371) + (0 \times -0.2803213) \\ + (0 \times -0.28641921)$$

$$E_1 = 0$$

Lakukan perhitungan berulang hingga data yang terakhir, sehingga akan menghasilkan data seperti berikut.

**Tabel 3.13 Hasil Perhitungan Error**

D	E
1	0
2	0
3	0
4	0
5	0

Kemudian menghitung delta alfa untuk memperbarui nilai alfa yang baru sebagai parameter bobot setiap data, perhitungan dilakukan menggunakan Rumus 2.19. sebagai berikut.

$$\delta\alpha_1 = \min(\max[\gamma(1 - E_1), \alpha_1], C - \alpha_1)$$

$$\delta\alpha_1 = \min(\max[0.001(1 - 0), 0], 1 - 0)$$

$$\delta\alpha_1 = \min(\max[0.001, 0], 1)$$

$$\delta\alpha_1 = \min(0.001, 1)$$

$$\delta\alpha_1 = 0.001$$

Terapkan perhitungan tersebut secara berulang dalam seluruh data, maka berikut hasil semua delta alfa.

**Tabel 3.14 Hasil Perhitungan Delta Alfa**

D	$\delta\alpha$
1	0.001
2	0.001
3	0.001
4	0.001
5	0.001

Setelah mendapatkan nilai delta alfa pada setiap data, maka dapat dilakukan pembaruan nilai pada alfa dengan menggunakan Rumus 2.20 berikut.

$$\alpha_1 = \alpha_1 + \delta\alpha_1$$

$$\alpha_1 = 0 + 0.001$$

$$\alpha_1 = 0.001$$

**Tabel 3.15 Hasil Perhitungan Pembaruan Alfa**

D	$\alpha$
1	0.001
2	0.001
3	0.001
4	0.001
5	0.001

Terakhir melakukan perhitungan tersebut secara iterasi untuk setiap  $i = 1, 2, 3, \dots, n$ . Iterasi ini akan terus dilakukan hingga nilai  $|\delta\alpha| < \varepsilon$ . Tahapan iterasi ini diawali dengan menghitung nilai *error* hingga pembaruan nilai alfa. Berikut hasil perhitungan pada iterasi kedua.

**Tabel 3.16 Hasil Perhitungan Pembaruan Alfa Iterasi Kedua**

D	$\alpha$
1	0.002
2	0.002
3	0.002
4	0.002
5	0.002

Setelah banyak iterasi yang dilakukan sehingga akan mendapatkan nilai akhir alfa yaitu sebagai berikut.

**Tabel 3.17 Hasil Akhir Perhitungan Pembaruan Alfa**

D	$\alpha$
1	1
2	1
3	1
4	1
5	1

Setelah mendapatkan nilai akhir alfa, selanjutnya dilakukan perhitungan nilai bias dengan menggunakan Rumus 2.21. nilai bias tersebut merupakan posisi bidang relatif terhadap koordinat pusat. Nilai  $K(x_i, x^+)$  merupakan nilai *support vector* kelas yang paling besar, sedangkan nilai  $K(x_i, x^-)$  merupakan nilai *support vector* kelas yang paling kecil. Nilai  $K(x_i, x^+)$  akan diambil dari nilai data pertama dan nilai  $K(x_i, x^-)$  akan diambil pada nilai data ketiga. Maka perhitungannya akan seperti berikut.

$$K(x_i, x^+) = \sum_{i=1}^m \alpha_i y_i K(x_i, x^+)$$

$$\begin{aligned} K(x_i, x^+) &= (1 \times 1 \times 1.25) + (1 \times 1 \times 0.58070886) + (1 \times -1 \times -0.27627371) \\ &\quad + (1 \times -1 \times -0.2803213) + (1 \times -1 \times -0.28641921) \end{aligned}$$

$$K(x_i, x^+) = 2.67372308$$

$$K(x_i, x^-) = \sum_{i=1}^m \alpha_i y_i K(x_i, x^-)$$

$$\begin{aligned} K(x_i, x^-) &= (1 \times 1 \times -0.27627371) + (1 \times 1 \times -0.27452981) + (1 \times -1 \times 1.25) \\ &\quad + (1 \times -1 \times 0.25481004) + (1 \times -1 \times 0.25508454) \end{aligned}$$

$$K(x_i, x^-) = -2.37909594$$

Selanjutnya menghitung bias.

$$b = -\frac{1}{2} \left[ \sum_{i=1}^m \alpha_i y_i K(x_i, x^+) + \sum_{i=1}^m \alpha_i y_i K(x_i, x^-) \right]$$

$$b = -\frac{1}{2} [2.67372308 + (-2.37909594)]$$

$$b = -\frac{0.29462714}{2}$$

$$b = -0.14731357$$

Setelah semua pelatihan pada data selesai dilakukan, maka selanjutnya akan dilakukan *testing* data dengan menghitung fungsi  $f(x)$ . Langkah yang dilakukan pertama yaitu menghitung nilai kernel pada data *testing* ke seluruh data pelatihan dengan menggunakan rumus RBF seperti sebelumnya. Berikut sebagai contoh perhitungan dengan mengambil sampel data *testing* dengan kelas ‘ba’/positif.

$$K(x_{testing}, x_1) = \exp\left(-\frac{\|x_{testing}, x_1\|^2}{2\sigma^2}\right)$$

$$K(x_{testing}, x_1) = \exp\left(-\frac{785.396}{2(15.81)^2}\right) = 0.20782308$$

Lakukan perhitungan tersebut secara berulang kepada seluruh nilai  $x$ , berikut hasil perhitungannya.

**Tabel 3.18 Hasil Perhitungan Kernel Data Testing**

$K(x_{testing}, x_1)$	$K(x_{testing}, x_2)$	$K(x_{testing}, x_3)$	$K(x_{testing}, x_4)$	$K(x_{testing}, x_5)$
0.20782308	0.23541007	0.02091285	0.03079944	0.03741394

Setelah mendapatkan nilai masing-masing kernel kemudian dilakukan klasifikasi data dengan menggunakan Rumus 2.22 sebagai berikut.

$$f(x_{testing}) = \sum_{i=1}^m \alpha_i y_i K(x_{testing}, x_i) + b$$

$$\begin{aligned} f(x_{testing}) &= (1 \times 1 \times 0.20782308) + (1 \times 1 \times 0.23541007) \\ &\quad + (1 \times -1 \times 0.02091285) + (1 \times -1 \times 0.03079944) \\ &\quad + (1 \times -1 \times 0.03741394) + (-0.14731357) \end{aligned}$$

$$f(x_{testing}) = 0.20679335$$

Setelah mendapatkan nilai  $f(x)$  pada data  $testing$ , klasifikasi menghasilkan dua kelas yaitu kelas ‘ba’ dan kelas selain ‘ba’ dengan ketentuan yaitu untuk kelas ‘ba’ bernilai 1 dan kelas selain ‘ba’ bernilai -1. Nilai  $> 0$  menjelaskan bahwa data tersebut masuk kedalam kategori kelas 1 yaitu kelas ‘ba’, sedangkan nilai  $< 0$  menjelaskan bahwa data tersebut termasuk dalam kategori kelas -1 yaitu kelas selain ‘ba’. Sehingga dengan nilai yang telah dihitung yaitu  $f(x_{testing}) = 0.20679335$  terprediksi sebagai kelas ‘ba’ dengan nilai aktual yang sama yaitu pada kelas ‘ba’.

Dengan metode *SVM Multiclass one vs all*, perhitungan serupa dilakukan juga terhadap kelas-kelas yang lainnya, seperti klasifikasi kelas ‘ma’ dilatih dengan kelas selain ‘ma’ dan seterusnya. Sehingga akhir dari pelatihan SVM ini akan menghasilkan nilai alfa sebanyak  $750 \times K$  dan nilai bias sebanyak  $1 \times K$ .

### 3.4.2. Optimasi Model

Optimasi model dilakukan dengan menggunakan metode *Grid Search CV*. *Grid Search CV* merupakan metode dalam *parameter tuning* dengan menggunakan pendekatan terhadap konsep *k-fold cross validation* terhadap pelatihan model. Parameter SVM yang akan dioptimasi pada penelitian ini adalah nilai  $C$  sebagai nilai parameter regularization dan  $\gamma$  sebagai nilai koefisien dari persamaan gaussian. Kedua parameter tersebut sangat berpengaruh dalam menentukan performa pelatihan SVM kernel gaussian RBF.

Selain kedua parameter pelatihan model SVM tersebut, terdapat parameter lain yang dapat mempengaruhi hasil performa model setelah dilatih yaitu pada ukuran data gambar yang digunakan dan jumlah *centroids* pada K-Means ketika melakukan ekstraksi fitur SIFT

pada tahap *data preparation* sebelum *modelling*. Jumlah centroids pada K-Means ini merupakan ukuran fitur yang akan dihasilkan setelah SIFT dilakukan.

Berikut merupakan skenario kombinasi parameter yang akan diuji dalam melakukan optimasi model.

**Tabel 3.19 Parameter Optimasi Model**

No	Size	K	C	G
1	128x128	180	[1,3,6,10,15]	[0.24574, 0.25086, 0.25598, 0.2611, 0.26622]
2		250		[0.20504, 0.20931, 0.21358, 0.21785, 0.22212]
3		500		[0.13567, 0.1385, 0.14133, 0.14416, 0.14699]
4		750		[0.10508, 0.10727, 0.10946, 0.11165, 0.11384]
5		1000		[0.09142, 0.09332, 0.09522, 0.09712, 0.09902]
6	160x160	180	[1,3,6,10,15]	[0.25495, 0.26026, 0.26557, 0.27088, 0.27619]
7		250		[0.20114, 0.20533, 0.20952, 0.21371, 0.2179]
8		500		[0.13022, 0.13293, 0.13564, 0.13835, 0.14106]
9		750		[0.10009, 0.10217, 0.10425, 0.10633, 0.10841]
10		1000		[0.08725, 0.08907, 0.09089, 0.09271, 0.09453]
11	192x192	180	[1,3,6,10,15]	[0.26274, 0.26821, 0.27368, 0.27915, 0.28462]
12		250		[0.20183, 0.20603, 0.21023, 0.21443, 0.21863]
13		500		[0.12668, 0.12932, 0.13196, 0.1346, 0.13724]
14		750		[0.10235, 0.10448, 0.10661, 0.10874, 0.11087]
15		1000		[0.08487, 0.08664, 0.08841, 0.09018, 0.09195]
16	224x224	180	[1,3,6,10,15]	[0.23885, 0.24383, 0.24881, 0.25379, 0.25877]
17		250		[0.19131, 0.1953, 0.19929, 0.20328, 0.20727]
18		500		[0.11421, 0.11659, 0.11897, 0.12135, 0.12373]
19		750		[0.09026, 0.09214, 0.09402, 0.0959, 0.09778]
20		1000		[0.07609, 0.07767, 0.07925, 0.08083, 0.08241]

### 3.5. Evaluation

Tahapan evaluasi pada penelitian dilakukan dengan pengujian terhadap model yang telah dibangun dengan mencari nilai performa akurasi pada setiap skenario yang telah ditentukan. Setelah melakukan optimasi model dan mendapatkan model yang terbaik, selanjutnya menghitung ulang akurasi model tersebut serta menghitung nilai *F1-score*-nya sebagai nilai *confidence*. Berikut contoh perhitungan akurasi pada data *testing* menggunakan Rumus 2.26.

$$\text{accuracy} = \frac{\text{total\_true\_predict}}{\text{total\_data}}$$

$$\text{accuracy} = \frac{403}{441}$$

$$\text{accuracy} = 0.9138 = 91.38\%$$

Adapun hasil performa tersebut akan dibandingkan dengan model tanpa menggunakan ekstraksi fitur SIFT untuk melihat besaran pengaruh metode SIFT pada hasil klasifikasi.

Hasil model terbaik yang didapatkan akan dilakukan peluncuran model ke dalam aplikasi yang dibangun di tahapan selanjutnya.

### **3.6. Deployment**

Tahapan terakhir pada penilitian ini yaitu *deployment* yang merupakan tahapan pengembangan perangkat lunak dari model yang telah dibangun agar dapat diimplementasikan langsung kepada pengguna dalam melakukan prediksi karakter hanacaraka aksara jawa. Pada penelitian ini metode RAD diterapkan dalam pengembangan aplikasi berbasis web, proses yang dilakukan pada metode RAD diantaranya yaitu *analysis and quick design, prototype cycles (build, demonstrate, refine), testing, implementation*.

#### **3.6.1. Analysis and Quick Design**

Tahapan pertama yang dilakukan sebelum mengembangkan perangkat lunak yaitu *Analysis* dimana tahapan ini melakukan analisis kebutuhan perangkat lunak yang akan dibangun. Kebutuhan ini mencakup proses ataupun hal perencanaan yang akan dibangun dalam aplikasi, pada penelitian ini analisis kebutuhan perangkat lunak adalah sebagai berikut.

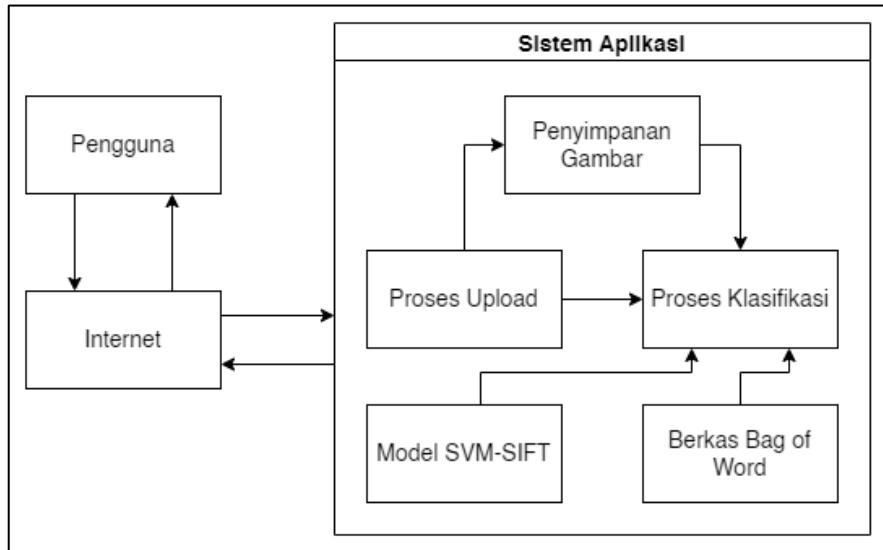
- a. Pengguna dapat melakukan input gambar ke dalam perangkat lunak
- b. Sistem dapat menerima dan menyimpan gambar yang diinput oleh pengguna
- c. Sistem dapat melakukan proses *preprocessing* data gambar yang diterima
- d. Sistem dapat melakukan prediksi klasifikasi dari gambar hasil *preprocessing*
- e. Sistem dapat menampilkan hasil klasifikasi kepada pengguna

Setelah menentukan kebutuhan perangkat lunak, selanjutnya yaitu tahapan perancangan desain arsitektur, desain proses sistem dan desain antarmuka. Tahapan ini dilakukan bertujuan untuk mendapatkan gambaran umum sistem yang akan dikembangkan. Berikut desain perangkat lunak pada penelitian ini.

- a. Perancangan Arsitektur

Arsitektur sistem yang dibuat pada penelitian ini terdiri dari pengguna aplikasi, internet, sistem aplikasi yang didalamnya terdapat beberapa komponen diantaranya proses upload, penyimpanan gambar, proses klasifikasi, berkas *bag of word*, dan model SVM-SIFT. Pengguna disini merupakan orang yang akan menggunakan sistem klasifikasi gambar karakter hanacaraka aksara jawa. Sebelum masuk dalam aplikasi pengguna membutuhkan sebuah koneksi internet selanjutnya dapat mengakses aplikasi melalui *link website*. Dalam aplikasi tersebut pengguna dapat melakukan klasifikasi dengan menginput gambar, kemudian sistem aplikasi akan menyimpan gambar tersebut dalam direktori “static/template” yang sudah disediakan pada *platform Github*. Selanjutnya sistem akan mengambil dan mengolah gambar melalui tahapan *preprocessing* yang didalamnya terdapat proses ekstraksi fitur SIFT serta BoW. Lalu proses diteruskan ke dalam klasifikasi gambar menggunakan model SVM yang telah dilakukan pelatihan sebelumnya. Terakhir hasil klasifikasi akan dikirim dan

ditampilkan kepada pengguna melalui internet. Berikut gambaran perancangan arsitektur yang dibangun.

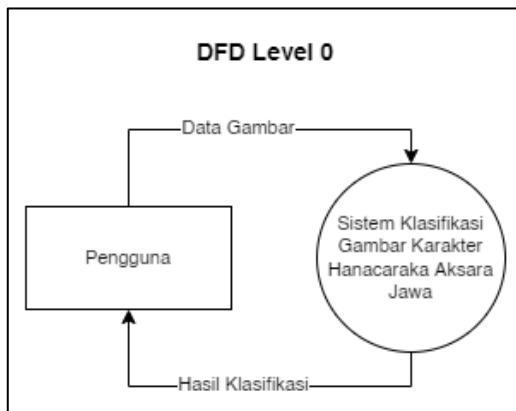


**Gambar 3.18 Perancangan Arsitektur Sistem**

#### b. Perancangan Proses

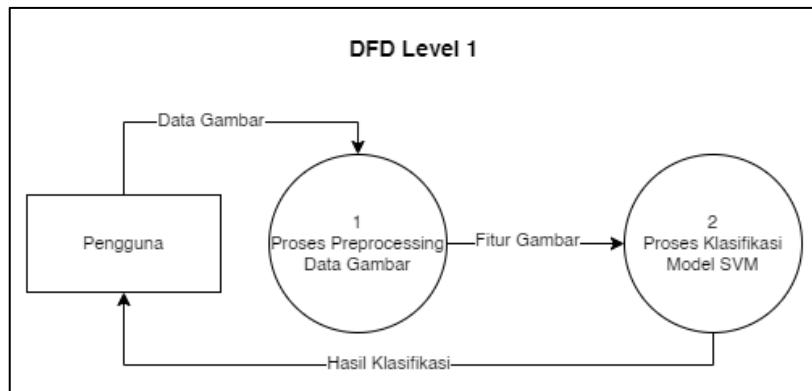
Perancangan proses sistem pada penelitian ini dilakukan dengan merancang aliran data menggunakan *data flow diagram* baik *level 0* dan *level 1*. DFD ini sebuah cara untuk mewakili aliran data melalui proses atau sistem.

Berikut hasil perancangan DFD level 0 dimana hanya terdapat entitas pengguna dan sistem yang saling berinteraksi. Pengguna akan mengirimkan data gambar ke dalam sistem kemudian sistem akan mengirimkan hasil klasifikasi kepada pengguna.



**Gambar 3.19 Proses DFD Level 0**

Kemudian berikut adalah hasil perancangan DFD level 1. Pada DFD level 1 terdapat dua cabang proses utama yaitu proses *preprocessing* data gambar yang termasuk juga ekstraksi fitur SIFT dan proses klasifikasi menggunakan model SVM. Proses pertama akan menghasilkan data fitur gambar yang akan digunakan dalam klasifikasi pada proses kedua.

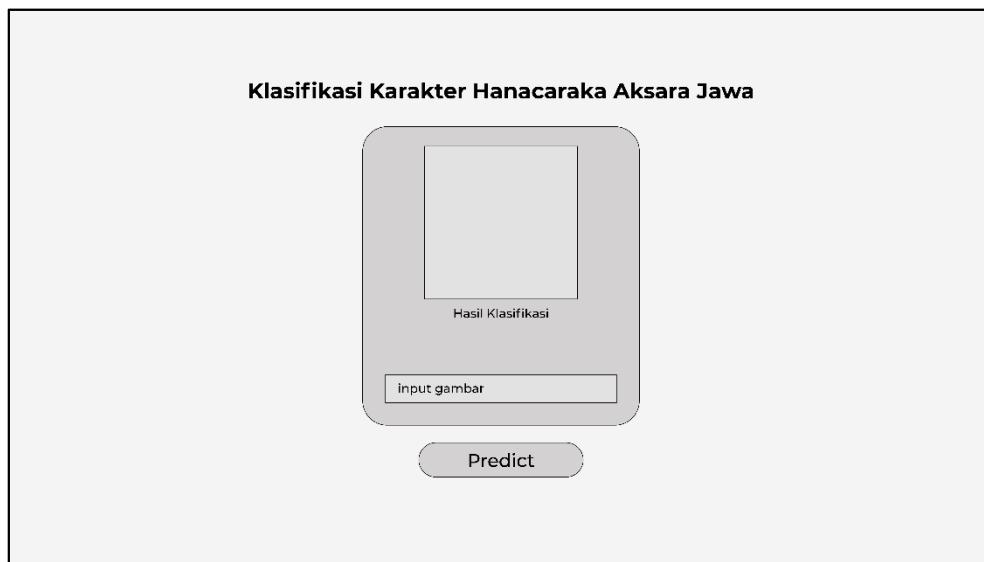


**Gambar 3.20 Proses DFD Level 1**

c. Perancangan Antarmuka

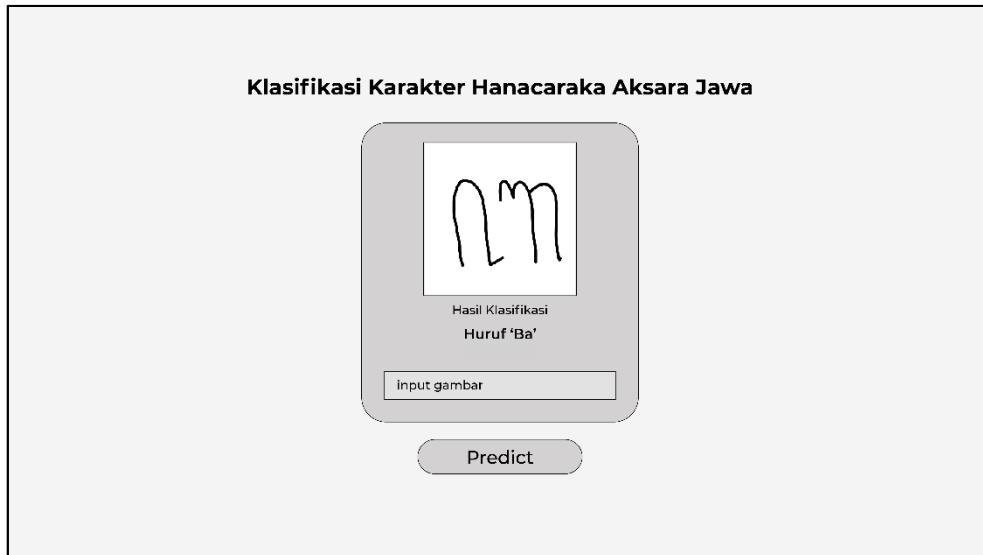
Perancangan antarmuka dilakukan dengan merancang desain kerangka tampilan halaman dari aplikasi yang akan dibuat. Kerangka tampilan tersebut akan menjadi acuan dalam pembuatan UI saat proses *development*.

Pada penelitian ini menggunakan satu halaman dengan menawarkan input gambar dari pengguna kemudian pengguna menekan tombol *predict* untuk memulai proses prediksi klasifikasi dari data gambar yang diinput. Berikut tampilan awal sebelum melakukan klasifikasi.



**Gambar 3.21 Kerangka Antarmuka Sebelum Proses Klasifikasi**

Selanjutnya sistem akan melakukan proses *preprocessing* dari data yang diinput oleh pengguna dan melakukan proses klasifikasi dari fitur hasil *preprocessing*. Sehingga tampilan halaman akan menampilkan gambar yang diinput dan hasil proses klasifikasi karakter hanacaraka aksara jawa beserta nilai akurasinya.



Gambar 3.22 Kerangka Antarmuka Setelah Proses Klasifikasi

### 3.6.2. Prototype Cycles

Tahap *prototype cycles* berisi dengan proses *development* diantaranya yaitu *build* sebagai implementasi kode program, *demonstrate* dilakukan dalam pengujian singkat terhadap fitur yang dibangun, dan *refine* sebagai perbaikan-perbaikan kecil terhadap fitur agar dapat disempurnakan. Tahapan ini berbentuk sebuah *cycles* atau perputaran dimana proses diimplementasikan dengan membangun fitur-fitur kecil dalam aplikasi kemudian digabungkan menjadi satu aplikasi besar. Pada penelitian ini tahapan tersebut dilakukan dengan mengimplementasikan seluruh rancangan desain untuk membuat sistem ke dalam program. Bahasa yang digunakan dalam membuat program tersebut adalah bahasa pemrograman python, serta menggunakan *framework flask* yang merupakan sebuah *framework* berbasis web. Sedangkan pada pembuatan program antarmuka pengguna, kode program ditulis dengan menggunakan HTML dan CSS sederhana.

### 3.6.3. Testing

Pada penelitian ini tahapan *testing* menerapkan pengujian sistem dengan metode *black box testing* untuk menguji fungsional sistem dalam menjalankan proses klasifikasi. Skenario dalam pengujian fungsional sistem yaitu:

- a. Pengguna dapat melakukan input data gambar ke dalam sistem sebagai data yang akan diklasifikasikan.
- b. Pengguna dapat menekan tombol *predict* untuk memulai proses klasifikasi.
- c. Sistem dapat menerima input data gambar dari pengguna dan melakukan proses klasifikasi dengan lancar.
- d. Sistem dapat menampilkan hasil klasifikasi kepada pengguna yaitu menampilkan data gambar yang diinput dan data kelas.

### **3.6.4. Implementation**

Setelah semua fungsi berjalan dengan lancar, selanjutnya masuk pada tahapan *implementation*. Tahapan ini merupakan proses peluncuran aplikasi ke ruang publik sehingga aplikasi tersebut sudah tersedia untuk digunakan secara global oleh pengguna bebas. Tahapan ini dilakukan dengan cara menyimpan atau mengunggah program ke dalam *platform* Github yang diintegrasikan dengan *platform* Heroku sebagai *web service* yang *open-source*. Dengan cara tersebut aplikasi dapat diakses dengan *link web* aplikasi yang diberikan oleh *platform* Heroku.

Setelah aplikasi berhasil diluncurkan, tahapan selanjutnya yaitu proses *monitoring* ataupun *maintenance* terhadap aplikasi yang sudah tersedia secara global. Tahapan ini bertujuan untuk menjaga stabilitas dari jalannya aplikasi agar dapat secara langsung mengatasi kekurangan ataupun *bug/error* yang tidak terduga pada aplikasi. Tahapan *review* juga dapat mengambil *feedback* dari pengguna agar aplikasi dapat ditingkatkan seiring berjalannya waktu dan kebutuhan pengguna.

## BAB IV

### HASIL DAN PEMBAHASAN

Pada bagian hasil dan pembahasan membahas mengenai tahapan dari implementasi, hasil, dan pembahasan terkait perancangan yang telah dilakukan pada bagian bab 3 yang dimulai dari tahap *business understanding* (hasil tahapan ini telah dijabarkan pada bab 1), *data understanding*, *data preparation*, *modelling*, *evaluation*, dan *deployment process*. Tahapan ini akan membuktikan permasalahan yang diangkat akan diselesaikan dengan solusi yang diberikan pada penelitian ini, hasil penelitian akan membahas bagaimana proses perancangan/pembangunan dan analisis performa akurasi terhadap model dari metode SVM-SIFT yang akan dibandingkan dengan model dari metode SVM tanpa bantuan SIFT.

#### 4.1. Implementasi

Pada bagian implementasi ini menjelaskan secara lengkap bagaimana proses rancangan pada bab 3, kemudian direalisasikan menjadi aplikasi dan hasil rancangan tersebut. Proses perancangan diimplementasikan dalam kode-kode pemrograman untuk membangun sistem. Berikut merupakan tampilan dari aplikasi pengenalan karakter hanacaraka aksara jawa yang diilustrasikan pada Gambar 4.1.



Gambar 4.1. Halaman Aplikasi Hasil Prediksi

Aplikasi pada Gambar 4.1 di atas, terdapat hasil halaman prediksi gambar dimana terdapat bagian gambar yang merupakan tampilan gambar yang dimasukkan oleh pengguna, kemudian terdapat kolom input data gambar dan tombol “Prediksi Gambar” untuk melakukan proses pengenalan atau klasifikasi dari gambar yang telah dimasukkan, setelah itu terdapat hasil klasifikasi yang dilakukan dalam bentuk teks sebagai contoh “Huruf ‘nya’”.

Pada implementasinya, kode dibuat dan ditulis dengan beberapa dukungan *library* yang digunakan dalam membantu pembuatan aplikasi dan perancangan model. Seluruh *library* tersebut bersifat *open-source* sehingga kode-kode tersebut tidak membutuhkan lisensi maupun perizinan dalam penggunaannya. Berikut adalah *library* yang digunakan dalam proses implementasi.

---

**Algoritma 1: Library yang digunakan**

---

```
# load data
import os

# computing
import numpy as np
import pandas as pd

# visualization
import matplotlib.pyplot as plt
import seaborn as sns

# feature extraction
import cv2
from scipy.spatial.distance import cdist
from sklearn.cluster import Kmeans

# modelling
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
import pickle

# evalutaion
from     sklearn.metrics      import      classification_report,      confusion_matrix,
accuracy_score

# deployment
from flask import Flask, render_template, request, send_from_directory
```

---

**Modul Program 4.1 Library yang Digunakan****4.1.1. Data Understanding**

Tahapan *data understanding* terdapat dua bagian di dalamnya yaitu proses pengumpulan data dan memahami data yang telah diambil.

Pada tahap pertama yaitu pengumpulan dataset dilakukan dengan mengambil data dari Internet yang bersumber dari platform Kaggle dan di-*publish* oleh akun Phiard namun telah dipilih beberapa data saja untuk menyesuaikan kebutuhan penelitian ini. Selanjutnya data digunakan dalam sistem dengan menggunakan modul program berikut. Pertama yang perlu disiapkan adalah bagaimana cara mengambil data gambar kedalam sistem sebagai berikut.

---

**Algoritma 2: Proses Import Data Gambar**

---

```
def import_image(file):
    image = cv2.imread(file)
    image = cv2.bitwise_not(image)
    return image
```

---

**Modul Program 4.2 Proses Import Gambar**

Selanjutnya proses pelabelan data juga dilakukan secara bersamaan ketika data sedang diambil. Berikut proses pelabelan data.

---

**Algoritma 3: Proses Pelabelan Data Gambar**

---

```
def label_image(character, length):
    array = np.full(length, character)
    return array
```

---

**Modul Program 4.3 Proses Pelabelan Gambar**

Kemudian kedua proses tersebut digabungkan dalam satu proses pengambilan data dari direktori berdasarkan nama karakter yang akan diambil. Prosesnya sebagai berikut.

---

**Algoritma 4: Proses Gabungan Import dan Pelabelan Data**

---

```
def import_data(character, path) :  
    file_path = os.listdir(path)  
    images = np.array([import_image(path + '/' + file) for file in file_path])  
    label = label_image(character, len(file_path))  
    return images, label
```

---

**Modul Program 4.4 Proses Gabungan Pengambilan Data**

Kode sumber diatas akan mengembalikan seluruh data gambar beserta labelnya berdasarkan karakter yang ingin diambil. Sebagai contoh ingin mengambil data huruf ‘nya’ maka hasil yang diperoleh adalah seluruh gambar dan label ‘nya’. Untuk dapat mengambil seluruh huruf dalam hancaraka maka perlu membuat perulangan dalam kode sumber tersebut. Beriku kode sumbernya.

---

**Algoritma 5: Proses Import Seluruh Data**

---

```
Hanacaraka= ('ba', 'ca', 'da', 'dha',  
'ga', 'ha', 'ja', 'ka', 'la',  
'ma', 'na', 'nga', 'nya', 'pa',  
'ra', 'sa', 'ta', 'tha', 'wa', 'ya')  
  
raw_X = np.empty((0, 224, 224, 3), np.uint8)  
raw_y = np.empty((0))  
  
folder_path = ".../Dataset/campuran/"  
  
for character in hanacaraka:  
    dataset_path = folder_path + character  
    temp_X, temp_y = import_data(character, dataset_path)  
  
    print("Size of", character, ":", temp_X.shape[0])  
  
    raw_X = np.append(raw_X, temp_X, axis=0)  
    raw_y = np.append(raw_y, temp_y, axis=0)
```

---

**Modul Program 4.5 Proses Import Seluruh Data**

Setelah semua data diambil selanjutnya pada bagian memahami data dapat dilakukan beberapa hal termasuk diantaranya yaitu mencari jumlah data yang telah diambil, mencari dimensi data, dan melihat hasilnya secara langsung bagaimana visualisasi gambar tersebut. Berikut cara mengetahui informasi tentang jumlah dan dimensi data.

---

**Algoritma 6: Mengetahui Jumlah dan Dimensi Data**

---

```
print("Number of data:", raw_X.shape[0])  
print("Image dimension:", raw_X[0].shape)
```

---

**Modul Program 4.6 Proses Mengetahui Jumlah dan Dimensi Data**

Kode diatas memberikan sebuah informasi bahwa jumlah data yang telah diambil sebesar 420 data pada seluruh kelas, maka jumlah data pada setiap kelas sebanyak 21 data, selanjutnya informasi dimensi data yaitu 224x224 piksel dengan memiliki 3 *channel* warna RGB yang dimiliki. Lalu untuk melihat secara langsung bagaimana visualisasi gambar yaitu dengan cara berikut.

---

**Algoritma 7: Visualisasi Data Secara Acak**

---

```
np.random.seed(42)
fig = plt.figure(figsize=(16, 4))
columns = 8
rows = 1
for i in range(1, columns*rows + 1):
    img = np.array(raw_X[np.random.randint(0, raw_X.shape[0]), :])
    fig.add_subplot(rows, columns, i)
    plt.imshow(img)
    plt.axis('off')
plt.show()
```

---

**Modul Program 4.7 Proses Visualisasi Data**

Pada modul program diatas akan mengeluarkan beberapa visualisasi gambar secara acak dari data yang tersedia.



**Gambar 4.2 Hasil Keluaran Modul Program 4.7**

#### 4.1.2. *Data Preparation*

Tahapan *data preparation* merupakan proses yang paling panjang dalam perancangan dan pembangunan sistem pada penelitian ini. Tahapan ini terbagi menjadi empat bagian yaitu diantaranya *data augmentation*, *image preprocessing*, *feature extraction*, *feature preprocessing*.

##### 1. *Data Augmentation & Image Preprocessing*

Karena kurangnya data yang diperoleh dari Kaggle, implementasi data augmentasi dilakukan bertujuan untuk menciptakan berbagai varian data yang baru sehingga kebutuhan data pada penilitian ini dapat terpenuhi. Terdapat tujuh varian yang akan dibuat telah dijelaskan pada bab 3 yaitu berkaitan dengan proses *rotation*, *zoom out*, dan *shear*. Dalam penulisan kode sumbernya dapat dilihat sebagai berikut.

---

**Algoritma 8: Tiga Fungsi Augmentasi**

---

```
# Augmented Method
def rotate(img, degree):
    (h, w) = img.shape[:2]
    (cX, cY) = (w // 2, h // 2)

    M = cv2.getRotationMatrix2D((cX, cY), degree, 1.0)
    img_rotated = cv2.warpAffine(img, M, (w, h))

    return img_rotated
```

---

**Modul Program 4.8 Fungsi Augmentasi**

---

**Algoritma 9: Tiga Fungsi Augmentasi (Lanjutan)**

---

```
def zoom_out(img, zoom_scale):
    img_zoom = np.zeros((img.shape[0], img.shape[1]), dtype=np.uint8)

    h, w = img.shape[0]/zoom_scale, img.shape[1]/zoom_scale
    h, w = int(h), int(w)
    x, y = (img.shape[0]-w)//2, (img.shape[1]-h)//2
    img_resize = cv2.resize(img, (h, w), interpolation = cv2.INTER_AREA)

    img_zoom[x:w+x, y:h+y] = img_resize
    return img_zoom

def shear(img, x, y):
    rows, cols = img.shape
    M = np.float32([[1, x, 0],
                    [y, 1, 0]])
    sheared_img = cv2.warpAffine(img, M, (cols, rows))
    return sheared_img
```

---

**Modul Program 4.9 Fungsi Augmentasi Lanjutan**

Ketiga fungsi-fungsi tersebut dipanggil dengan beberapa parameter-parameter yang berbeda sehingga akan menghasilkan varian yang berbeda juga. Proses augmentasi tersebut diimplementasikan bersamaan dengan proses *image preprocessing* pada data. Tahapan *image preprocessing* juga memiliki tiga proses yang dilakukan untuk memperbagus gambar yaitu diantaranya *resize*, *grayscale*, dan *equalization histogram*. Berikut merupakan implementasi kode fungsi yang digunakan dalam *image preprocessing*.

---

**Algoritma 10: Tiga Fungsi Image Preprocessing**

---

```
# Preprocessing Image
def equalizing(img):
    if (img.ndim == 3):
        b, g, r = cv2.split(img)
        red = cv2.equalizeHist(r)
        green = cv2.equalizeHist(g)
        blue = cv2.equalizeHist(b)
        img = cv2.merge((blue, green, red))
    else:
        img = cv2.equalizeHist(img)
    return img

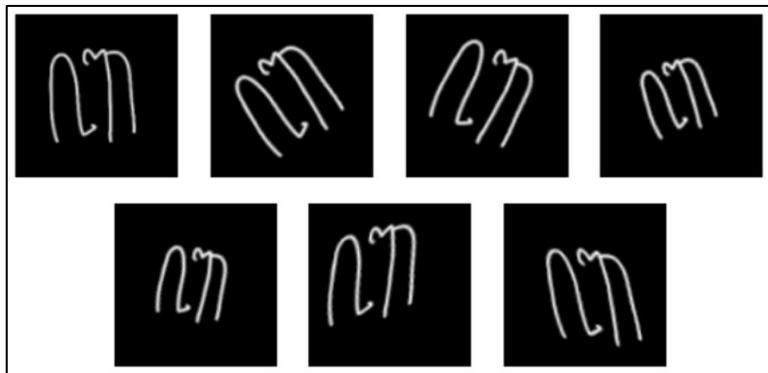
def grayscaling(image):
    image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
    return image

def resizing(image, size):
    image = cv2.resize(image, (size, size), interpolation = cv2.INTER_AREA)
    return image
```

---

**Modul Program 4.10 Fungsi Image Preprocessing**

Setelah semua proses tersebut dijalankan, data gambar akan bertambah hingga menjadi sebesar 2940 data dengan 147 data pada setiap kelas. Berikut sampel yang dihasilkan dari data augmentasi dan *image preprocessing* pada kelas huruf “ba”.



**Gambar 4.3 Hasil Augmentasi dan *Preprocessing***

## 2. *Feature Extraction*

Tahapan ini mengimplementasikan metode SIFT dalam ekstraksi fitur dan K-Means serta BoW sebagai metode tambahan untuk menunjang menyediakan data yang relevan terhadap kebutuhan model.

Metode SIFT diimplementasikan menggunakan *library* dari opencv-python yang bersifat *open-source*, namun sayangnya detail kode yang ada di dalam *library* tersebut tidak dapat dilihat/dibuka. Proses ekstraksi dari SIFT terlihat sederhana seperti berikut dengan memasukkan parameter dari daftar gambar yang akan diekstrak.

---

### **Algoritma 11: Ekstraksi Fitur SIFT**

---

```
def extract_sift_features(list_image):
    image_descriptors = []
    sift = cv2.SIFT_create()
    for image in list_image:
        _, descriptor = sift.detectAndCompute(image, None)
        image_descriptors.append(descriptor)

    return image_descriptors
```

---

### **Modul Program 4.11 Fungsi Ekstraksi Fitur SIFT**

Modul program diatas akan mengembalikan seluruh hasil ekstraksi fitur dari setiap gambar dalam bentuk deskriptor ukuran 128. Berikut ini gambar hasil ekstraksi fitur SIFT dengan beberapa *keypoint* yang dihasilkan.



**Gambar 4.4 Hasil Ekstraksi Fitur SIFT**

Selanjutnya yaitu membuat BoW dengan menggunakan metode klustering dari K-Means. Untuk membuat BoW tersebut membutuhkan input parameter dari

gabungan seluruh deskriptor hasil ekstraksi fitur pada setiap gambar dan input data jumlah klaster sebagai jumlah data/kamus dalam BoW. Berikut ini adalah fungsi dari proses pembuatan BoW.

---

**Algoritma 12: Pembuatan BoW**

---

```
def kmeans_bow(all_descriptors, num_cluster):
    if not os.path.isfile(BOW_FILE_PICKLE):
        kmeans = KMeans(n_clusters = num_cluster, verbose=1)
        kmeans.fit(all_descriptors)
        bow_dict = kmeans.cluster_centers_
        save_file_pickle(bow_dict, BOW_FILE_PICKLE)
    else:
        bow_dict = load_file_pickle(BOW_FILE_PICKLE)

    return bow_dict
```

---

**Modul Program 4.12 Fungsi Pembuatan BoW**

Dari modul program diatas bahwa fungsi yang dipanggil akan mengembalikan seluruh nilai titik dari seluruh klaster hasil klustering. Seluruh nilai titik klaster ini yang akan menjadi data-data di dalam kamus BoW. Jumlah klaster ditentukan dari parameter yang dimasukkan pada fungsi. Selain itu, di dalam proses fungsi tersebut terdapat proses menyimpan BoW dalam bentuk *file pickle* yang bertujuan agar proses tidak akan dilakukan berulang ketika telah dijalankan sebelumnya.

Tahapan terakhir pada ekstraksi fitur adalah membuat fitur akhir dari BoW yang telah dibuat sebelumnya dengan menginput setiap gambar. Fitur akhir ini akan berukuran NxM dengan N banyak data gambar yang akan diekstrak dan M jumlah klaster yang telah ditentukan sebelumnya. Berikut implementasi proses pembuatan fitur akhir dari BoW dengan memasukkan beberapa parameter yaitu deskriptor SIFT dari setiap gambar, kamus BoW yang digunakan, dan jumlah klaster.

---

**Algoritma 13: Pembuatan Fitur dari BoW**

---

```
def create_feature_bow(image_descriptors, bow, num_cluster):
    X_features = []

    for i in range(len(image_descriptors)):
        features = np.array([0] * num_cluster, dtype=float)

        if image_descriptors[i] is not None:
            distance = cdist(image_descriptors[i], bow)
            argmin = np.argmin(distance, axis = 1)

            for j in argmin:
                features[j] += 1.0
            X_features.append(features)

    return np.array(X_features)
```

---

**Modul Program 4.13 Fungsi Pembuatan Fitur dari BoW**

Dari modul program diatas, proses dilakukan dengan menghitung jarak setiap fitur deskriptor dengan seluruh klaster dari BoW menggunakan metode euclidean distance, kemudian nilai deskriptor akan dikelompokkan pada klaster yang memiliki jarak terpendek, selanjutnya menambahkan nilai 1 pada indeks kelompok klaster tersebut.

### 3. Feature Preprocessing

Tahapan *feature preprocessing* pada penelitian ini terbagi menjadi dua proses yaitu melakukan *feature scaling* dan *splitting dataset*. Pertama *feature scaling* dilakukan menggunakan metode normalisasi *minmaxscaling* dengan menskalakan nilai fitur dalam rentang 0-1, metode ini dapat diimplementasikan dengan mudah menggunakan *library* dari *scikit-learn* dengan memanggil fungsi *MinMaxScaler*, kemudian yang kedua splitting dataset akan memecahkan dataset menjadi data latih dan data uji dengan rasio 85% data latih dan 15% data uji. Tahapan *splitting dataset* ini juga diimplementasikan menggunakan *library* dari *scikit-learn* dengan memanggil fungsi *train\_test\_split* fungsi ini membutuhkan beberapa parameter diantaranya data fitur, data label, ukuran pemecahan data, dan *random\_state* sebagai nilai acak yang sama setiap pemanggilan fungsi. Kedua tahapan tersebut dapat diimplementasikan dalam bentuk kode sebagai berikut.

---

**Algoritma 14: Feature Preprocessing**

---

```
# normalization
scaler = MinMaxScaler().fit(X_feature)
X_scale = scaler.transform(X_feature)

# splitting data
X_train, X_test, y_train, y_test = train_test_split(
    X_scale,
    Y,
    test_size=0.15,
    random_state=42
)
```

---

**Modul Program 4.14 Fungsi Feature Preprocessing**

Dari modul program diatas dapat dilihat bahwa hasil akhir yang dikembalikan adalah data X\_train, X\_test, y\_train, y\_test. Dari data tersebut sudah siap digunakan untuk melakukan pelatihan model SVM di tahapan berikutnya.

#### 4.1.3. Modelling

Tahapan *modelling* merupakan sebuah proses pelatihan model *machine learning* dengan menggunakan metode SVM. Saat yang bersamaan *hyperparameter tuning* pada model SVM juga dijalankan untuk mencari parameter terbaik dalam hasil pelatihan model agar mendapatkan model yang optimal. Proses pelatihan model ini diimplementasikan dengan menerapkan *library* dari *scikit-learn* pada fungsi kelas SVC dan GridSearchCV. Kelas SVC digunakan dalam pelatihan model SVM sedangkan kelas GridSearchCV digunakan dalam proses *hyperparameter tuning* sebagai optimasi model dengan menerapkan konsep *k-fold cross validation*.

Parameter SVM yang akan dioptimasi pada penelitian ini adalah parameter C sebagai nilai toleransi SVM dan parameter gamma sebagai nilai parameter kernel RBF. Berikut adalah implementasi kode sumber dalam pelatihan model bersamaan dengan optimasi parameternya.

---

**Algoritma 15: Pelatihan SVM & Optimasi Model**

---

```
# Tuning Parameter
svm_ws_params = {
    'C': [1, 3, 6, 10, 15],
    'gamma': [0.10234, 0.10447, 0.1066, 0.10873, 0.11086]
}
# Grid Search
svm_ws = GridSearchCV(
    estimator=SVC(kernel='rbf', probability=True),
    param_grid=svm_ws_params,
    cv=5, verbose=3
)
# Training SVM
svm_ws.fit(X_train, y_train)
# Save model
save_file_pickle(svm_ws, SVM_WS_FILE_PICKLE)
```

---

**Modul Program 4.15 Pelatihan SVM & Optimasi Model**

Dari modul program diatas dapat dilihat bahwa fungsi kelas GridSearchCV membutuhkan empat parameter yaitu diantaranya *estimator*, *param\_grid*, *cv*, dan *verbose*. *Estimator* menampung metode *machine learning* yang akan digunakan dalam pelatihan model, pada penelitian ini metode yang digunakan adalah SVC yang membutuhkan dua parameter yaitu kernel yang digunakan yaitu RBF dan *probability* sebagai keluaran data saat ingin melakukan prediksi model agar dapat melihat nilai probabilitas dari hasil prediksi. *Param\_grid* menampung beberapa parameter yang ingin dilakukan *tuning* untuk mencari yang terbaik. Parameter CV menampung jumlah *cross validation* yang dilakukan setiap pelatihan dalam satuan kombinasi parameter. Dan terakhir parameter *verbose* yang digunakan sebagai keluaran *log* untuk memonitoring hasil pelatihan.

Setelah kelas GridSearchCV diinisialisasikan selanjutnya melakukan pelatihan model dengan memanggil fungsi *fit* pada variabel *svm\_ws*, fungsi ini membutuhkan dua parameter penting yaitu fitur data dan label data yang akan dilatih. Berikut tampilan proses pelatihan model beserta optimasi parameter.

```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
[CV 1/5] END .....C=1, gamma=0.10234, score=0.898 total time= 8.3s
[CV 2/5] END .....C=1, gamma=0.10234, score=0.884 total time= 8.3s
[CV 3/5] END .....C=1, gamma=0.10234, score=0.932 total time= 8.0s
[CV 4/5] END .....C=1, gamma=0.10234, score=0.894 total time= 7.9s
[CV 5/5] END .....C=1, gamma=0.10234, score=0.916 total time= 7.9s
[CV 1/5] END .....C=1, gamma=0.10447, score=0.898 total time= 7.8s
[CV 2/5] END .....C=1, gamma=0.10447, score=0.884 total time= 7.8s
[CV 3/5] END .....C=1, gamma=0.10447, score=0.932 total time= 7.9s
[CV 4/5] END .....C=1, gamma=0.10447, score=0.902 total time= 7.8s
[CV 5/5] END .....C=1, gamma=0.10447, score=0.916 total time= 7.8s
[CV 1/5] END .....C=1, gamma=0.1066, score=0.898 total time= 7.9s
[CV 2/5] END .....C=1, gamma=0.1066, score=0.884 total time= 7.9s
[CV 3/5] END .....C=1, gamma=0.1066, score=0.932 total time= 7.9s
[CV 4/5] END .....C=1, gamma=0.1066, score=0.902 total time= 7.9s
[CV 5/5] END .....C=1, gamma=0.1066, score=0.916 total time= 7.9s
[CV 1/5] END .....C=1, gamma=0.10873, score=0.898 total time= 7.9s
[CV 2/5] END .....C=1, gamma=0.10873, score=0.882 total time= 7.8s
[CV 3/5] END .....C=1, gamma=0.10873, score=0.932 total time= 7.8s
[CV 4/5] END .....C=1, gamma=0.10873, score=0.908 total time= 7.8s
[CV 5/5] END .....C=1, gamma=0.10873, score=0.914 total time= 7.8s
[CV 1/5] END .....C=1, gamma=0.11086, score=0.896 total time= 7.8s
[CV 2/5] END .....C=1, gamma=0.11086, score=0.882 total time= 7.9s
[CV 3/5] END .....C=1, gamma=0.11086, score=0.932 total time= 7.9s
[CV 4/5] END .....C=1, gamma=0.11086, score=0.908 total time= 7.9s
[CV 5/5] END .....C=1, gamma=0.11086, score=0.922 total time= 7.9s
...
[CV 2/5] END .....C=15, gamma=0.11086, score=0.900 total time= 9.3s
[CV 3/5] END .....C=15, gamma=0.11086, score=0.932 total time= 8.0s
[CV 4/5] END .....C=15, gamma=0.11086, score=0.920 total time= 8.0s
[CV 5/5] END .....C=15, gamma=0.11086, score=0.932 total time= 8.0s
GridSearchCV(cv=5, estimator=SVC(probability=True),
    param_grid={'C': [1, 3, 6, 10, 15],
                'gamma': [0.10234, 0.10447, 0.1066, 0.10873, 0.11086]},
    verbose=3)
```

Gambar 4.5 Proses Pelatihan dan Optimasi Parameter

Ketika proses pelatihan dan optimasi selesai dijalankan, model akan disimpan dalam bentuk *file pickle* dengan tujuan agar model dapat digunakan kembali tanpa perlu melakukan pelatihan ulang ketika ingin melakukan proses prediksi. Selanjutnya dari model tersebut hasil optimasi parameter dan nilai akurasi optimal dari parameter yang dipilih dapat ditampilkan dengan cara seperti berikut.

---

**Algoritma 16: Hasil Pelatihan dari Optimasi Parameter**

---

```
print("Best Model:", svm_ws.best_estimator_)
print("Best CV Score:", svm_ws.best_score_)
```

---

**Modul Program 4.16 Proses Menampilkan Hasil Pelatihan dan Optimasi**

Modul program diatas akan menampilkan model dengan parameter yang terbaik dan nilai akurasi dari model hasil parameter tersebut. Berikut hasil yang ditampilkan.

```
Best Model: SVC(C=6, gamma=0.10234, probability=True)
Best CV Score: 0.9211735470941884
```

**Gambar 4.6 Tampilan Hasil Pelatihan dan Optimasi**

Gambar diatas menunjukkan bahwa parameter terbaik yang diperoleh adalah  $C = 6$  dan  $gamma = 0.10234$  dengan nilai akurasi sebesar 92.11%.

Model SVM dilatih dan menemukan parameter yang terbaik langkah selanjutnya yaitu melakukan prediksi terhadap model. Untuk melakukan prediksi terhadap model terdapat dua langkah yang harus dilakukan yaitu mengambil model yang telah disimpan sebelumnya dari hasil pelatihan dan memanggil fungsi *predict* dengan memasukkan parameter data gambar yang akan diprediksi. Berikut implementasi kode yang digunakan.

---

**Algoritma 17: Prediksi**

---

```
# Load model
svm_ws = load_file_pickle(SVM_WS_FILE_PICKLE)
# Predict model
y_pred = svm_ws.predict(X_test)
```

---

**Modul Program 4.17 Proses Prediksi Model****4.1.4. Evaluation**

Meskipun pengukuran performa pada penelitian ini berfokus pada tingkat akurasi terhadap model, namun implementasi yang dilakukan tidak hanya mencari nilai akurasi dari suatu model yang digunakan. Terdapat tiga proses evaluasi yang dilakukan sebelum mengambil kesimpulan apakah tingkat akurasi sebuah model layak/baik untuk diakui. Tiga proses tersebut yaitu melihat hasil *confusion matrix* pada setiap kelas yang diprediksi, melihat hasil dari nilai *F1-score* pada setiap kelas sebagai tingkat nilai *confidence* pada hasil prediksi, dan terakhir melihat nilai akurasi keseluruhan prediksi.

Proses pembuatan *confusion matrix* dapat dilakukan secara mudah dengan menggunakan bantuan *library* dari *scikit-learn* pada fungsi *confussion\_matrix*. Fungsi ini membutuhkan dua parameter yaitu data label aktual dan data label hasil prediksi. Berikut implementasi pada kode yang dibuat.

---

**Algoritma 18: Confussion Matrix**

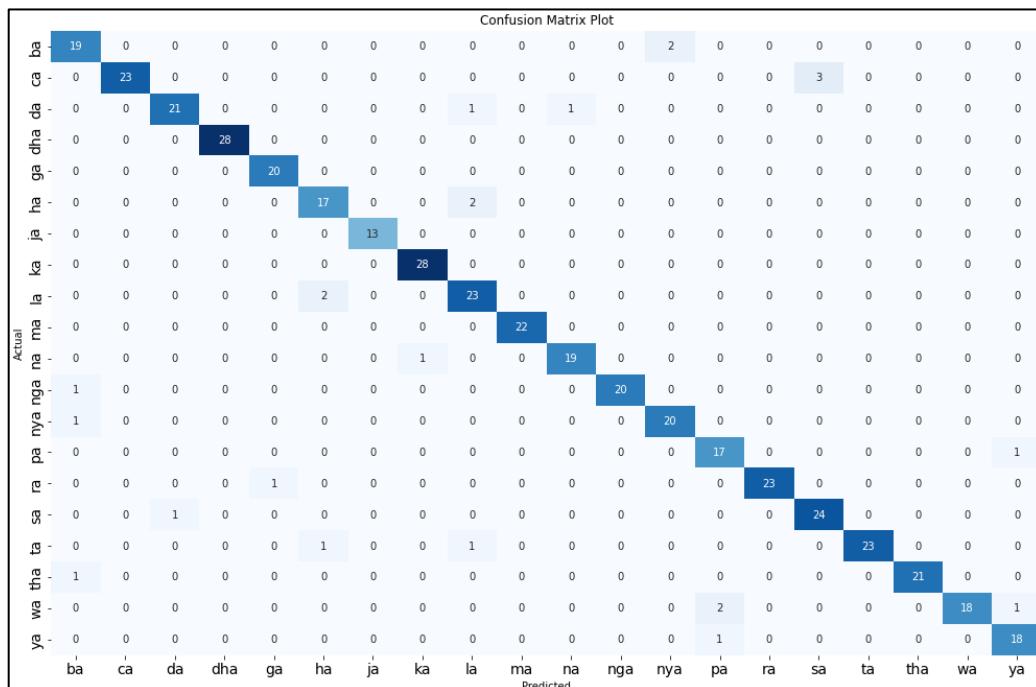
---

```
conf_ws = pd.DataFrame(  
    confusion_matrix(y_test, y_pred),  
    columns=hanacaraka,  
    index=hanacaraka  
)  
  
plt.figure(figsize=(17,11))  
res = sns.heatmap(conf_ws, annot=True, fmt="g", cmap="Blues", cbar=False)  
res.set_yticklabels(res.get_ymajorticklabels(), fontsize = 14)  
res.set_xticklabels(res.get_xmajorticklabels(), fontsize = 14)  
plt.xlabel("Predicted")  
plt.ylabel("Actual")  
plt.title("Confusion Matrix Plot")  
plt.show()
```

---

**Modul Program 4.18 Proses Menampilkan Confussion Matrix**

Hasil modul program diatas apabila dijalankan akan memberikan tampilan plot *confussion matrix* pada setiap kelas seperti berikut.

**Gambar 4.7 Hasil Plot Confussion Matrix**

Selanjutnya melihat nilai *F1-Score* pada setiap kelas dapat dilakukan seperti sebelumnya dengan bantuan *scikit-learn* menggunakan fungsi *classification\_report* yang membutuhkan dua parameter yaitu data label aktual dan data label hasil prediksi. Berikut implementasi kode yang digunakan.

---

**Algoritma 19: Classification Report**

---

```
class_rep_ws = classification_report(y_test, y_pred)  
  
print("Classification report result:")  
print(class_rep_ws)
```

---

**Modul Program 4.19 Proses Menampilkan Classification Report**

Hasil modul program diatas apabila dijalankan akan memberikan keluaran berupa nilai *precision*, *recall*, *f1-score*, dan *support* pada setiap kelas. Berikut hasil yang dikeluarkan.

Classification report result:				
	precision	recall	f1-score	support
ba	0.95	0.90	0.93	21
ca	1.00	0.96	0.98	26
da	1.00	0.96	0.98	23
dha	0.96	0.93	0.95	28
ga	0.87	1.00	0.93	20
ha	0.78	0.95	0.86	19
ja	1.00	1.00	1.00	13
ka	1.00	1.00	1.00	28
la	0.86	0.76	0.81	25
ma	1.00	0.95	0.98	22
na	0.95	1.00	0.98	20
nga	1.00	1.00	1.00	21
nya	0.91	0.95	0.93	21
pa	0.98	1.00	0.95	18
ra	1.00	0.83	0.91	24
sa	0.96	1.00	0.98	25
ta	1.00	0.96	0.98	25
tha	1.00	1.00	1.00	22
wa	0.91	0.95	0.93	21
ya	1.00	1.00	1.00	19
accuracy			0.95	441
macro avg	0.95	0.96	0.95	441
weighted avg	0.96	0.95	0.95	441

**Gambar 4.8 Hasil Classification Report**

Setelah semuanya yakin mendapatkan hasil yang cukup bagus, tahap terakhir yaitu menghitung nilai akurasi secara keseluruhan model. Proses ini juga dilakukan dengan bantuan *scikit-learn* pada fungsi *accuracy\_score* yang membutuhkan dua parameter yaitu data label aktual dan data label hasil prediksi. Berikut implementasi kode yang digunakan.

---

#### Algoritma 20: Accuracy Score

---

```
acc_ws = accuracy_score(y_test, y_pred)
print("Accuracy score:", acc_ws)
```

---

#### Modul Program 4.20 Proses Menghitung Accuracy Score

Hasil modul program diatas apabila dijalankan hanya memberikan keluaran berupa nilai akurasi model seperti berikut.

Accuracy score: 0.9523809523809523

**Gambar 4.9 Hasil Perhitungan Akurasi**

#### 4.1.5. Deployment

*Deployment* merupakan tahapan terakhir dalam metodologi penelitian ini. Pada tahap ini proses dilakukan dengan membangun aplikasi berbasis *website* yang terdapat fitur *machine learning* di dalamnya. Untuk melakukan hal tersebut, *deployment* dilakukan dengan bantuan dari *library* Flask yang menyediakan integrasi bahasa *python* dengan pemrograman web. Berikut merupakan implementasi penggunaan Flask dalam melakukan prediksi sebuah gambar.

---

**Algoritma 21: Prediksi Gambar Menggunakan Flask**

---

```
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = './static/uploads/'

@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if request.files:
            image = request.files['image']
            img_path = os.path.join(app.config['UPLOAD_FOLDER'], image.filename)
            image.save(img_path)
            prediction, accuracy = predict_process(img_path)
            return render_template('index.html', uploaded_image=image.filename,
prediction=prediction, accuracy=accuracy)

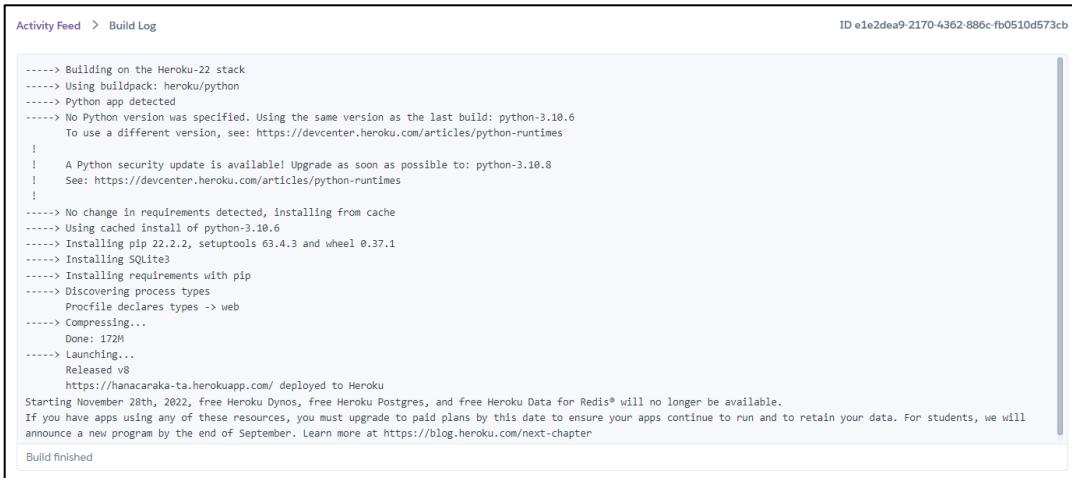
    return render_template('index.html')
```

---

**Modul Program 4.21 Proses Prediksi Gambar Menggunakan *Flask***

Pada modul program diatas fungsi *Flask* diinisialisasi terlebih dahulu, kemudian kode `@app.route` digunakan untuk memanggil halaman dengan fungsi *index* untuk prediksi gambar apabila terdapat *request post*. Proses prediksi dilakukan setelah menerima gambar dari inputan pengguna, kemudian memanggil fungsi *predict\_process()*.

Kemudian seluruh kode program yang telah dibangun akan diunggah ke dalam *platform github* agar bisa terintegrasi dengan *internet*. Kode yang ada di *github* tersebut selanjutnya diintegrasikan dengan *platform heroku* sebagai penyedia layanan server gratis. Dari *heroku* kode program akan dilakukan *build* atau *deploy* aplikasi dan mendapatkan *link* aplikasi yang sudah selesai di-*build*. Berikut tampilan proses *build/deploy* aplikasi dari *heroku*.



The screenshot shows a terminal window with the title "Activity Feed > Build Log" and an ID "e1e2dea9-2170-4362-886c-fb0510d572cb". The log output is as follows:

```
Activity Feed > Build Log
ID e1e2dea9-2170-4362-886c-fb0510d572cb

-----> Building on the Heroku-22 stack
-----> Using buildpack: heroku/python
-----> Python app detected
-----> No Python version was specified. Using the same version as the last build: python-3.10.6
      To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
!
!   A Python security update is available! Upgrade as soon as possible to: python-3.10.8
!   See: https://devcenter.heroku.com/articles/python-runtimes
!
-----> No change in requirements detected, installing from cache
-----> Using cached install of python-3.10.6
-----> Installing pip 22.2.2, setuptools 63.4.3 and wheel 0.37.1
-----> Installing SQLite
-----> Installing requirements with pip
-----> Discovering process types
      Procfile declares types -> web
-----> Compressing...
      Done: 172M
-----> Launching...
      Released v8
      https://hanacaraka-ta.herokuapp.com/ deployed to Heroku
Starting November 28th, 2022, free Heroku Dynos, free Heroku Postgres, and free Heroku Data for Redis® will no longer be available.
If you have apps using any of these resources, you must upgrade to paid plans by this date to ensure your apps continue to run and to retain your data. For students, we will announce a new program by the end of September. Learn more at https://blog.heroku.com/next-chapter
Build finished
```

**Gambar 4.10 Proses *Build/Deploy Heroku***

Hasil *build* aplikasi tersebut dapat diakses pada *link* berikut yaitu <https://hanacaraka-ta.herokuapp.com>.

## 4.2. Hasil

Pada bagian ini akan dijelaskan bagaimana hasil pengujian terhadap model yang dibangun, dimana skenario pengujian telah dijabarkan pada bab sebelumnya. Skenario pengujian klasifikasi akan dibagi menjadi 20 bagian skenario sesuai dengan Tabel 3.19. dan ditambahkan dengan pengujian perbandingan hasil model SVM-SIFT terbaik terhadap hasil model SVM tanpa metode SIFT. Pada pengujian dilakukan dengan bermacam-macam optimasi parameter diantaranya yaitu parameter *Size*, nilai K pada K-Means atau jumlah fitur, nilai C sebagai nilai parameter tolenrasmi SVM, dan nilai gamma (G) sebagai nilai parameter kernel RBF. Selanjutnya pengujian juga dilakukan dengan mengambil beberapa sampel gambar yang akan diklasifikasi.

### 4.2.1. Pengujian Model

Seperti yang dijelaskan sebelumnya beberapa parameter yang akan dioptimasi adalah parameter *Size*, nilai K, nilai C, dan nilai G. Pada parameter *Size* terdapat 4 nilai yang akan dikombinasikan dalam satuan piksel yaitu 128x128, 160x160, 192x192, dan 224x224. Kemudian pada parameter nilai K terdapat 5 nilai yang akan dikombinasikan yaitu 180, 250, 500, 750, dan 1000. Selanjutnya pada parameter nilai C dan G memiliki nilai yang dinamis menyesuaikan ukuran dan distribusi masing-masing data. Berikut adalah tabel hasil optimasi pengujian parameter yang dilakukan.

**Tabel 4.1 Hasil Pengujian Optimasi Parameter SVM-SIFT**

#	Size	K	C	G	R_Train	R_Test
1	128	180	[1,3,6,10,15]	[ <b>0.24574</b> , 0.25086, 0.25598, 0.2611, 0.26622]	89.31%	92.51%
2		250	[1,3,6,10,15]	[ <b>0.20504</b> , 0.20931, 0.21358, 0.21785, 0.22212]	90.63%	92.97%
3		500	[1,3,6,10,15]	[ <b>0.13567</b> , 0.1385, 0.14133, 0.14416, 0.14699]	91.55%	95.01%
4		750	[1,3,6,10,15]	[ <b>0.10508</b> , 0.10727, 0.10946, 0.11165, 0.11384]	91.99%	94.10%
5		1000	[1,3,6,10,15]	[ <b>0.09142</b> , 0.09332, 0.09522, 0.09712, 0.09902]	91.27%	94.55%
6	160	180	[1,3,6,10,15]	[ <b>0.25495</b> , 0.26026, 0.26557, 0.27088, 0.27619]	89.07%	91.69%
7		250	[1,3,6,10,15]	[0.20114, <b>0.20533</b> , 0.20952, 0.21371, 0.2179]	89.95%	90.92%
8		500	[1,3,6,10,15]	[ <b>0.13022</b> , 0.13293, 0.13564, 0.13835, 0.14106]	90.63%	91.83%
9		750	[1,3,6,10,15]	[ <b>0.10009</b> , 0.10217, 0.10425, 0.10633, 0.10841]	90.79%	93.19%
10		1000	[1,3,6,10,15]	[ <b>0.08725</b> , 0.08907, 0.09089, 0.09271, 0.09453]	90.07%	92.97%
11	192	180	[1,3,6,10,15]	[0.26274, <b>0.26821</b> , 0.27368, 0.27915, 0.28462]	88.47%	90.92%
12		250	[1,3,6,10,15]	[ <b>0.20183</b> , 0.20603, 0.21023, 0.21443, 0.21863]	90.91%	91.60%
13		500	[1,3,6,10,15]	[0.12668, 0.12932, <b>0.13196</b> , 0.1346, 0.13724]	91.83%	93.42%
14		750	[1,3,6,10,15]	[ <b>0.10235</b> , 0.10448, 0.10661, 0.10874, 0.11087]	92.11%	94.55%
15		1000	[1,3,6,10,15]	[ <b>0.08487</b> , 0.08664, 0.08841, 0.09018, 0.09195]	92.67%	95.01%
16	224	180	[1,3,6, <b>10</b> ,15]	[0.23885, <b>0.24383</b> , 0.24881, 0.25379, 0.25877]	85.83%	90.47%
17		250	[1,3,6,10,15]	[ <b>0.19131</b> , 0.1953, 0.19929, 0.20328, 0.20727]	87.99%	90.92%
18		500	[1,3,6,10,15]	[0.11421, <b>0.11659</b> , 0.11897, 0.12135, 0.12373]	89.19%	93.19%
19		750	[1,3,6,10,15]	[ <b>0.09026</b> , 0.09214, 0.09402, 0.0959, 0.09778]	90.91%	92.51%
20		1000	[1,3,6,10,15]	[ <b>0.07609</b> , 0.07767, 0.07925, 0.08083, 0.08241]	90.47%	92.74%

Pada tabel diatas menjelaskan hasil akurasi terbaik pada seluruh 20 skenario yang mana pada parameter C dan G terdapat 5 varian nilai parameter setiap skenario, kedua parameter tersebut yang dicetak tebal adalah hasil parameter terbaik dari 5 varian yang ada. Kemudian pada kolom R\_Train adalah hasil nilai akurasi yang dilakukan pada dataset latih sedangkan pada kolom R\_Test adalah hasil nilai akurasi yang dialakukan pada dataset uji.

Hasil pengujian parameter tersebut telah menemukan parameter C dan G yang terbaik. Sehingga untuk parameter Size dan K dapat dilihat dengan menyimpulkan datanya dalam perhitungan rata-rata pada akurasi terkait, berikut adalah tabel hasil kesimpulan yang dirumuskan pada kombinasi parameter Size.

**Tabel 4.2 Rata-rata Akurasi Terhadap Parameter Size**

Size	R_Train	R_Test
128	90.95%	93.83%
160	90.10%	92.12%
<b>192</b>	<b>91.20%</b>	<b>93.10%</b>
224	88.88%	91.97%

Kemudian berikut adalah tabel hasil kesimpulan yang dirumuskan pada kombinasi parameter nilai K.

**Tabel 4.3 Rata-rata Akurasi Terhadap Parameter K**

K	R_Train	R_Test
180	88.17%	91.40%
250	89.87%	91.60%
500	90.80%	93.36%
<b>750</b>	<b>91.45%</b>	<b>93.59%</b>
1000	91.12%	93.82%

Kedua data diatas dapat dilihat bahwa pada parameter *Size* nilai konsisten yang terbaik adalah pada ukuran gambar 192x192 piksel, sedangkan pada parameter K nilai konsisten yang terbaik adalah pada nilai 750. Kedua parameter ini dapat menghasilkan nilai akurasi yang sama diatas 91% pada dataset latih dan diatas 93% pada dataset uji.

Sehingga nilai keempat parameter tersebut yang terbaik yaitu pada skenario nomor 14 yang mana parameter *Size* bernilai 192, parameter K bernilai 750, parameter C bernilai 6, dan parameter G bernilai 0.10235. Keempat parameter tersebut dapat memperoleh tingkat akurasi yang tinggi yaitu 92.11% pada dataset latih dan 94.55% pada dataset uji. Melihat hasil performa akurasi tersebut maka parameter itu yang akan digunakan pada hasil akhir pelatihan model SVM yang dilakukan.

Detail dari hasil pelatihan dan pengujian model yang dilakukan pada setiap skenario dapat dilihat pada bagian lampiran dari laporan penelitian ini. Berikut ini adalah akurasi hasil pelatihan SVM-SIFT termasuk optimasi GridSearchCV yang dilakukan menggunakan parameter terbaik yang sudah ditentukan sebelumnya yaitu pada skenario 14.

**Tabel 4.4 Hasil Pelatihan SVM-SIFT Skenario 14**

kernel	params	rank test score	mean test score	std test score
15_0.10235	{'C': 15, 'gamma': 0.10235}	1	0.921174	0.011467
10_0.10235	{'C': 10, 'gamma': 0.10235}	1	0.921174	0.011467
6_0.10235	{'C': 6, 'gamma': 0.10235}	1	0.921174	0.011467
6_0.10661	{'C': 6, 'gamma': 0.10661}	4	0.920774	0.011535
15_0.10661	{'C': 15, 'gamma': 0.10661}	4	0.920774	0.011535
15_0.10448	{'C': 15, 'gamma': 0.10448}	4	0.920774	0.011039
10_0.10661	{'C': 10, 'gamma': 0.10661}	4	0.920774	0.011535
10_0.10448	{'C': 10, 'gamma': 0.10448}	4	0.920774	0.011039
6_0.10448	{'C': 6, 'gamma': 0.10448}	4	0.920774	0.011039
10_0.10874	{'C': 10, 'gamma': 0.10874}	10	0.920773	0.011043
15_0.10874	{'C': 15, 'gamma': 0.10874}	10	0.920773	0.011043
6_0.10874	{'C': 6, 'gamma': 0.10874}	10	0.920773	0.011043
10_0.11087	{'C': 10, 'gamma': 0.11087}	13	0.919973	0.011771
6_0.11087	{'C': 6, 'gamma': 0.11087}	13	0.919973	0.011771
15_0.11087	{'C': 15, 'gamma': 0.11087}	13	0.919973	0.011771
3_0.10235	{'C': 3, 'gamma': 0.10235}	16	0.919574	0.012767
3_0.10661	{'C': 3, 'gamma': 0.10661}	17	0.919174	0.012904
3_0.10448	{'C': 3, 'gamma': 0.10448}	17	0.919174	0.012462
3_0.10874	{'C': 3, 'gamma': 0.10874}	19	0.918374	0.013077
3_0.11087	{'C': 3, 'gamma': 0.11087}	20	0.917574	0.013906
1_0.10235	{'C': 1, 'gamma': 0.10235}	21	0.907567	0.016443
1_0.10448	{'C': 1, 'gamma': 0.10448}	22	0.907167	0.016500
1_0.10874	{'C': 1, 'gamma': 0.10874}	23	0.906766	0.016648
1_0.10661	{'C': 1, 'gamma': 0.10661}	23	0.906766	0.016259
1_0.11087	{'C': 1, 'gamma': 0.11087}	25	0.905966	0.016524

Dari tabel diatas dapat dilihat bahwa terdapat lebih dari satu hasil terbaik yang diperoleh dalam pelatihan model yaitu pada parameter C yang bernilai 15, 10, 6. Namun pada kasus penelitian ini parameter yang terbaik akan diambil dari nilai parameter yang terkecil karena apabila semakin kecil nilai yang digunakan maka kompleksitas perhitungan akan semakin berkurang.

Kemudian pengujian juga dilakukan dengan mencoba prediksi seluruh gambar pada dataset uji dan menampilkan hasilnya dalam bentuk *confusion matrix* yang berisi perbandingan jumlah kebenaran hasil label prediksi dengan data label yang aktual. *Confusion matrix* tersebut akan mempresentasikan seluruh 20 label yang ada pada penelitian ini. Berikut adalah gambar hasil *confusion matrix* tersebut.

		Confusion Matrix Plot																			
		Confusion Matrix Plot																			
Actual	Predicted	ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
		19	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
ba	ca	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
ca	da	0	0	21	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
da	dha	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
dha	ga	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ga	ha	0	0	0	0	0	17	0	0	2	0	0	0	0	0	0	0	0	0	0	0
ha	ja	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
ja	ka	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
ka	la	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0
la	ma	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0
ma	na	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0
na	nga	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
nga	nya	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
nya	pa	1	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0
pa	ra	0	0	0	0	0	1	0	0	0	0	0	0	0	0	23	0	0	0	0	0
ra	sa	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	24	0	0	0	0
sa	ta	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	23	0	0	0
ta	tha	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
tha	wa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	1	0
wa	ya	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	18
ya	ba	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Gambar 4.11 Hasil Confusion Matrix SVM-SIFT Skenario 14

Dari gambar *confusion matrix* diatas terlihat bahwa model dapat melakukan klasifikasi dengan baik dengan tingkat kesalahan yang kecil pada rata-rata terdapat hanya 1 atau 2 gambar kesalahan yang terjadi pada setiap labelnya.

Terakhir model diuji dengan menghitung nilai *classification report* (*precision*, *recall*, dan *F1-Score* pada setiap labelnya), serta menghitung total akurasi pada dataset uji. Pengujian ini dilakukan dengan tujuan mencari nilai *confidence* pada model tersebut agar dapat valid dengan tingkat akurasi yang diperoleh. Berikut adalah hasil perhitungannya.

Classification report result:				
	precision	recall	f1-score	support
ba	0.86	0.90	0.88	21
ca	1.00	0.88	0.94	26
da	0.95	0.91	0.93	23
dha	1.00	1.00	1.00	28
ga	0.95	1.00	0.98	20
ha	0.85	0.89	0.87	19
ja	1.00	1.00	1.00	13
ka	0.97	1.00	0.98	28
la	0.85	0.92	0.88	25
ma	1.00	1.00	1.00	22
na	0.95	0.95	0.95	20
nga	1.00	0.95	0.98	21
nya	0.91	0.95	0.93	21
pa	0.85	0.94	0.89	18
ra	1.00	0.96	0.98	24
sa	0.89	0.96	0.92	25
ta	1.00	0.92	0.96	25
tha	1.00	0.95	0.98	22
wa	1.00	0.86	0.92	21
ya	0.90	0.95	0.92	19
accuracy			0.95	441
macro avg	0.95	0.95	0.95	441
weighted avg	0.95	0.95	0.95	441

Gambar 4.12 Hasil Classification Report SVM-SIFT Skenario 14

Dari gambar diatas dapat dilihat bahwa nilai *F1-Score* rata-rata sudah diatas 90%, hal ini membuktikan bahwa keseluruhan data secara merata dapat terklasifikasi dengan sangat baik, tidak ada yang buruk di satu sisi. Kemudian nilai akurasi model pada dataset uji yang diperoleh keseluruhan mencapai 94.55%. Maka dari itu dapat dikatakan bahwa model SVM-SIFT dengan parameter skenario 14 sudah bekerja dengan sangat baik dan memiliki nilai akurasi yang valid dibuktikan dengan hasil *classification report* yang ada.

Selain itu pengujian tidak hanya dilakukan pada model SVM-SIFT, pengujian juga dilakukan pada model SVM tanpa bantuan metode SIFT dengan dataset yang sama. Pengujian ini dilakukan untuk membandingkan bagaimana perbedaan performa antara kedua model tersebut. Namun perlakuan terhadap dataset tidak dapat menerapkan hal yang sama persis dengan model SVM-SIFT, dataset yang ada dilakukan *preprocessing resizing* yang berbeda hal ini disebabkan karena kebutuhan input model SVM tidak dapat menerima fitur yang sangat besar sehingga ukuran data gambar perlu diperkecil. *Preprocessing* pada *resizing* dilakukan dengan mengubah ukuran data gambar menjadi 32x32 piksel agar fitur yang dapat diterima oleh model sebesar 1024 fitur.

Pada pengujian model SVM tanpa SIFT ini juga melakukan optimasi *hyperparameter tuning* terhadap beberapa parameter dari model SVM yaitu nilai C dan nilai G. Kombinasi parameter tersebut yaitu sebagai berikut untuk nilai C = [1, 3, 6, 10, 15] dan untuk nilai G = [0.02265, 0.02312, 0.02359, 0.02406, 0.02453]. Berikut ini adalah hasil pelatihan optimasi model menggunakan GridSearchCV pada model SVM tanpa SIFT.

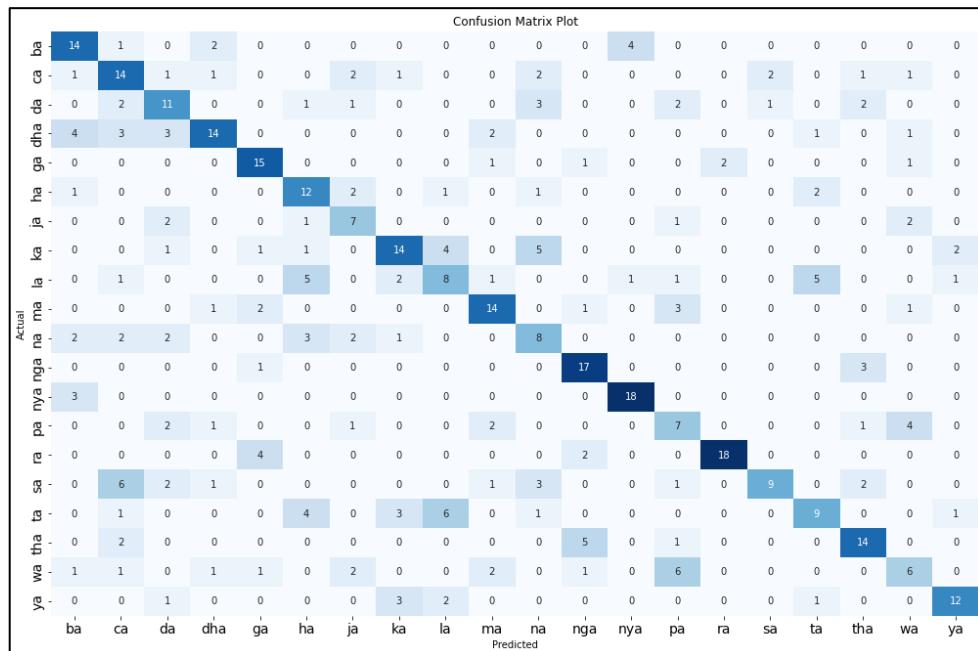
**Tabel 4.5 Hasil Pelatihan SVM tanpa SIFT**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.02312	{'C': 3, 'gamma': 0.02312}	1	0.511807	0.011146
3_0.02265	{'C': 3, 'gamma': 0.02265}	2	0.511407	0.011168
3_0.02406	{'C': 3, 'gamma': 0.02406}	3	0.510608	0.010277
3_0.02359	{'C': 3, 'gamma': 0.02359}	4	0.510207	0.010708
15_0.02265	{'C': 15, 'gamma': 0.02265}	5	0.509408	0.013446
10_0.02265	{'C': 10, 'gamma': 0.02265}	5	0.509408	0.013446
6_0.02265	{'C': 6, 'gamma': 0.02265}	5	0.509408	0.012712
6_0.02312	{'C': 6, 'gamma': 0.02312}	8	0.509407	0.013052
3_0.02453	{'C': 3, 'gamma': 0.02453}	9	0.509007	0.010085
10_0.02312	{'C': 10, 'gamma': 0.02312}	9	0.509007	0.012622
15_0.02312	{'C': 15, 'gamma': 0.02312}	9	0.509007	0.012622
6_0.02406	{'C': 6, 'gamma': 0.02406}	12	0.507407	0.014696
10_0.02406	{'C': 10, 'gamma': 0.02406}	13	0.507406	0.013294
10_0.02359	{'C': 10, 'gamma': 0.02359}	13	0.507406	0.012423
15_0.02359	{'C': 15, 'gamma': 0.02359}	13	0.507406	0.012423
15_0.02406	{'C': 15, 'gamma': 0.02406}	13	0.507406	0.013294

**Tabel 4.6 Hasil Pelatihan SVM tanpa SIFT Lanjutan**

kernel	params	rank_test_score	mean_test_score	std_test_score
6_0.02359	{'C': 6, 'gamma': 0.02359}	17	0.507006	0.013539
6_0.02453	{'C': 6, 'gamma': 0.02453}	18	0.507004	0.014284
10_0.02453	{'C': 10, 'gamma': 0.02453}	19	0.506604	0.014053
15_0.02453	{'C': 15, 'gamma': 0.02453}	19	0.506604	0.014053
1_0.02453	{'C': 1, 'gamma': 0.02453}	21	0.449780	0.006010
1_0.02312	{'C': 1, 'gamma': 0.02312}	22	0.447782	0.010193
1_0.02406	{'C': 1, 'gamma': 0.02406}	23	0.447780	0.006575
1_0.02359	{'C': 1, 'gamma': 0.02359}	24	0.447381	0.007565
1_0.02265	{'C': 1, 'gamma': 0.02265}	25	0.445781	0.009792

Dari tabel diatas menunjukkan bahwa parameter yang paling optimal pada model SVM tanpa SIFT terdapat pada nilai C = 3 dan nilai G = 0.02312 dengan akurasi pada dataset latih mencapai 51.18%. Kemudian pengujian juga dilakukan seperti sebelumnya yaitu mencoba prediksi seluruh gambar pada dataset uji dan menampilkan hasilnya dalam bentuk *confusion matrix*. Berikut adalah gambar hasil *confusion matrix* tersebut.



**Gambar 4.13 Hasil Confusion Matrix SVM tanpa SIFT**

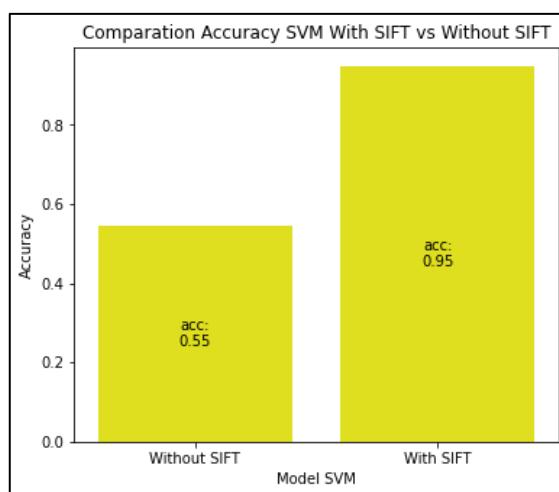
Pada hasil gambar *confusion matrix* diatas menunjukkan bahwa masih sangat banyak terjadi kesalahan prediksi, bahkan terdapat label yang terklasifikasi salah lebih banyak dari pada terklasifikasi benar, contohnya pada label ‘na’ yang hanya memiliki 8 data terklasifikasi benar dan 12 sisanya terklasifikasi salah. Terakhir model SVM tanpa SIFT diuji dengan menghitung nilai *classification report* (*precision*, *recall*, dan *F1-Score* pada setiap labelnya), serta menghitung total akurasi pada dataset uji. Berikut adalah hasil perhitungannya.

Classification report result:				
	precision	recall	f1-score	support
ba	0.54	0.67	0.60	21
ca	0.42	0.54	0.47	26
da	0.44	0.48	0.46	23
dha	0.67	0.50	0.57	28
ga	0.62	0.75	0.68	20
ha	0.44	0.63	0.52	19
ja	0.41	0.54	0.47	13
ka	0.58	0.50	0.54	28
la	0.38	0.32	0.35	25
ma	0.61	0.64	0.62	22
na	0.35	0.40	0.37	20
nga	0.63	0.81	0.71	21
nya	0.78	0.86	0.82	21
pa	0.32	0.39	0.35	18
ra	0.90	0.75	0.82	24
sa	0.75	0.36	0.49	25
ta	0.50	0.36	0.42	25
tha	0.61	0.64	0.62	22
wa	0.38	0.29	0.32	21
ya	0.75	0.63	0.69	19
accuracy			0.55	441
macro avg	0.55	0.55	0.54	441
weighted avg	0.56	0.55	0.54	441

Gambar 4.14 Hasil Classification Report SVM tanpa SIFT

Hasil *confusion matrix* diatas menyimpulkan bahwa performa model sangat rendah dan tidak stabil karena beberapa label memiliki nilai F1-Score yang sangat rendah rata-rata sekitar 50-60%. Dan pada label ‘la’ dan ‘pa’ memiliki performa terendah yaitu hanya 35%. Kemudian nilai akurasi terhadap model SVM tanpa SIFT pada dataset uji yang diperoleh keseluruhan hanya mencapai 54.64%.

Dari hasil pengujian model yang telah dilakukan dapat disimpulkan bahwa pada model SVM-SIFT dan model SVM tanpa SIFT terlihat memiliki perbedaan performa akurasi yang cukup besar yang mana model pada SVM-SIFT dapat mencapai akurasi sebesar 94.55% sedangkan pada model SVM tanpa SIFT mencapai akurasi sebesar 54.64% pada dataset uji. Berikut ini adalah diagram perbandingan akurasi dari kedua model tersebut.



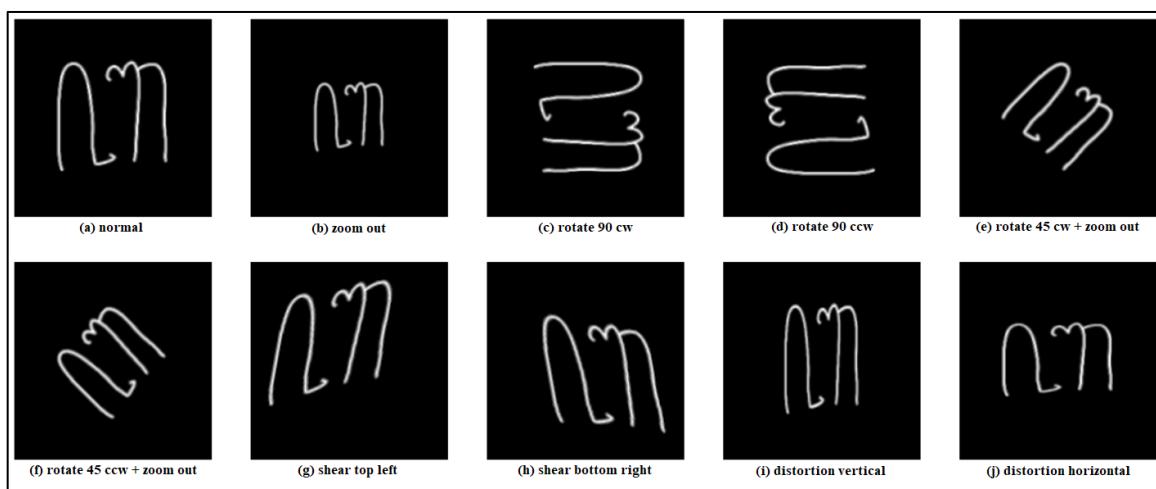
Gambar 4.15 Perbandingan Akurasi SVM-SIFT dan SVM tanpa SIFT

#### 4.2.2. Pengujian Sampel Data

Pengujian lainnya juga dilakukan dengan pendekatan yang berbeda dimana pada bagian ini pengujian akan dilakukan terhadap beberapa sampel data yang baru dengan variasi-variasi yang berbeda. Pada bagian ini model SVM-SIFT akan diuji bagaimana performa model terhadap data baru dengan variasi yang cukup ekstrim. Pengujian tersebut bertujuan untuk menunjukkan kemampuan metode SIFT dalam hal pengambilan fitur terhadap gambar yang akan diklasifikasi, karena pada dasarnya metode SIFT memiliki kemampuan yang cukup bagus terhadap data yang memiliki skala yang tidak beraturan seperti contohnya memiliki rotasi dan jarak yang acak. Sehingga beberapa sampel data yang diuji pada bagian ini memiliki variasi skala yang berbeda-beda. Setiap label kelas pada karakter hanacaraka aksara jawa akan diambil satu sampel dengan 10 variasi skala yang berbeda yaitu seperti berikut.

- a. Gambar normal
- b. Pengecilan gambar
- c. Rotasi 90 derajat searah jarum jam
- d. Rotasi 90 derajat berlawanan arah jarum jam
- e. Rotasi 45 derajat searah jarum jam dan pengecilan gambar
- f. Rotasi 45 derajat berlawanan arah jarum jam dan pengecilan gambar
- g. Shear gambar kearah atas kiri
- h. Shear gambar kearah bawah bawah
- i. Distorsi secara vertikal
- j. Distorsi secara horizontal

Berikut ini merupakan salah satu contoh 10 variasi data gambar yang divisualisasikan pada karakter label ‘ba’.



Gambar 4.16 Contoh Variasi Sampel Data Karakter ‘Ba’

Sehingga dari variasi data tersebut terdapat 200 sampel data dimana dari 20 kelas label yang ada terdapat masing-masing 10 variasi yang berbeda. Kemudian 200 data tersebut dilakukan prediksi klasifikasi dari model SVM-SIFT terbaik. Berikut ini adalah hasil klasifikasi yang dilakukan.

Classification report result:					
	precision	recall	f1-score	support	
ba	0.90	0.90	0.90	10	
ca	1.00	0.90	0.95	10	
da	0.91	1.00	0.95	10	
dha	0.90	0.90	0.90	10	
ga	0.91	1.00	0.95	10	
ha	0.75	0.90	0.82	10	
ja	1.00	0.90	0.95	10	
ka	1.00	1.00	1.00	10	
la	0.88	0.70	0.78	10	
ma	0.91	1.00	0.95	10	
na	1.00	0.90	0.95	10	
nga	1.00	1.00	1.00	10	
nya	0.90	0.90	0.90	10	
pa	0.90	0.90	0.90	10	
ra	1.00	1.00	1.00	10	
sa	0.91	1.00	0.95	10	
ta	1.00	1.00	1.00	10	
tha	1.00	1.00	1.00	10	
wa	0.90	0.90	0.90	10	
ya	1.00	0.90	0.95	10	
accuracy				0.94	200
macro avg				0.94	200
weighted avg				0.94	200

Gambar 4.17 Hasil Classification Report SVM-SIFT pada Sampel Data

Pada gambar diatas dapat dilihat bahwa hasil klasifikasi yang dilakukan memiliki performa yang cukup baik dimana rata-rata nilai *F1-Score* pada setiap label dapat mencapai diatas 90% dan nilai akurasi dari keseluruhan data dapat mencapai 93.5%. Namun terdapat satu kelas label yang memberikan hasil *F1-Score* kurang baik yaitu pada label ‘la’ hanya mencapai 78%. Kemudian hasil klasifikasi pada setiap variasi data tersebut dapat dilihat pada tabel berikut ini.

Tabel 4.7 Detail Hasil Klasifikasi Sampel Data

label	var_a	var_b	var_c	var_d	var_e	var_f	var_g	var_h	var_i	var_j	total
ba	ha	9									
ca	sa	9									
da	10										
dha	ma	9									
ga	10										
ha	la	ha	ha	9							
ja	pa	ja	ja	ja	9						

**Tabel 4.8 Detail Hasil Klasifikasi Sampel Data Lanjutan**

label	var_a	var_b	var_c	var_d	var_e	var_f	var_g	var_h	var_i	var_j	total
<b>ka</b>	ka	<b>10</b>									
<b>la</b>	la	la	la	la	la	la	ha	la	ga	ha	<b>7</b>
<b>ma</b>	ma	<b>10</b>									
<b>na</b>	na	da	na	<b>9</b>							
<b>nga</b>	nga	<b>10</b>									
<b>nya</b>	nya	ba	nya	<b>9</b>							
<b>pa</b>	pa	wa	<b>9</b>								
<b>ra</b>	ra	<b>10</b>									
<b>sa</b>	sa	<b>10</b>									
<b>ta</b>	ta	<b>10</b>									
<b>tha</b>	tha	<b>10</b>									
<b>wa</b>	wa	wa	wa	wa	wa	wa	dha	wa	wa	wa	<b>9</b>
<b>ya</b>	ya	nya	<b>9</b>								
<b>total</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>20</b>	<b>17</b>	<b>19</b>	<b>17</b>	<b>14</b>	<b>187</b>

Dari hasil klasifikasi seperti pada tabel diatas dapat dilihat bahwa dari 200 sampel data yang tersedia terdapat 187 diantaranya berhasil diklasifikasikan dengan benar dan 13 data lainnya diklasifikasikan dengan salah. Kemudian pada kolom pertama terdapat 10 variasi data yang diurutkan dari variasi (a) hingga variasi (j) direferensikan dari daftar 10 variasi yang sudah dijelaskan pada paragraf awal sebelumnya. Dari hasil klasifikasi tersebut, variasi (a) hingga (f) dapat diklasifikasikan dengan sempurna tanpa ada kesalahan, selanjutnya pada variasi (g) dan (h) hanya beberapa kesalahan yang terjadi, terakhir pada variasi (i) dan (j) memiliki tingkat kesalahan yang lebih besar dari yang lainnya. Variasi (j) memiliki tingkat kesalahan klasifikasi yang paling besar sebanyak 6 kesalahan dari 20 data, variasi ini memiliki tingkat akurasi mencapai 70%.

Data hasil klasifikasi tersebut dapat membuktikan bahwa ekstraksi fitur pada gambar menggunakan metode SIFT dapat dilakukan dengan baik meskipun terdapat gambar yang memiliki skala dan rotasi yang cukup ekstrim (variasi a-f), kemudian kemampuan metode ini perlahan melemah ketika data gambar divariasikan selain skala dan rotasi yaitu contohnya pada *shear* dan distorsi gambar (variasi g-j). Sehingga dapat disimpulkan bahwa kurangnya kemampuan ekstraksi fitur pada metode SIFT ini terjadi karena adanya perubahan bentuk yang terjadi pada gambar.

#### 4.3. Pembahasan

Pada bagian pembahasan menjelaskan makna dan hasil yang diperoleh untuk menjawab masalah dari penelitian ini. Masalah pada penelitian ini terkait dengan bagaimana penerapan model SVM dan bantuan ekstraksi fitur SIFT dalam melakukan klasifikasi pada karakter hanacaraka aksara jawa, kemudian penelitian juga mengidentifikasi bagaimana performa akurasi metode yang digunakan. Proses tahapan penelitian telah dilakukan secara sistematis dari awal hingga akhir.

Penelitian dilakukan dengan menggunakan dataset dari website *Kaggle* oleh akun Phiard yang telah dilakukan augmentasi tujuh varian berupa rotasi, *zoom*, dan *shear* selain

itu juga dilakukan *image preprocessing* berupa *resizing*, *grayscale*, dan *equalization histogram*. Sehingga data yang diperoleh untuk digunakan sebanyak 2940 data yang berisi 20 kelas dengan ukuran gambar 192x192 piksel. Kemudian dari data tersebut dilakukan ekstraksi fitur menggunakan metode SIFT dengan metode tambahan BoW sebagai tempat penampungan fitur penting dalam SIFT dengan ukuran kamus sebesar 750, penerapan BoW ini dilakukan dengan metode K-Means dengan jumlah *centroids* sebanyak 750, hal ini perlu dilakukan karena hasil ekstraksi fitur SIFT mengeluarkan jumlah fitur yang dinamis dan tidak dapat diterapkan secara langsung kepada kebutuhan model. Selanjutnya data fitur dilakukan normalisasi menggunakan *MinMaxScaling* dan data tersebut dipecah menjadi data latih dan data uji dengan rasio 85:15 sehingga menghasilkan data dengan jumlah data latih sebesar 2499 dan data uji sebesar 441. Tahapan berikutnya pelatihan model menggunakan metode SVM dengan parameter nilai  $C = 6$  dan nilai  $G = 0.10235$ . Hasil pelatihan model mendapatkan performa akurasi yang tinggi pada data latih yaitu mencapai 92.11%. Terakhir pengujian model dilakukan dengan melakukan prediksi pada data uji dan mendapatkan performa akurasi yang tinggi mencapai 94.55%. Terakhir hasil pelatihan model tersebut diluncurkan ke dalam perangkat lunak berbasis *website*. Pengembangan aplikasi ini menggunakan *library* Flask, kemudian disimpan ke dalam *platform* Github dan di-publish ke dalam server *platform* Heroku.

Seluruh tahapan tersebut tentunya penelitian melakukan berbagai macam pengujian untuk optimasi model SVM-SIFT agar mendapatkan hasil performa yang terbaik. Optimasi diterapkan dengan melakukan *hyperparameter tuning* pada beberapa kombinasi parameter *Size*, nilai  $K$ , nilai  $C$ , dan nilai  $G$  yang telah dijelaskan sebelumnya pada Tabel 4.1. Optimasi tersebut memperoleh hasil terbaik dengan nilai performa akurasi yang tinggi pada skenario 14 dengan parameter  $Size = 192$ ,  $K = 750$ ,  $C = 6$ , dan  $G = 0.10235$ , yaitu memperoleh akurasi terhadap data latih mencapai 92.11% dan data uji mencapai 94.55%. Pengujian juga dilakukan pada model SVM tanpa SIFT dengan optimasi parameter yang dilakukan pada nilai  $C$  dan nilai  $G$ . Model SVM tanpa SIFT memperoleh performa akurasi yang tidak cukup baik yaitu mencapai 51.18% pada data latih dan 54.64% pada data uji dengan parameter nilai  $C = 3$  dan nilai  $G = 0.02312$ . Perbedaan performa akurasi terhadap model SVM-SIFT dan SVM tanpa SIFT memiliki selisih yang sangat besar, perbedaan tersebut membuktikan bahwa penggunaan metode ekstraksi fitur SIFT terhadap model SVM memiliki pengaruh yang besar terhadap peningkatan akurasi klasifikasi. Tidak hanya membantu meningkatkan akurasi, model SVM-SIFT ini juga berhasil diterapkan dengan performa akurasi yang diperoleh sangat baik diatas 90% pada dataset yang digunakan penelitian ini.

Hasil performa pengujian akurasi model pada penelitian ini dibandingkan dengan penelitian lainnya yang telah dilakukan pada pengenalan tulisan tangan hanacaraka aksara jawa. Perbandingan dilakukan dengan mengambil metode dan ekstraksi fitur yang digunakan terhadap objek penelitian yang sama. Terdapat sepuluh penelitian yang dibandingkan dan dapat dilihat pada Tabel 4.9.

**Tabel 4.9 Perbandingan Akurasi dengan Penelitian Lain**

No	Penulis	Metode	Ekstraksi Fitur	Akurasi
1	Rismiyati et al. (2017)	CNN	-	70.22%
2	Rismiyati et al. (2018)	SVM	<i>Zone Based</i>	81.98%
3	Widiarti & Wastu (2009)	HMM	<i>Horizontal &amp; Vertical Vector</i>	85.7%
4	Sari et al. (2018)	KNN	<i>Roundness &amp; Eccentricity</i>	87.5%
5	Rismiyati et al. (2018)	SVM	HOG	88.45%
6	Dewa et al. (2018)	CNN	-	89%
7	Yulianti et al. (2019)	SVM	<i>Moment Invariant</i>	92.57%
8	Wibowo et al. (2018)	CNN	-	94.57%
9	Rasyidi et al. (2021)	<i>Random Forest</i>	-	97.77%
10	Susanto et al. (2021)	KNN	HOG	98.5%
11	Penelitian ini	SVM	SIFT	94.55%

Terlepas dari variasi dan jumlah *dataset* yang digunakan oleh penelitian sebelumnya, hasil performa akurasi tertinggi mencapai 98.5% yaitu pada penelitian dari Susanto et al. (2021) dengan menggunakan metode KNN dan ekstraksi fitur HOG. Sedangkan hasil performa akurasi terendah mencapai 70.22% yaitu pada penelitian dari Rismiyati et al. (2017) dengan menggunakan metode CNN tanpa tambahan ekstraksi fitur yang dilakukan. Pada penelitian ini memiliki performa yang hampir sama dengan penelitian yang dilakukan oleh Wibowo et al. (2018) dimana penelitian tersebut menggunakan metode CNN tanpa tambahan ekstraksi fitur dan mencapai tingkat akurasi 94.57%. Berdasarkan Tabel 4.9. performa akurasi pada penelitian ini dapat menyaingi penelitian-penelitian sebelumnya dengan urutan keempat dari akurasi tertinggi dan selisih akurasi dari urutan ketiga hanya sebesar 0.02%.

Selain itu, pengujian pada penelitian ini juga dilakukan dengan pendekatan yang berbeda dimana pengujian dilakukan menggunakan sampel data dengan 10 varian yang cukup ekstrim. Varian data yang digunakan terkait dengan kombinasi dari pengecilan gambar, rotasi, *shear*, dan distorsi gambar. Detail varian tersebut dapat dilihat pada penjelasan bagian 4.2.2. Pengujian ini bertujuan untuk menunjukkan kemampuan ekstraksi fitur metode SIFT. Hasil dari pengujian tersebut menunjukkan bahwa sampel data dengan varian rotasi dan skala yang ekstrim dapat diklasifikasikan dengan sempurna, namun performa metode SIFT perlahan menurun ketika sampel data dengan varian *shear* dan distorsi dengan jumlah kesalahan 1-3 pada *shear* dan 3-6 pada distorsi dari 20 total sampel data varian. Penurunan performa tersebut terjadi karena adanya perubahan bentuk dari gambar asli yang pernah diuji. Dari hasil tersebut menunjukkan bahwa performa dari metode SIFT sangat baik dalam pengambilan fitur dari gambar yang divariasikan dengan rotasi dan skala yang cukup ekstrim. Sedangkan performa metode SIFT perlahan menurun dalam pengambilan fitur dari gambar apabila menemukan gambar dengan variasi yang baru selain rotasi dan skala.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1. Kesimpulan**

Berdasarkan hasil penelitian yang dilakukan, maka kesimpulan yang dapat diperoleh yaitu sebagai berikut:

- a. Pengenalan 20 karakter hanacaraka aksara jawa menggunakan metode SVM dan SIFT sebagai ekstraksi fitur berhasil diterapkan dengan jumlah data 2940 yang telah dilakukan augmentasi (rotasi, pengecilan gambar, *shear*) dan telah dilakukan *image preprocessing* (*resize*, *grayscale*, *equalization histogram*).
- b. Metode SIFT berhasil bekerja dengan melakukan ekstraksi fitur dari data gambar, kemudian fitur keypoint yang penting pada gambar dipilih dan dimasukkan ke dalam BoW dengan menggunakan metode K-Means. BoW tersebut digunakan untuk membuat fitur akhir dari SIFT agar menyesuaikan ukuran yang dibutuhkan oleh model.
- c. Metode SVM berhasil bekerja menggunakan kernel RBF. Data yang dimasukkan ke dalam model dipecah menjadi data latih dan data uji dengan rasio 85:15% sehingga jumlah data latih sebesar 2499 dan data uji sebesar 441.
- d. Evaluasi model pada penelitian ini dilakukan dengan mengidentifikasi performa akurasi dari model SVM-SIFT yang telah dibangun. Dalam prosesnya, dalam meningkatkan akurasi dilakukan *parameter tuning* dari beberapa parameter yaitu ukuran gambar, nilai K pada K-Means, nilai C pada SVM, dan nilai G pada RBF. Terdapat 20 skenario yang dilakukan dalam *parameter tuning*, hasil performa terbaik diperoleh pada skenario 14 dimana ukuran gambar = 192, K = 750, C = 6, dan G = 0.10235. Skenario tersebut memperoleh tingkat akurasi yang tinggi yaitu 92.11% pada data latih dan 94.55% pada data uji. Performa akurasi yang diperoleh pada model SVM-SIFT sangat tinggi dibandingkan dengan performa akurasi yang diperoleh pada model SVM tanpa SIFT yaitu hanya dapat mencapai akurasi 54.64% pada data uji.
- e. Pengujian lainnya dengan pendekatan yang berbeda pada model SVM-SIFT berfokus pada performa metode SIFT dilakukan dengan sampel data berupa 10 varian cukup ekstrim dari rotasi, skala, *shear*, dan distorsi pada setiap label. Pengujian ini memberikan hasil bahwa kemampuan ekstraksi fitur SIFT sangat baik tanpa ada kesalahan pada varian rotasi dan skala, sedangkan metode SIFT melemah ketika dihadapi dengan data yang memiliki perubahan bentuk dari gambar yang pernah ditemui seperti pada varian *shear* dan distorsi.

## **5.2. Saran**

Dari kesimpulan yang telah dijabarkan dalam penelitian ini, terdapat beberapa hal yang bisa dikembangkan lebih lanjut yaitu:

- a. Pada proses tahapan *image preprocessing* dapat menggunakan metode lainnya untuk dapat meningkatkan kualitas dari data gambar.
- b. Dalam penelitian lebih lanjut dapat melakukan segmentasi huruf hanacaraka dari serangkaian kata ataupun kalimat.
- c. Dalam hal evaluasi penelitian juga dapat diperluas lebih lanjut dengan memperhatikan bagaimana performa selain akurasi seperti kompleksitas waktu maupun komputasi.

## DAFTAR PUSTAKA

- Ali A., Suresha, M., & Ahmed, M. (2019). Different Handwritten Character Recognition Methods: A Review. *2019 Global Conference for Advancement in Technology, GCAT 2019* (pp. 1-8). Bangalore: IEEE.
- Ambarwari, A., Adrian, Q. J., & Herdiyeni, Y. (2020). Analisis Pengaruh Data Scaling Terhadap Performa Algoritme Machine Learning untuk Identifikasi Tanaman. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 4(1), 117–122.
- Andini, E., Faisal, M., R., Herteno, R., Nugroho, R., A., Abadi, F., & Muliadi. (2022). Peningkatan Kinerja Prediksi Cacat Software Dengan Hyperparameter Tuning Pada Algoritma Klasifikasi Deep Forest. *Jurnal MNEMONIC*, 5(2), 119-127.
- Chen, X., Jin, Z., Wang, Q., Yang, W., Liao, Q., & Meng, H. (2021). Unsupervised visual feature learning based on similarity guidance. *Neurocomputing*.
- Dewa, C. K., Fadhilah, A. L., & Afiahayati, A. (2018). Convolutional Neural Networks for Handwritten Javanese Character Recognition. *Indonesian Journal of Computing and Cybernetics Systems*, 12(1), 83-94.
- Doush, I. A., & AL-Btoush, S. (2017). Currency recognition using a smartphone: Comparison between color SIFT and gray scale SIFT algorithms. *Journal of King Saud University – Computer and Information Sciences*, 29(4), 484-492.
- Ghadekar, P., Ingole, S., & Sonone, D. (2018). Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM Classifier. *Fourth International Conference on Computing Communication Control and Automation* (pp. 1-6). Pune: IEEE.
- Gupta, S., Thakur, K., & Kumar, M. (2020). 2D-human face recognition using SIFT and SURF descriptors of face's feature regions. *The Visual Computer*, 37, 447–456.
- Hadiprakoso, R., B., & Qomariasih, N. (2022). Deteksi Masker Wajah Menggunakan Deep Transfer Learning Dan Augmentasi Gambar. *Jurnal Informatika dan Komputer*, 5(1), 12-18.
- Harahap, E. H., Muflikhah, L., & Rahayudi, B. (2018). Implementasi Algoritma Support Vector Machine (SVM) Untuk Penentuan Seleksi Atlet Pencak Silat. 2(10), 3843–3848.
- Hasanah, M., A., Soim, S., & Handayani, A., S. (2021). Implementasi CRISP-DM Model Menggunakan Metode Decision Tree dengan Algoritma CART untuk Prediksi Curah Hujan Berpotensi Banjir. *Journal of Applied Informatics and Computing*, 5(2), 103-108.
- Hassan, A. K. A., Mahdi, B. S., & Mohammed, A. A. (2019). Arabic Handwriting Word Recognition Based on Scale Invariant Feature Transform and Support Vector Machine. *Iraqi Journal of Science*, 60(2), 381-387.
- Hidayat, N., & Hati, K. (2021). Penerapan Metode Rapid Application Development (RAD) dalam Rancang Bangun Sistem Informasi Rapor Online (SIRALINE). *Jurnal Sistem Informasi STMIK Antar Bangsa*, 10(1), 8-17.

- Hidayat, A. & Shofa, R. N. (2016). Self Organizing Maps (Som) Suatu Metode Untuk Pengenalan Aksara Jawa. *Jurnal Siliwangi Sains Teknologi*, 2(1), 64-70.
- Kamble, P. M., & Hegadi, R. S. (2017). Comparative Study of Handwritten Marathi Characters Recognition Based on KNN and SVM Classifier. *Recent Trends in Image Processing and Pattern Recognition*, 709, 93-101.
- Kusumanto, RD. & Tompu, A. N. (2011). Pengolahan Citra Digital Untuk Mendekripsi Objek Menggunakan Pengolahan Warna Model Normalisasi Rgb. *Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2011*. Palembang: Semantik.
- Liu, Y., Bi, J. W., & Fan, Z. P. (2017). Multi-class sentiment classification: The experimental comparisons of feature selection and machine learning algorithms. *Expert Systems with Applications*, 80, 323–339.
- Lorentius, C., A., Adipranata, R., & Tjondrowiguno A. (2019). Pengenalan Aksara Jawa dengan Menggunakan Metode Convolutional Neural Network. *Jurnal Infra*, 7(1).
- Lutfiani, N., Harahap, E., P., Aini, Q., Ahmad, A., D., A., R., & Rahardja, U. (2020). Inovasi Manajemen Proyek I-Learning Menggunakan Metode Agile Scrumban. *InfoTekJar: Jurnal Nasional Informatika dan Teknologi Jaringan*, 5(1), 96-101.
- Mardiana, T., & Nyoto, R. D. (2015). Kluster Bag of Word Menggunakan Weka. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, 1(1), 1–5.
- Mohaiminul, M., & Sultana, N. (2018). Comparative Study on Machine Learning Algorithms for Sentiment Classification. *International Journal of Computer Applications*, 182(21), 1– 7.
- Mortensen, E. N., Deng, H., & Shapiro, L. (2005). A SIFT Descriptor with Global Context. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego: IEEE.
- Munantri, N. Z., Sofyan, H., & Yanu F. M. (2019). Aplikasi Pengolahan Citra Digital Untuk Identifikasi Umur Pohon. *Telematika*, 16(2), 97-104.
- Narang, S. R., Jindal, M. K., Ahuja, S., & Kumar, M. (2020). On the recognition of Devanagari ancient handwritten characters using SIFT and Gabor features. *Soft Computing*, 24, 17279–17289.
- Naufal, M. F., Kusuma, S. F., Prayuska, Z. A., & Alexander, A. (2021). Comparative Analysis of Image Classification Algorithms for Face Mask Detection. *Journal of Information Systems Engineering and Business Intelligence*, 7(1), 56-66.
- Noertjahyana, A. (2004). Studi Analisis Rapid Application Development Sebagai Salah Satu Alternatif Metode Pengembangan Perangkat Lunak. *Jurnal Informatika Petra*, 3(2), 74-79.
- Oktaviana, U., N., & Azhar, Y. (2021). Klasifikasi Sampah Menggunakan Ensemble DenseNet169. *Jurnal RESTI*, 5(6), 1206-1215.
- Pradhan, A. (2012). Support Vector Machine-A Survey. *International Journal of Emerging Technology and Advanced Engineering*, 2(8), 82-85.
- Pradnyana, G. A., & Permana, A. A. J. (2018). Sistem Pembagian Kelas Kuliah Mahasiswa Dengan Metode K-Means Dan K-Nearest Neighbors Untuk Meningkatkan Kualitas Pembelajaran. *JUTI: Jurnal Ilmiah Teknologi Informasi*, 16(1), 59.

- Rajesh S., R., Beaula, A., Marikkannu, P., Sungheetha, A., & Sahana, C. (2016). Comparative study of distinctive image classification techniques. *2016 10th International Conference on Intelligent Systems and Control*. Coimbatore: IEEE.
- Rajput, G. G. & Ummapure, S. B. (2017). Script Identification from Handwritten Documents using SIFT Method. *IEEE International Conference on Power, Control, Signals and Instrumentation Engineering* (pp. 520-526). Chennai: IEEE.
- Rasyidi, M. A., Bariyah, T., Riskajaya, Y. I., & Septyani, A. D. (2021). Classification of handwritten javanese script using random forest algorithm. *Bulletin of Electrical Engineering and Informatics*, 10 (3), 1308-1315.
- Rismiyati, Khadijah, & Adi, N. (2017). Deep learning for handwritten Javanese character recognition. *2017 1st International Conference on Informatics and Computational Sciences* (pp. 59-63). Semarang: IEEE.
- Rismiyati, Khadijah, & Riyanto, D. (2018). HOG and Zone Base Features for Handwritten Javanese Character Classification. *2018 2nd International Conference on Informatics and Computational Science* (pp. 131-135). Semarang: IEEE.
- Robby, G. A., Tandra, A., Susanto, I., Harefa, J., & Chowanda, A. (2019). Implementation of Optical Character Recognition using Tesseract with the Javanese Script Target in Android Application. *Procedia Computer Science*, 157, 449-505.
- Sanjaya, J., & Ayub, M. (2020). Augmentasi DataPengenalan Citra Mobil Menggunakan Pendekatan Random Crop, Rotate, dan Mixup. *Jurnal Teknik Informatika dan Sistem Informasi*, 6(2), 311-323.
- Saputra, A. C., Sitepu, A. B., Stanley, Yohanes Sigit, P. W. P., Sarto Aji Tetuko, P. G., & Nugroho, G. C. (2019). The Classification of the Movie Genre based on Synopsis of the Indonesian Film. *Proceeding - 2019 International Conference of Artificial Intelligence and Information Technology, ICAIIT 2019*, 201–204.
- Sari, C. A., Kuncoro, M. W., Setiadi, D. R. I. M., & Rachmawanto, E. H. (2018). Roundness and eccentricity feature extraction for Javanese handwritten character recognition based on K-nearest neighbor. *2018 International Seminar on Research of Information Technology and Intelligent Systems* (pp. 5-10). Yogyakarta: IEEE.
- Sari, V., N., Yupianti, Y., & Maharani, D. (2018). Penerapan Metode K-Means Clustering Dalam Menentukan Predikat Kelulusan Mahasiswa Untuk Menganalisa Kualitas Lulusan. *Jurteksi*, 4(2), 133–140.
- Satriyo, A., (2003). Support Vector Machine. *Teori dan Aplikasinya dalam Bioinformatika*, (I), 1-11.
- Schröera, C., Kruse, F., & Gómez, J., M. (2021). A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science*, 181, 526-534.
- Setiawan, A., Prabowo, A. S., & Puspaningrum, E. Y. (2019). Handwriting Character Recognition Javanese Letters Based on Artificial Neural Network. *International Journal of Computer, Network Security and Information System*, 1(1), 39-42.
- Setiawan, A., & Sulaiman, A. M. (2015). Hancaraka: Aksara Jawa Dalam Karakter Font dan Aplikasinya Sebagai Brand Image. *Ornamen Jurnal Kriya*, 12(1), 33-47.

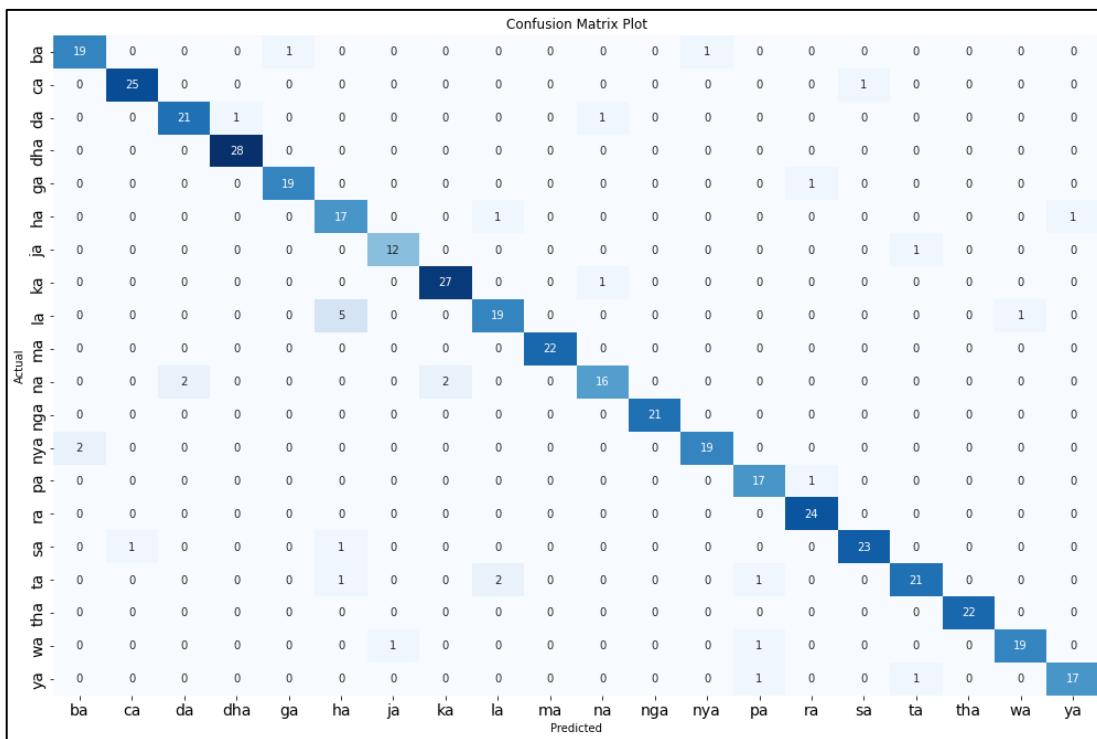
- Sriwathsan, W., Ramanan, M., & Weerasinghe, A. R. (2020). Offline Handwritten Signature Recognition Based on SIFT and SURF Features Using SVMs. *Asian Research Journal of Mathematics*, 16(1), 84-91.
- Surinta, O., Karaaba, M. F., Schomaker, L. R. B., & Wiering, M. A. (2015). Recognition of handwritten characters using local gradient feature descriptors. *Engineering Applications of Artificial Intelligence*, 45, 405-414.
- Susanto, A., Sari, C. A., Mulyono, I. U. W., & Doheir, M. (2021). Histogram of Gradient in K-Nearest Neighbor for Javanese Alphabet Classification. *Scientific Journal of Informatics*, 8(2), 289-296.
- Thamilselvana, P., & Sathiaseelan, J. G. R. (2015). A Comparative Study of Data Mining Algorithms for Image Classification. *International Journal of Education and Management Engineering*, 5(2), 1-9.
- Trisari, W., Putri, H., Hendrowati, R., & Belakang, L. (2020). *Penggalian Teks Dengan Model Bag of Words Terhadap*. 2(1), 129–138.
- Vhallah, I., Sumijan, S., & Santony, J. (2018). Pengelompokan Mahasiswa Potensial Drop Out Menggunakan Metode Clustering K-Means. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 2(2), 572–577.
- Wang, Y., Chen, J., Hu, B., Yang, X., & Ban, X. (2015). License plate recognition based on SIFT feature. *Optik - International Journal for Light and Electron Optics*, 126(21), 2895-2901.
- Wang, Y., Li, Z., Wang, L., & Wang, M. (2013). A Scale Invariant Feature Transform Based Method. *Journal of Information Hiding and Multimedia Signal Processing*, 4(2), 73-89.
- Wibowo, M. A., Soleh, M., Pradani, W., Hidayanto, A. N., & Arymurthy, A. M. (2017). Handwritten Javanese Character Recognition using Descriptive Deep Learning Technique. *2017 2nd International Conferences on Information Technology, Information Systems and Electrical Engineering* (pp. 325-330). Yogyakarta: IEEE.
- Widiarti, A. R., & Wastu, P. N. (2009). Javanese Character Recognition Using Hidden Markov Model. *International Journal of Computer and Information Engineering*, 3(9), 2201-2204.
- Yulianti, R., Wijaya, I. G. P. S., & Bimantoro, F. (2019). Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode Moment Invariant dan Support Vector Machine. *Journal of Computer Science and Informatics Engineering*, 3(2), 91-98.
- Yunita, F. (2018). Penerapan Data Mining Menggunakan Algoritma K-Means Clustering Pada Penerimaan Mahasiswa Baru. *Sistemasi*, 7(3), 238.

# **LAMPIRAN**

## LAMPIRAN A. Pengujian Parameter Skenario 1

**Tabel 1. Hasil Pelatihan SVM-SIFT Skenario 1**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.24574	{'C': 3, 'gamma': 0.24574}	1	0.893145	0.017379
3_0.25086	{'C': 3, 'gamma': 0.25086}	2	0.892345	0.016273
3_0.26622	{'C': 3, 'gamma': 0.26622}	3	0.891544	0.016953
3_0.25598	{'C': 3, 'gamma': 0.25598}	4	0.891145	0.015627
6_0.24574	{'C': 6, 'gamma': 0.24574}	5	0.890745	0.015871
6_0.26622	{'C': 6, 'gamma': 0.26622}	6	0.890745	0.016666
3_0.2611	{'C': 3, 'gamma': 0.2611}	6	0.890745	0.015523
15_0.26622	{'C': 15, 'gamma': 0.26622}	8	0.890345	0.016364
10_0.26622	{'C': 10, 'gamma': 0.26622}	8	0.890345	0.016364
15_0.24574	{'C': 15, 'gamma': 0.24574}	10	0.889945	0.015114
6_0.25086	{'C': 6, 'gamma': 0.25086}	10	0.889945	0.015635
10_0.24574	{'C': 10, 'gamma': 0.24574}	10	0.889945	0.015114
15_0.25086	{'C': 15, 'gamma': 0.25086}	13	0.889145	0.014877
15_0.2611	{'C': 15, 'gamma': 0.2611}	13	0.889145	0.014385
6_0.2611	{'C': 6, 'gamma': 0.2611}	13	0.889145	0.015196
10_0.25086	{'C': 10, 'gamma': 0.25086}	13	0.889145	0.014877
10_0.2611	{'C': 10, 'gamma': 0.2611}	13	0.889145	0.014385
10_0.25598	{'C': 10, 'gamma': 0.25598}	18	0.888745	0.014549
15_0.25598	{'C': 15, 'gamma': 0.25598}	18	0.888745	0.014549
6_0.25598	{'C': 6, 'gamma': 0.25598}	20	0.888345	0.014371
1_0.25598	{'C': 1, 'gamma': 0.25598}	21	0.883945	0.010908
1_0.25086	{'C': 1, 'gamma': 0.25086}	22	0.883545	0.010334
1_0.2611	{'C': 1, 'gamma': 0.2611}	23	0.883545	0.011655
1_0.26622	{'C': 1, 'gamma': 0.26622}	23	0.883545	0.011655
1_0.24574	{'C': 1, 'gamma': 0.24574}	25	0.882745	0.010110



Gambar 1. Hasil Confusion Matrix SVM-SIFT Skenario 1

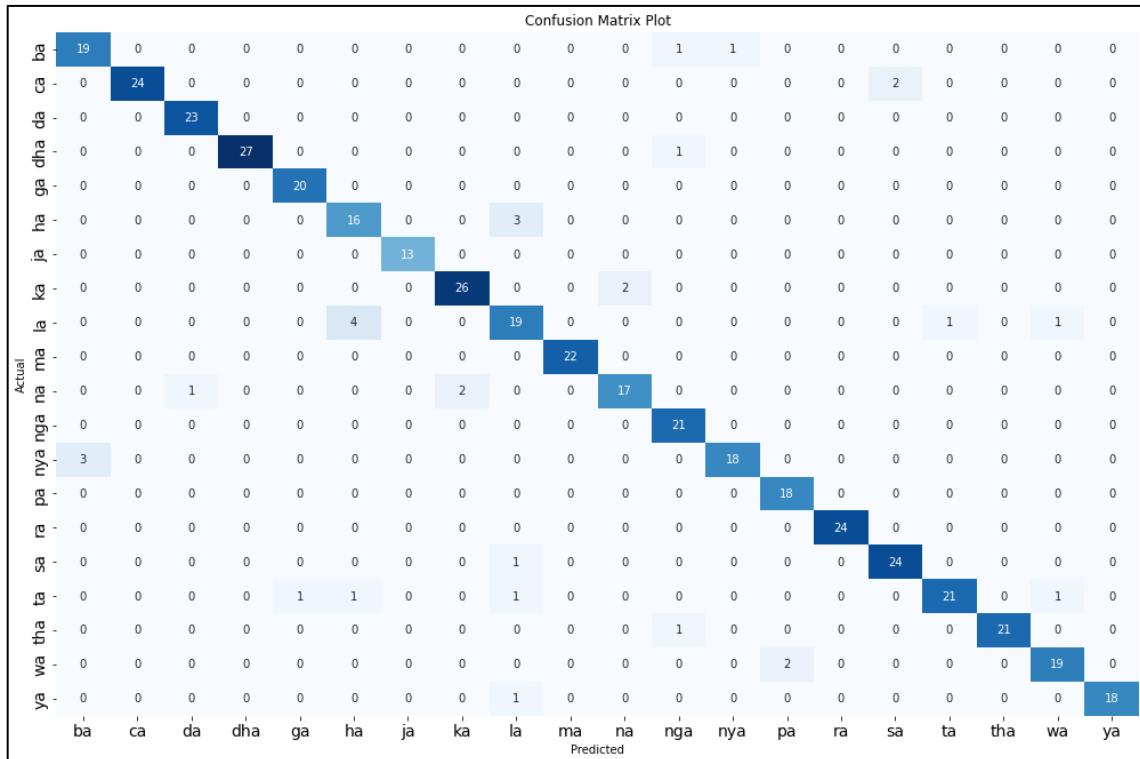
Classification report result:				
	precision	recall	f1-score	support
ba	0.90	0.90	0.90	21
ca	0.96	0.96	0.96	26
da	0.91	0.91	0.91	23
dha	0.97	1.00	0.98	28
ga	0.95	0.95	0.95	20
ha	0.71	0.89	0.79	19
ja	0.92	0.92	0.92	13
ka	0.93	0.96	0.95	28
la	0.86	0.76	0.81	25
ma	1.00	1.00	1.00	22
na	0.89	0.80	0.84	20
nga	1.00	1.00	1.00	21
nya	0.95	0.90	0.93	21
pa	0.85	0.94	0.89	18
ra	0.92	1.00	0.96	24
sa	0.96	0.92	0.94	25
ta	0.91	0.84	0.87	25
tha	1.00	1.00	1.00	22
wa	0.95	0.90	0.93	21
ya	0.94	0.89	0.92	19
accuracy				0.93
macro avg				0.92
weighted avg				0.93
				441

Gambar 2. Hasil Classification Report SVM-SIFT Skenario 1

## LAMPIRAN B. Pengujian Parameter Skenario 2

**Tabel 2. Hasil Pelatihan SVM-SIFT Skenario 2**

kernel	params	rank_test_score	mean_test_score	std_test_score
6_0.20504	{'C': 6, 'gamma': 0.20504}	1	0.906361	0.004494
15_0.20504	{'C': 15, 'gamma': 0.20504}	2	0.905961	0.004932
10_0.20504	{'C': 10, 'gamma': 0.20504}	2	0.905961	0.004932
6_0.21358	{'C': 6, 'gamma': 0.21358}	4	0.905561	0.005738
6_0.20931	{'C': 6, 'gamma': 0.20931}	4	0.905561	0.004661
15_0.21358	{'C': 15, 'gamma': 0.21358}	6	0.905161	0.006294
15_0.20931	{'C': 15, 'gamma': 0.20931}	6	0.905161	0.005179
10_0.21358	{'C': 10, 'gamma': 0.21358}	6	0.905161	0.006294
10_0.20931	{'C': 10, 'gamma': 0.20931}	6	0.905161	0.005179
6_0.21785	{'C': 6, 'gamma': 0.21785}	6	0.905161	0.006294
3_0.20504	{'C': 3, 'gamma': 0.20504}	11	0.905158	0.006822
10_0.21785	{'C': 10, 'gamma': 0.21785}	12	0.904761	0.006898
15_0.21785	{'C': 15, 'gamma': 0.21785}	12	0.904761	0.006898
10_0.22212	{'C': 10, 'gamma': 0.22212}	14	0.904361	0.006390
15_0.22212	{'C': 15, 'gamma': 0.22212}	14	0.904361	0.006390
6_0.22212	{'C': 6, 'gamma': 0.22212}	16	0.903961	0.007054
3_0.21785	{'C': 3, 'gamma': 0.21785}	17	0.903959	0.006730
3_0.21358	{'C': 3, 'gamma': 0.21358}	18	0.903958	0.006977
3_0.20931	{'C': 3, 'gamma': 0.20931}	18	0.903958	0.006122
3_0.22212	{'C': 3, 'gamma': 0.22212}	20	0.902759	0.006794
1_0.21785	{'C': 1, 'gamma': 0.21785}	21	0.897551	0.011286
1_0.21358	{'C': 1, 'gamma': 0.21358}	21	0.897551	0.011907
1_0.20931	{'C': 1, 'gamma': 0.20931}	23	0.897549	0.013078
1_0.22212	{'C': 1, 'gamma': 0.22212}	24	0.897151	0.011011
1_0.20504	{'C': 1, 'gamma': 0.20504}	25	0.896349	0.012764



Gambar 3. Hasil Confusion Matrix SVM-SIFT Skenario 2

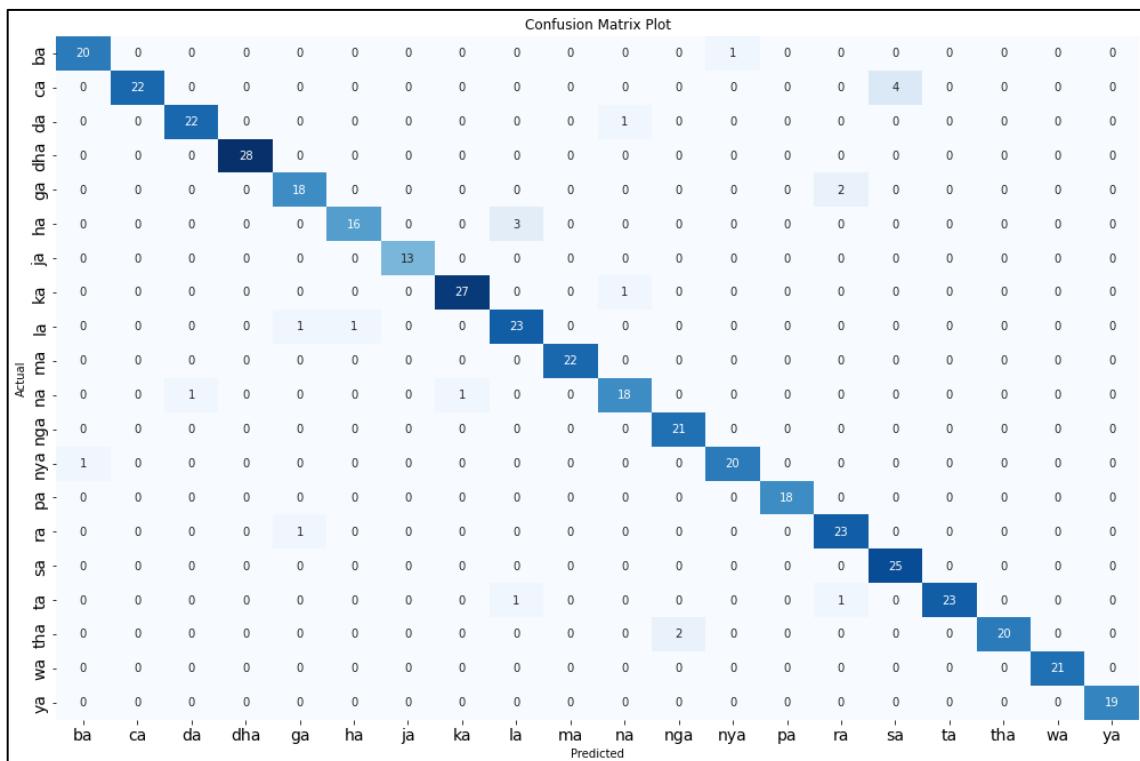
Classification report result:				
	precision	recall	f1-score	support
ba	0.86	0.90	0.88	21
ca	1.00	0.92	0.96	26
da	0.96	1.00	0.98	23
dha	1.00	0.96	0.98	28
ga	0.95	1.00	0.98	20
ha	0.76	0.84	0.80	19
ja	1.00	1.00	1.00	13
ka	0.93	0.93	0.93	28
la	0.76	0.76	0.76	25
ma	1.00	1.00	1.00	22
na	0.89	0.85	0.87	20
nga	0.88	1.00	0.93	21
nya	0.95	0.86	0.90	21
pa	0.90	1.00	0.95	18
ra	1.00	1.00	1.00	24
sa	0.92	0.96	0.94	25
ta	0.95	0.84	0.89	25
tha	1.00	0.95	0.98	22
wa	0.90	0.90	0.90	21
ya	1.00	0.95	0.97	19
accuracy				0.93
macro avg				0.93
weighted avg				0.93
				441

Gambar 4. Hasil Classification Report SVM-SIFT Skenario 2

**LAMPIRAN C. Pengujian Parameter Skenario 3**

**Tabel 3. Hasil Pelatihan SVM-SIFT Skenario 3**

kernel	params	rank_test_score	mean_test_score	std_test_score
15_0.13567	{'C': 15, 'gamma': 0.13567}	1	0.915566	0.010077
6_0.13567	{'C': 6, 'gamma': 0.13567}	1	0.915566	0.010077
10_0.13567	{'C': 10, 'gamma': 0.13567}	1	0.915566	0.010077
10_0.1385	{'C': 10, 'gamma': 0.1385}	4	0.915166	0.010327
15_0.1385	{'C': 15, 'gamma': 0.1385}	4	0.915166	0.010327
6_0.1385	{'C': 6, 'gamma': 0.1385}	4	0.915166	0.010327
10_0.14133	{'C': 10, 'gamma': 0.14133}	7	0.914765	0.010410
6_0.14133	{'C': 6, 'gamma': 0.14133}	7	0.914765	0.010410
15_0.14133	{'C': 15, 'gamma': 0.14133}	7	0.914765	0.010410
3_0.1385	{'C': 3, 'gamma': 0.1385}	10	0.914764	0.010868
3_0.13567	{'C': 3, 'gamma': 0.13567}	11	0.914364	0.010778
10_0.14416	{'C': 10, 'gamma': 0.14416}	12	0.913965	0.011668
15_0.14416	{'C': 15, 'gamma': 0.14416}	12	0.913965	0.011668
6_0.14416	{'C': 6, 'gamma': 0.14416}	12	0.913965	0.011668
3_0.14133	{'C': 3, 'gamma': 0.14133}	15	0.913963	0.011263
3_0.14416	{'C': 3, 'gamma': 0.14416}	15	0.913963	0.012217
10_0.14699	{'C': 10, 'gamma': 0.14699}	17	0.913164	0.011781
15_0.14699	{'C': 15, 'gamma': 0.14699}	17	0.913164	0.011781
6_0.14699	{'C': 6, 'gamma': 0.14699}	17	0.913164	0.011781
3_0.14699	{'C': 3, 'gamma': 0.14699}	20	0.912762	0.012324
1_0.14699	{'C': 1, 'gamma': 0.14699}	21	0.905960	0.007409
1_0.14416	{'C': 1, 'gamma': 0.14416}	22	0.905558	0.009458
1_0.14133	{'C': 1, 'gamma': 0.14133}	22	0.905558	0.009458
1_0.1385	{'C': 1, 'gamma': 0.1385}	24	0.904758	0.009301
1_0.13567	{'C': 1, 'gamma': 0.13567}	24	0.904758	0.010045



Gambar 5. Hasil Confusion Matrix SVM-SIFT Skenario 3

Classification report result:					
	precision	recall	f1-score	support	
ba	0.95	0.95	0.95	21	
ca	1.00	0.85	0.92	26	
da	0.96	0.96	0.96	23	
dha	1.00	1.00	1.00	28	
ga	0.90	0.90	0.90	20	
ha	0.94	0.84	0.89	19	
ja	1.00	1.00	1.00	13	
ka	0.96	0.96	0.96	28	
la	0.85	0.92	0.88	25	
ma	1.00	1.00	1.00	22	
na	0.90	0.90	0.90	20	
nga	0.91	1.00	0.95	21	
nya	0.95	0.95	0.95	21	
pa	1.00	1.00	1.00	18	
ra	0.88	0.96	0.92	24	
sa	0.86	1.00	0.93	25	
ta	1.00	0.92	0.96	25	
tha	1.00	0.91	0.95	22	
wa	1.00	1.00	1.00	21	
ya	1.00	1.00	1.00	19	
accuracy				0.95	441
macro avg				0.95	0.95
weighted avg				0.95	0.95
				441	

Gambar 6. Hasil Classification Report SVM-SIFT Skenario 3

**LAMPIRAN D. Pengujian Parameter Skenario 4**

**Tabel 4. Hasil Pelatihan SVM-SIFT Skenario 4**

kernel	params	rank_test_score	mean_test_score	std_test_score
15_0.10508	{'C': 15, 'gamma': 0.10508}	1	0.919965	0.007521
6_0.10508	{'C': 6, 'gamma': 0.10508}	1	0.919965	0.007521
10_0.10508	{'C': 10, 'gamma': 0.10508}	1	0.919965	0.007521
3_0.10508	{'C': 3, 'gamma': 0.10508}	4	0.919164	0.007799
3_0.11384	{'C': 3, 'gamma': 0.11384}	5	0.918764	0.007266
15_0.11384	{'C': 15, 'gamma': 0.11384}	6	0.918364	0.007671
6_0.11384	{'C': 6, 'gamma': 0.11384}	6	0.918364	0.007671
6_0.10727	{'C': 6, 'gamma': 0.10727}	6	0.918364	0.007671
3_0.11165	{'C': 3, 'gamma': 0.11165}	6	0.918364	0.007352
10_0.11384	{'C': 10, 'gamma': 0.11384}	6	0.918364	0.007671
15_0.10727	{'C': 15, 'gamma': 0.10727}	6	0.918364	0.007671
10_0.10727	{'C': 10, 'gamma': 0.10727}	6	0.918364	0.007671
15_0.11165	{'C': 15, 'gamma': 0.11165}	13	0.917964	0.007732
6_0.11165	{'C': 6, 'gamma': 0.11165}	13	0.917964	0.007732
10_0.11165	{'C': 10, 'gamma': 0.11165}	13	0.917964	0.007732
3_0.10727	{'C': 3, 'gamma': 0.10727}	16	0.917963	0.008145
15_0.10946	{'C': 15, 'gamma': 0.10946}	17	0.917164	0.006803
6_0.10946	{'C': 6, 'gamma': 0.10946}	17	0.917164	0.006803
10_0.10946	{'C': 10, 'gamma': 0.10946}	17	0.917164	0.006803
3_0.10946	{'C': 3, 'gamma': 0.10946}	20	0.917163	0.007800
1_0.10946	{'C': 1, 'gamma': 0.10946}	21	0.905959	0.004966
1_0.10508	{'C': 1, 'gamma': 0.10508}	21	0.905959	0.004966
1_0.10727	{'C': 1, 'gamma': 0.10727}	23	0.905559	0.004698
1_0.11165	{'C': 1, 'gamma': 0.11165}	24	0.905558	0.005920
1_0.11384	{'C': 1, 'gamma': 0.11384}	25	0.904758	0.005523

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	19	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0
	ca	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
Actual	da	0	0	22	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	dha	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
Actual	ga	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	ha	0	0	0	0	0	16	0	0	3	0	0	0	0	0	0	0	0	0	0	0
Actual	ja	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
	ka	0	0	0	0	0	0	0	26	0	0	0	2	0	0	0	0	0	0	0	0
Actual	la	0	0	0	0	0	1	0	0	23	0	0	0	0	0	0	0	1	0	0	0
	ma	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0
Actual	nya	0	0	1	0	0	0	1	0	0	18	0	0	0	0	0	0	0	0	0	0
	na	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0
Actual	pa	1	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
	ra	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0
Actual	sa	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0	0	0	0
	ta	0	0	0	0	0	2	0	0	0	0	0	0	1	0	0	0	22	0	0	0
Actual	tha	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	21	0	0
	wa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	18
ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya		

Gambar 7. Hasil Confusion Matrix SVM-SIFT Skenario 4

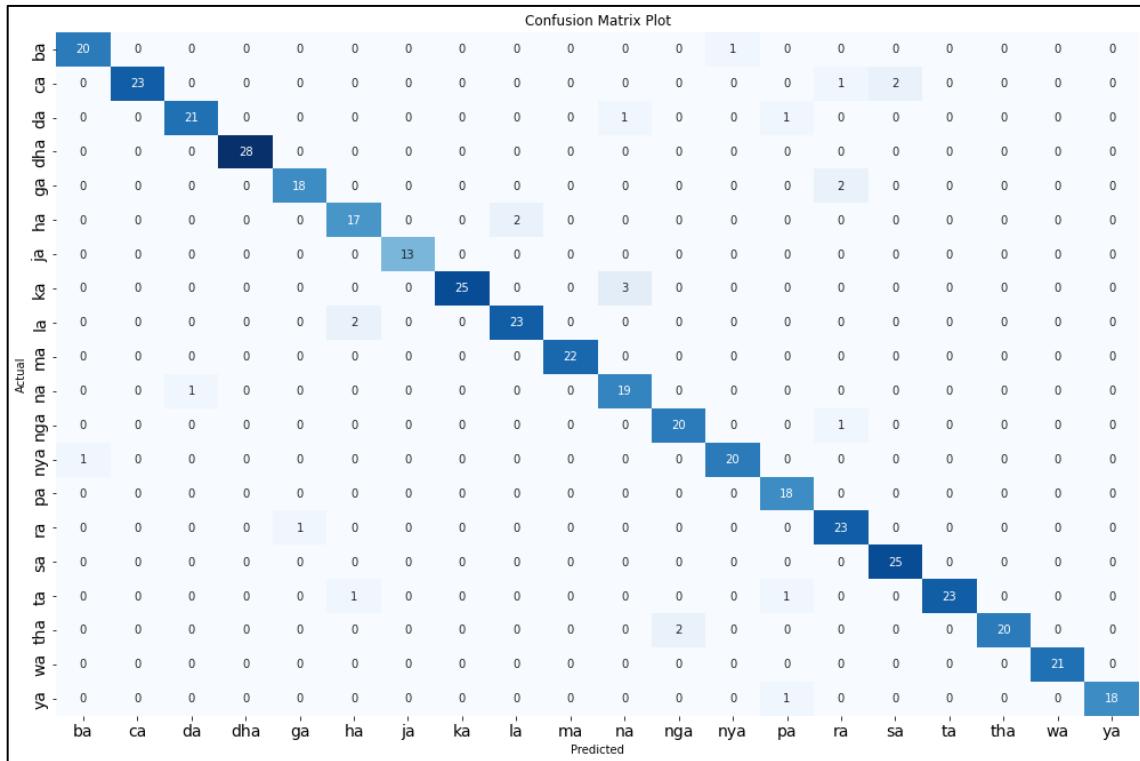
Classification report result:				
	precision	recall	f1-score	support
ba	0.95	0.90	0.93	21
ca	1.00	0.88	0.94	26
da	0.96	0.96	0.96	23
dha	1.00	0.93	0.96	28
ga	1.00	0.95	0.97	20
ha	0.84	0.84	0.84	19
ja	1.00	1.00	1.00	13
ka	0.96	0.93	0.95	28
la	0.85	0.92	0.88	25
ma	1.00	1.00	1.00	22
na	0.86	0.90	0.88	20
nga	0.91	1.00	0.95	21
nya	0.95	0.95	0.95	21
pa	0.90	1.00	0.95	18
ra	0.96	1.00	0.98	24
sa	0.89	1.00	0.94	25
ta	0.92	0.88	0.90	25
tha	1.00	0.95	0.98	22
wa	0.90	0.90	0.90	21
ya	1.00	0.95	0.97	19
accuracy				0.94
macro avg				0.94
weighted avg				0.94
				441

Gambar 8. Hasil Classification Report SVM-SIFT Skenario 4

**LAMPIRAN E. Pengujian Parameter Skenario 5**

**Tabel 5. Hasil Pelatihan SVM-SIFT Skenario 5**

kernel	params	rank_test_score	mean_test_score	std_test_score
6_0.09522	{'C': 6, 'gamma': 0.09522}	1	0.912762	0.010200
15_0.09522	{'C': 15, 'gamma': 0.09522}	1	0.912762	0.010200
15_0.09142	{'C': 15, 'gamma': 0.09142}	1	0.912762	0.010200
10_0.09522	{'C': 10, 'gamma': 0.09522}	1	0.912762	0.010200
10_0.09142	{'C': 10, 'gamma': 0.09142}	1	0.912762	0.010200
6_0.09142	{'C': 6, 'gamma': 0.09142}	1	0.912762	0.010200
15_0.09332	{'C': 15, 'gamma': 0.09332}	7	0.912361	0.010577
10_0.09332	{'C': 10, 'gamma': 0.09332}	7	0.912361	0.010577
6_0.09332	{'C': 6, 'gamma': 0.09332}	7	0.912361	0.010577
10_0.09902	{'C': 10, 'gamma': 0.09902}	10	0.911962	0.009242
10_0.09712	{'C': 10, 'gamma': 0.09712}	10	0.911962	0.010460
6_0.09902	{'C': 6, 'gamma': 0.09902}	10	0.911962	0.009242
6_0.09712	{'C': 6, 'gamma': 0.09712}	10	0.911962	0.010460
15_0.09902	{'C': 15, 'gamma': 0.09902}	10	0.911962	0.009242
15_0.09712	{'C': 15, 'gamma': 0.09712}	10	0.911962	0.010460
3_0.09142	{'C': 3, 'gamma': 0.09142}	16	0.911561	0.009455
3_0.09522	{'C': 3, 'gamma': 0.09522}	17	0.910761	0.009553
3_0.09332	{'C': 3, 'gamma': 0.09332}	18	0.910360	0.009954
3_0.09902	{'C': 3, 'gamma': 0.09902}	19	0.909961	0.008428
3_0.09712	{'C': 3, 'gamma': 0.09712}	19	0.909961	0.009749
1_0.09142	{'C': 1, 'gamma': 0.09142}	21	0.899958	0.005976
1_0.09332	{'C': 1, 'gamma': 0.09332}	22	0.899157	0.005796
1_0.09522	{'C': 1, 'gamma': 0.09522}	23	0.898357	0.005622
1_0.09902	{'C': 1, 'gamma': 0.09902}	24	0.897557	0.005893
1_0.09712	{'C': 1, 'gamma': 0.09712}	24	0.897557	0.005323



Gambar 9. Hasil Confusion Matrix SVM-SIFT Skenario 5

Classification report result:					
	precision	recall	f1-score	support	
ba	0.95	0.95	0.95	21	
ca	1.00	0.88	0.94	26	
da	0.95	0.91	0.93	23	
dha	1.00	1.00	1.00	28	
ga	0.95	0.90	0.92	20	
ha	0.85	0.89	0.87	19	
ja	1.00	1.00	1.00	13	
ka	1.00	0.89	0.94	28	
la	0.92	0.92	0.92	25	
ma	1.00	1.00	1.00	22	
na	0.83	0.95	0.88	20	
nga	0.91	0.95	0.93	21	
nya	0.95	0.95	0.95	21	
pa	0.86	1.00	0.92	18	
ra	0.85	0.96	0.90	24	
sa	0.93	1.00	0.96	25	
ta	1.00	0.92	0.96	25	
tha	1.00	0.91	0.95	22	
wa	1.00	1.00	1.00	21	
ya	1.00	0.95	0.97	19	
				accuracy	0.95
				macro avg	0.95
				weighted avg	0.95
					441

Gambar 10. Hasil Classification Report SVM-SIFT Skenario 5

**LAMPIRAN F. Pengujian Parameter Skenario 6**

**Tabel 6. Hasil Pelatihan SVM-SIFT Skenario 6**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.25495	{'C': 3, 'gamma': 0.25495}	1	0.890759	0.009570
6_0.25495	{'C': 6, 'gamma': 0.25495}	2	0.890358	0.007819
6_0.26557	{'C': 6, 'gamma': 0.26557}	3	0.889958	0.007979
6_0.26026	{'C': 6, 'gamma': 0.26026}	3	0.889958	0.006787
3_0.26557	{'C': 3, 'gamma': 0.26557}	5	0.889558	0.009639
3_0.26026	{'C': 3, 'gamma': 0.26026}	6	0.889158	0.009055
6_0.27088	{'C': 6, 'gamma': 0.27088}	7	0.889156	0.007752
6_0.27619	{'C': 6, 'gamma': 0.27619}	8	0.888756	0.009595
3_0.27619	{'C': 3, 'gamma': 0.27619}	9	0.888357	0.011042
3_0.27088	{'C': 3, 'gamma': 0.27088}	9	0.888357	0.009313
10_0.25495	{'C': 10, 'gamma': 0.25495}	11	0.888356	0.008325
10_0.26557	{'C': 10, 'gamma': 0.26557}	12	0.887956	0.008380
10_0.26026	{'C': 10, 'gamma': 0.26026}	12	0.887956	0.007144
15_0.25495	{'C': 15, 'gamma': 0.25495}	14	0.887956	0.008475
15_0.26026	{'C': 15, 'gamma': 0.26026}	15	0.887556	0.007296
15_0.26557	{'C': 15, 'gamma': 0.26557}	15	0.887556	0.008511
10_0.27619	{'C': 10, 'gamma': 0.27619}	17	0.887555	0.010149
15_0.27619	{'C': 15, 'gamma': 0.27619}	18	0.887155	0.010163
10_0.27088	{'C': 10, 'gamma': 0.27088}	19	0.887154	0.008359
15_0.27088	{'C': 15, 'gamma': 0.27088}	20	0.886754	0.008452
1_0.25495	{'C': 1, 'gamma': 0.25495}	21	0.881154	0.006746
1_0.26557	{'C': 1, 'gamma': 0.26557}	22	0.880754	0.006848
1_0.26026	{'C': 1, 'gamma': 0.26026}	23	0.880754	0.006121
1_0.27619	{'C': 1, 'gamma': 0.27619}	24	0.880354	0.007385
1_0.27088	{'C': 1, 'gamma': 0.27088}	24	0.880354	0.007276

		Confusion Matrix Plot																			
Actual	Predicted	ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
		18	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
ba	ba	18	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0
ca	ca	0	25	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
da	da	0	0	22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
dha	dha	0	0	0	27	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
ga	ga	0	0	0	0	19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
ha	ha	0	0	0	0	0	16	0	2	0	0	0	0	0	0	0	0	0	0	0	1
ja	ja	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
ka	ka	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
la	la	0	0	0	0	0	6	0	0	17	0	0	0	0	1	0	0	1	0	0	0
ma	ma	0	0	0	0	0	0	0	1	20	0	0	0	0	0	0	0	0	0	1	0
nya	nya	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
pa	pa	2	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0
ra	ra	0	0	0	0	0	0	0	1	0	0	0	0	0	16	0	0	0	0	0	2
sa	sa	0	2	0	0	0	1	0	0	0	0	0	0	0	0	3	20	0	0	0	0
ta	ta	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	22	0	1
tha	tha	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
wa	wa	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	20	0
ya	ya	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	18

Gambar 11. Hasil Confusion Matrix SVM-SIFT Skenario 6

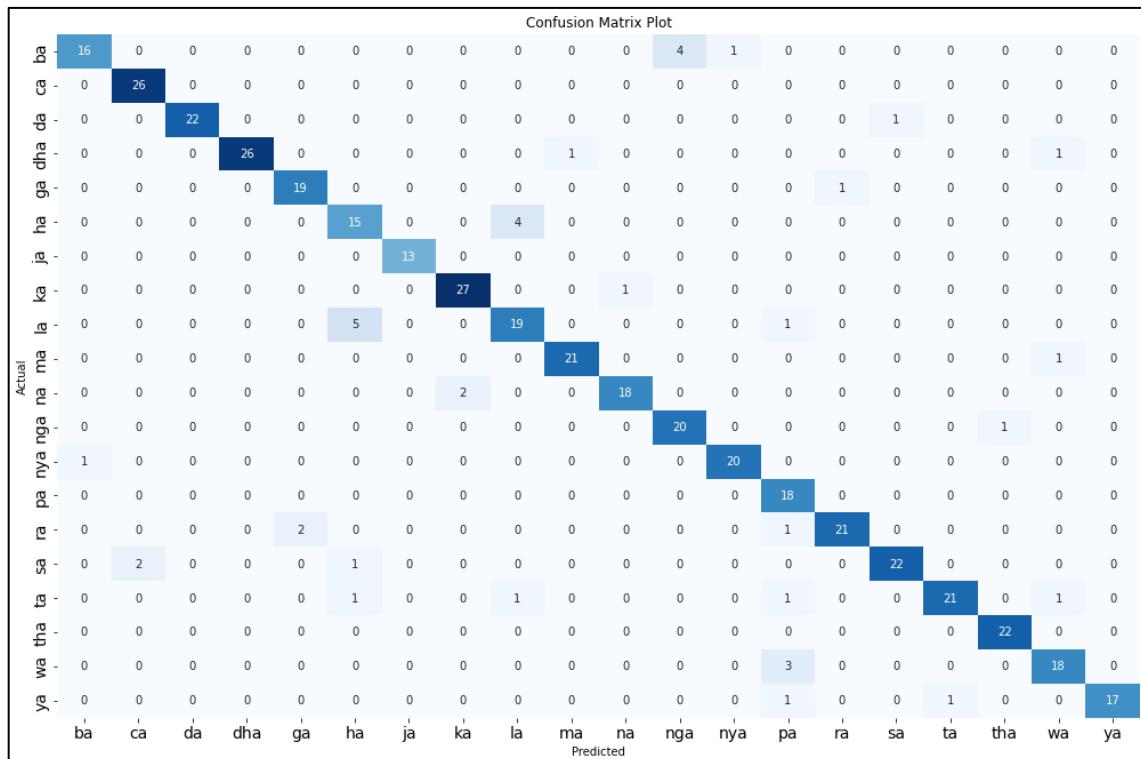
Classification report result:					
	precision	recall	f1-score	support	
ba	0.86	0.86	0.86	21	
ca	0.93	0.96	0.94	26	
da	1.00	0.96	0.98	23	
dha	0.96	0.96	0.96	28	
ga	1.00	0.95	0.97	20	
ha	0.70	0.84	0.76	19	
ja	1.00	1.00	1.00	13	
ka	1.00	1.00	1.00	28	
la	0.77	0.68	0.72	25	
ma	1.00	0.91	0.95	22	
na	1.00	1.00	1.00	20	
nga	0.84	1.00	0.91	21	
nya	1.00	0.90	0.95	21	
pa	0.73	0.89	0.80	18	
ra	0.95	0.83	0.89	24	
sa	0.96	0.88	0.92	25	
ta	0.92	0.88	0.90	25	
tha	1.00	0.95	0.98	22	
wa	0.91	0.95	0.93	21	
ya	0.86	0.95	0.90	19	
accuracy			0.92	441	
macro avg	0.92	0.92	0.92	441	
weighted avg	0.92	0.92	0.92	441	

Gambar 12 Hasil Classification Report SVM-SIFT Skenario 6

**LAMPIRAN G. Pengujian Parameter Skenario 7**

**Tabel 7. Hasil Pelatihan SVM-SIFT Skenario 7**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.20533	{'C': 3, 'gamma': 0.20533}	1	0.899561	0.008128
3_0.20952	{'C': 3, 'gamma': 0.20952}	2	0.899162	0.007836
3_0.21371	{'C': 3, 'gamma': 0.21371}	3	0.897962	0.009271
3_0.2179	{'C': 3, 'gamma': 0.2179}	3	0.897962	0.008170
3_0.20114	{'C': 3, 'gamma': 0.20114}	5	0.897961	0.008080
15_0.20533	{'C': 15, 'gamma': 0.20533}	6	0.897562	0.009209
10_0.20533	{'C': 10, 'gamma': 0.20533}	6	0.897562	0.009209
6_0.20533	{'C': 6, 'gamma': 0.20533}	8	0.897562	0.009042
15_0.20952	{'C': 15, 'gamma': 0.20952}	9	0.896762	0.008779
10_0.20952	{'C': 10, 'gamma': 0.20952}	9	0.896762	0.008779
15_0.2179	{'C': 15, 'gamma': 0.2179}	11	0.896762	0.009730
10_0.2179	{'C': 10, 'gamma': 0.2179}	11	0.896762	0.009730
6_0.2179	{'C': 6, 'gamma': 0.2179}	13	0.896762	0.009487
6_0.20952	{'C': 6, 'gamma': 0.20952}	13	0.896762	0.008603
10_0.20114	{'C': 10, 'gamma': 0.20114}	15	0.896362	0.008667
15_0.20114	{'C': 15, 'gamma': 0.20114}	15	0.896362	0.008667
6_0.20114	{'C': 6, 'gamma': 0.20114}	17	0.896362	0.008394
15_0.21371	{'C': 15, 'gamma': 0.21371}	18	0.895962	0.010318
10_0.21371	{'C': 10, 'gamma': 0.21371}	18	0.895962	0.010318
6_0.21371	{'C': 6, 'gamma': 0.21371}	20	0.895962	0.010168
1_0.2179	{'C': 1, 'gamma': 0.2179}	21	0.892360	0.006696
1_0.21371	{'C': 1, 'gamma': 0.21371}	22	0.891160	0.008016
1_0.20952	{'C': 1, 'gamma': 0.20952}	22	0.891160	0.008016
1_0.20533	{'C': 1, 'gamma': 0.20533}	22	0.891160	0.008016
1_0.20114	{'C': 1, 'gamma': 0.20114}	25	0.890759	0.007614



Gambar 13. Hasil Confusion Matrix SVM-SIFT Skenario 7

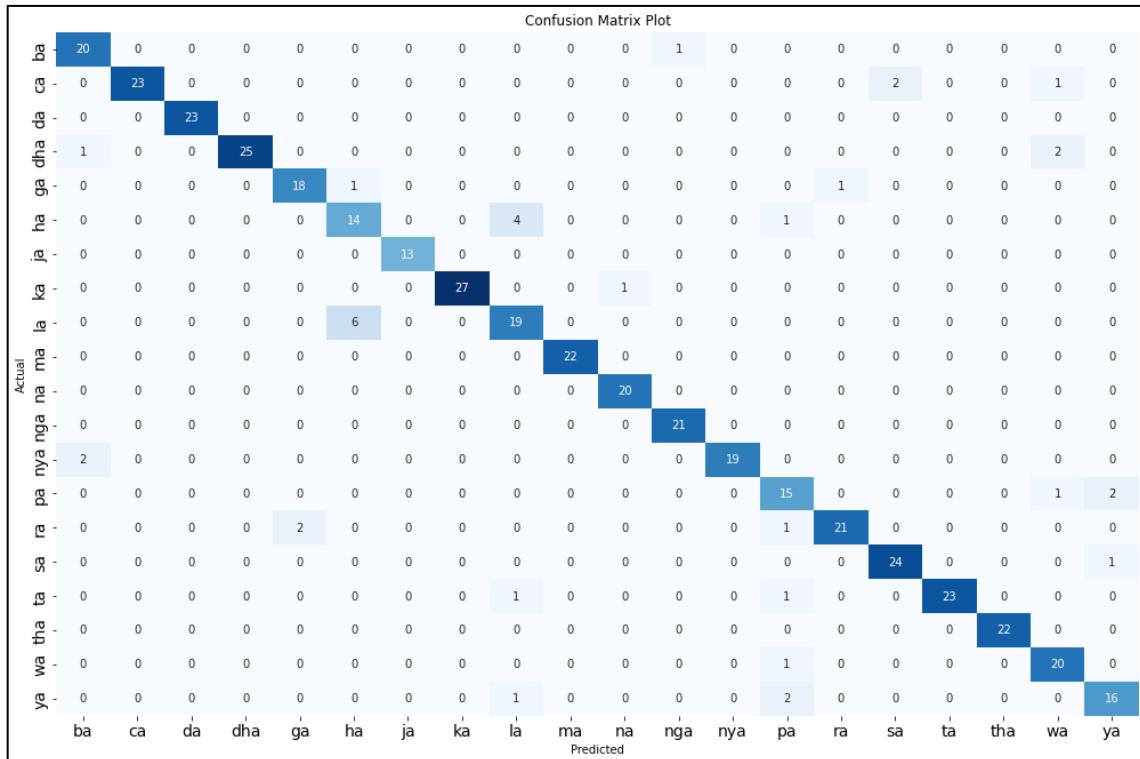
Classification report result:				
	precision	recall	f1-score	support
ba	0.94	0.76	0.84	21
ca	0.93	1.00	0.96	26
da	1.00	0.96	0.98	23
dha	1.00	0.93	0.96	28
ga	0.90	0.95	0.93	20
ha	0.68	0.79	0.73	19
ja	1.00	1.00	1.00	13
ka	0.93	0.96	0.95	28
la	0.79	0.76	0.78	25
ma	0.95	0.95	0.95	22
na	0.95	0.90	0.92	20
nga	0.83	0.95	0.89	21
nya	0.95	0.95	0.95	21
pa	0.72	1.00	0.84	18
ra	0.95	0.88	0.91	24
sa	0.96	0.88	0.92	25
ta	0.95	0.84	0.89	25
tha	0.96	1.00	0.98	22
wa	0.86	0.86	0.86	21
ya	1.00	0.89	0.94	19
accuracy			0.91	441
macro avg	0.91	0.91	0.91	441
weighted avg	0.92	0.91	0.91	441

Gambar 14. Hasil Classification Report SVM-SIFT Skenario 7

**LAMPIRAN H. Pengujian Parameter Skenario 8**

**Tabel 8. Hasil Pelatihan SVM-SIFT Skenario 8**

kernel	params	rank_test_score	mean_test_score	std_test_score
6_0.13022	{'C': 6, 'gamma': 0.13022}	1	0.906367	0.009873
6_0.13293	{'C': 6, 'gamma': 0.13293}	2	0.905567	0.009871
15_0.13022	{'C': 15, 'gamma': 0.13022}	3	0.905167	0.010437
10_0.13022	{'C': 10, 'gamma': 0.13022}	3	0.905167	0.010437
6_0.13564	{'C': 6, 'gamma': 0.13564}	5	0.904767	0.010738
6_0.13835	{'C': 6, 'gamma': 0.13835}	6	0.904766	0.009809
15_0.13293	{'C': 15, 'gamma': 0.13293}	7	0.904367	0.010342
10_0.13293	{'C': 10, 'gamma': 0.13293}	7	0.904367	0.010342
6_0.14106	{'C': 6, 'gamma': 0.14106}	9	0.904366	0.010430
15_0.13564	{'C': 15, 'gamma': 0.13564}	10	0.903567	0.011229
10_0.13564	{'C': 10, 'gamma': 0.13564}	10	0.903567	0.011229
10_0.13835	{'C': 10, 'gamma': 0.13835}	12	0.903566	0.010267
15_0.13835	{'C': 15, 'gamma': 0.13835}	12	0.903566	0.010267
10_0.14106	{'C': 10, 'gamma': 0.14106}	14	0.903566	0.010504
15_0.14106	{'C': 15, 'gamma': 0.14106}	14	0.903566	0.010504
3_0.13022	{'C': 3, 'gamma': 0.13022}	16	0.903166	0.009721
3_0.14106	{'C': 3, 'gamma': 0.14106}	17	0.903166	0.010051
3_0.13835	{'C': 3, 'gamma': 0.13835}	18	0.902766	0.009219
3_0.13564	{'C': 3, 'gamma': 0.13564}	19	0.902366	0.010026
3_0.13293	{'C': 3, 'gamma': 0.13293}	19	0.902366	0.009366
1_0.13293	{'C': 1, 'gamma': 0.13293}	21	0.899962	0.006896
1_0.13564	{'C': 1, 'gamma': 0.13564}	22	0.899562	0.006824
1_0.13022	{'C': 1, 'gamma': 0.13022}	22	0.899562	0.006824
1_0.14106	{'C': 1, 'gamma': 0.14106}	24	0.899562	0.007714
1_0.13835	{'C': 1, 'gamma': 0.13835}	25	0.898762	0.007083



Gambar 15. Hasil Confusion Matrix SVM-SIFT Skenario 8

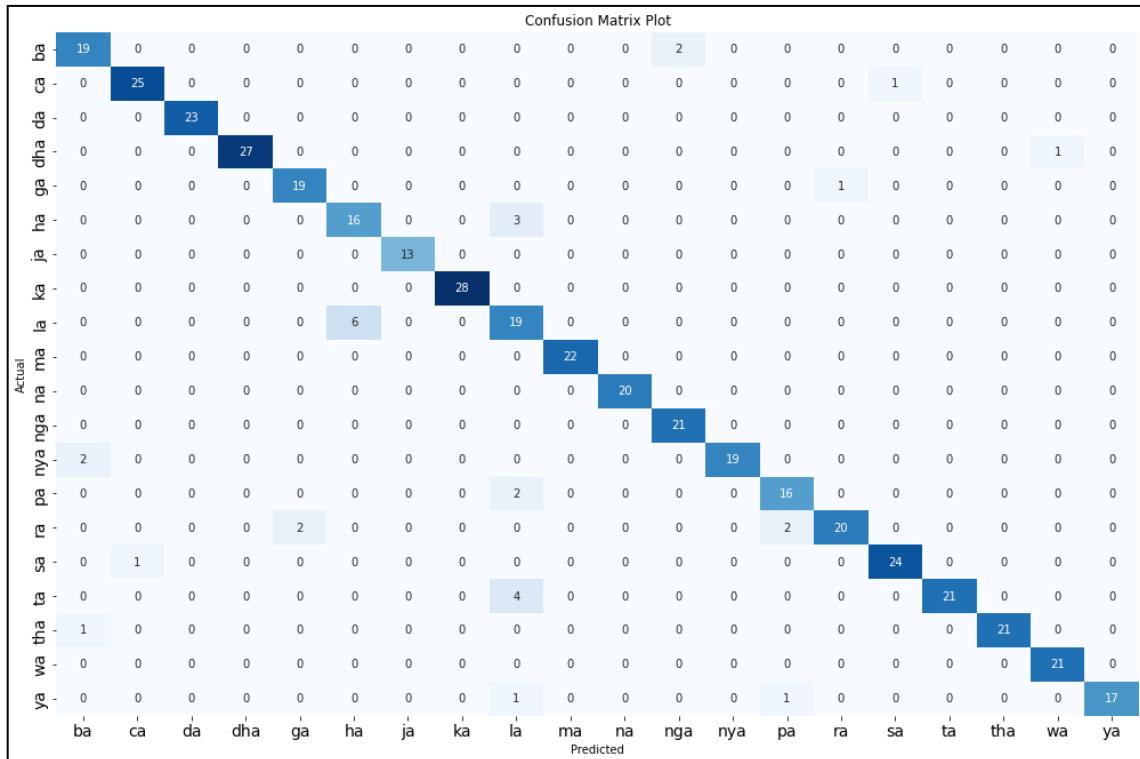
Classification report result:					
	precision	recall	f1-score	support	
ba	0.87	0.95	0.91	21	
ca	1.00	0.88	0.94	26	
da	1.00	1.00	1.00	23	
dha	1.00	0.89	0.94	28	
ga	0.90	0.90	0.90	20	
ha	0.67	0.74	0.70	19	
ja	1.00	1.00	1.00	13	
ka	1.00	0.96	0.98	28	
la	0.76	0.76	0.76	25	
ma	1.00	1.00	1.00	22	
na	0.95	1.00	0.98	20	
nga	0.95	1.00	0.98	21	
nya	1.00	0.90	0.95	21	
pa	0.71	0.83	0.77	18	
ra	0.95	0.88	0.91	24	
sa	0.92	0.96	0.94	25	
ta	1.00	0.92	0.96	25	
tha	1.00	1.00	1.00	22	
wa	0.83	0.95	0.89	21	
ya	0.84	0.84	0.84	19	
accuracy				0.92	441
macro avg				0.92	0.92
weighted avg				0.92	0.92
				441	

Gambar 16. Hasil Classification Report SVM-SIFT Skenario 8

**LAMPIRAN I. Pengujian Parameter Skenario 9**

**Tabel 9. Hasil Pelatihan SVM-SIFT Skenario 9**

kernel	params	rank_test_score	mean_test_score	std_test_score
15_0.10009	{'C': 15, 'gamma': 0.10009}	1	0.907967	0.009765
10_0.10009	{'C': 10, 'gamma': 0.10009}	1	0.907967	0.009765
6_0.10009	{'C': 6, 'gamma': 0.10009}	1	0.907967	0.009765
15_0.10217	{'C': 15, 'gamma': 0.10217}	4	0.907567	0.009631
10_0.10217	{'C': 10, 'gamma': 0.10217}	4	0.907567	0.009631
6_0.10217	{'C': 6, 'gamma': 0.10217}	4	0.907567	0.009631
6_0.10425	{'C': 6, 'gamma': 0.10425}	7	0.907167	0.009729
15_0.10425	{'C': 15, 'gamma': 0.10425}	7	0.907167	0.009729
10_0.10841	{'C': 10, 'gamma': 0.10841}	7	0.907167	0.010595
10_0.10633	{'C': 10, 'gamma': 0.10633}	7	0.907167	0.009729
10_0.10425	{'C': 10, 'gamma': 0.10425}	7	0.907167	0.009729
6_0.10841	{'C': 6, 'gamma': 0.10841}	7	0.907167	0.010595
6_0.10633	{'C': 6, 'gamma': 0.10633}	7	0.907167	0.009729
15_0.10841	{'C': 15, 'gamma': 0.10841}	7	0.907167	0.010595
15_0.10633	{'C': 15, 'gamma': 0.10633}	7	0.907167	0.009729
3_0.10841	{'C': 3, 'gamma': 0.10841}	16	0.906367	0.012062
3_0.10633	{'C': 3, 'gamma': 0.10633}	17	0.905967	0.011900
3_0.10217	{'C': 3, 'gamma': 0.10217}	17	0.905967	0.011628
3_0.10009	{'C': 3, 'gamma': 0.10009}	17	0.905967	0.011628
3_0.10425	{'C': 3, 'gamma': 0.10425}	20	0.905567	0.011655
1_0.10425	{'C': 1, 'gamma': 0.10425}	21	0.891157	0.010320
1_0.10009	{'C': 1, 'gamma': 0.10009}	21	0.891157	0.009925
1_0.10841	{'C': 1, 'gamma': 0.10841}	23	0.890756	0.010012
1_0.10633	{'C': 1, 'gamma': 0.10633}	23	0.890756	0.010012
1_0.10217	{'C': 1, 'gamma': 0.10217}	25	0.890357	0.009660



Gambar 17. Hasil Confusion Matrix SVM-SIFT Skenario 9

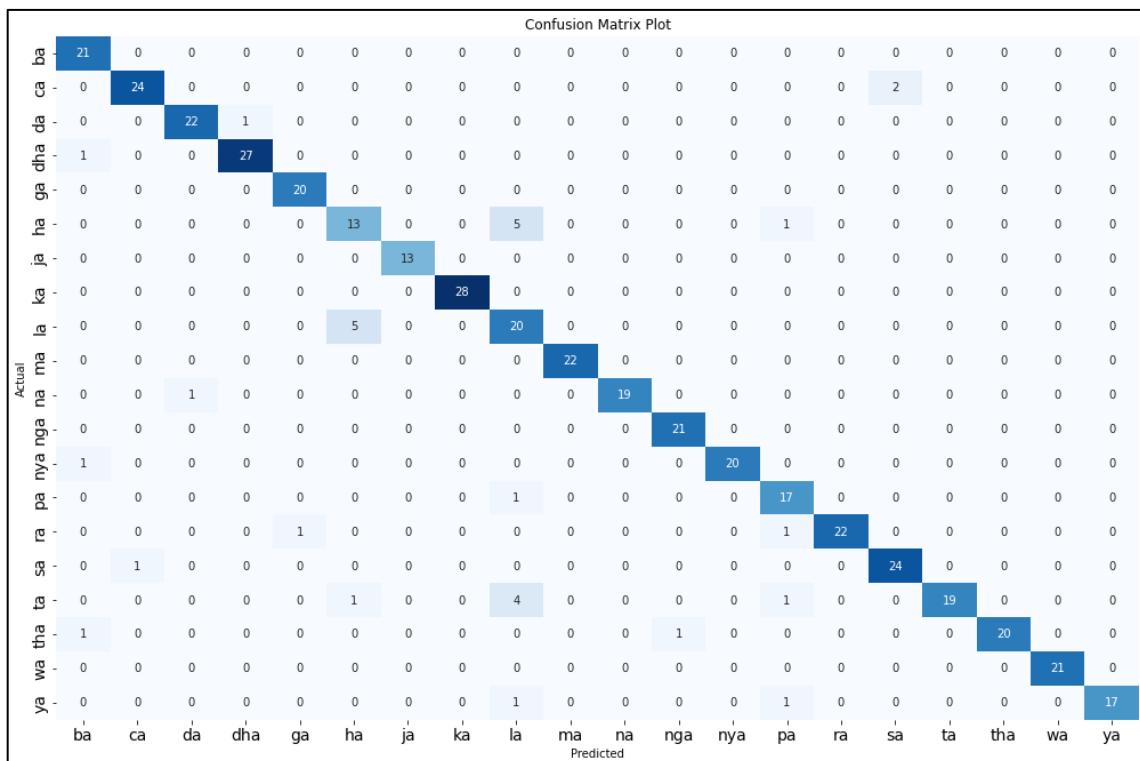
Classification report result:				
	precision	recall	f1-score	support
ba	0.86	0.90	0.88	21
ca	0.96	0.96	0.96	26
da	1.00	1.00	1.00	23
dha	1.00	0.96	0.98	28
ga	0.90	0.95	0.93	20
ha	0.73	0.84	0.78	19
ja	1.00	1.00	1.00	13
ka	1.00	1.00	1.00	28
la	0.66	0.76	0.70	25
ma	1.00	1.00	1.00	22
na	1.00	1.00	1.00	20
nga	0.91	1.00	0.95	21
nya	1.00	0.90	0.95	21
pa	0.84	0.89	0.86	18
ra	0.95	0.83	0.89	24
sa	0.96	0.96	0.96	25
ta	1.00	0.84	0.91	25
tha	1.00	0.95	0.98	22
wa	0.95	1.00	0.98	21
ya	1.00	0.89	0.94	19
accuracy				0.93
macro avg				0.94
weighted avg				0.94
				441

Gambar 18. Hasil Classification Report SVM-SIFT Skenario 9

**LAMPIRAN J. Pengujian Parameter Skenario 10**

**Tabel 10. Hasil Pelatihan SVM-SIFT Skenario 10**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.08725	{'C': 3, 'gamma': 0.08725}	1	0.900761	0.011699
6_0.08725	{'C': 6, 'gamma': 0.08725}	2	0.900361	0.010978
10_0.08725	{'C': 10, 'gamma': 0.08725}	3	0.899961	0.011020
15_0.08725	{'C': 15, 'gamma': 0.08725}	3	0.899961	0.011020
6_0.08907	{'C': 6, 'gamma': 0.08907}	5	0.899560	0.012092
6_0.09089	{'C': 6, 'gamma': 0.09089}	6	0.899559	0.012099
6_0.09271	{'C': 6, 'gamma': 0.09271}	6	0.899559	0.012099
3_0.09271	{'C': 3, 'gamma': 0.09271}	6	0.899559	0.013235
3_0.08907	{'C': 3, 'gamma': 0.08907}	6	0.899559	0.012868
3_0.09089	{'C': 3, 'gamma': 0.09089}	6	0.899559	0.012868
10_0.08907	{'C': 10, 'gamma': 0.08907}	11	0.899160	0.012170
15_0.08907	{'C': 15, 'gamma': 0.08907}	11	0.899160	0.012170
10_0.09271	{'C': 10, 'gamma': 0.09271}	13	0.899159	0.012111
15_0.09271	{'C': 15, 'gamma': 0.09271}	13	0.899159	0.012111
6_0.09453	{'C': 6, 'gamma': 0.09453}	13	0.899159	0.011361
15_0.09089	{'C': 15, 'gamma': 0.09089}	13	0.899159	0.012111
10_0.09089	{'C': 10, 'gamma': 0.09089}	13	0.899159	0.012111
10_0.09453	{'C': 10, 'gamma': 0.09453}	18	0.898759	0.011359
15_0.09453	{'C': 15, 'gamma': 0.09453}	18	0.898759	0.011359
3_0.09453	{'C': 3, 'gamma': 0.09453}	18	0.898759	0.011774
1_0.08725	{'C': 1, 'gamma': 0.08725}	21	0.892759	0.006851
1_0.09089	{'C': 1, 'gamma': 0.09089}	22	0.891959	0.008736
1_0.08907	{'C': 1, 'gamma': 0.08907}	22	0.891959	0.008736
1_0.09271	{'C': 1, 'gamma': 0.09271}	24	0.891958	0.008079
1_0.09453	{'C': 1, 'gamma': 0.09453}	25	0.891158	0.008976



Gambar 19. Hasil Confusion Matrix SVM-SIFT Skenario 10

Classification report result:					
	precision	recall	f1-score	support	
ba	0.88	1.00	0.93	21	
ca	0.96	0.92	0.94	26	
da	0.96	0.96	0.96	23	
dha	0.96	0.96	0.96	28	
ga	0.95	1.00	0.98	20	
ha	0.68	0.68	0.68	19	
ja	1.00	1.00	1.00	13	
ka	1.00	1.00	1.00	28	
la	0.65	0.80	0.71	25	
ma	1.00	1.00	1.00	22	
na	1.00	0.95	0.97	20	
nga	0.95	1.00	0.98	21	
nya	1.00	0.95	0.98	21	
pa	0.81	0.94	0.87	18	
ra	1.00	0.92	0.96	24	
sa	0.92	0.96	0.94	25	
ta	1.00	0.76	0.86	25	
tha	1.00	0.91	0.95	22	
wa	1.00	1.00	1.00	21	
ya	1.00	0.89	0.94	19	
				0.93	441
				accuracy	
				macro avg	0.94 0.93 0.93 441
				weighted avg	0.94 0.93 0.93 441

Gambar 20. Hasil Classification Report SVM-SIFT Skenario 10

**LAMPIRAN K. Pengujian Parameter Skenario 11**

**Tabel 11. Hasil Pelatihan SVM-SIFT Skenario 11**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.26821	{'C': 3, 'gamma': 0.26821}	1	0.884762	0.012750
3_0.27368	{'C': 3, 'gamma': 0.27368}	1	0.884762	0.012750
3_0.27915	{'C': 3, 'gamma': 0.27915}	1	0.884762	0.012561
6_0.27915	{'C': 6, 'gamma': 0.27915}	4	0.883962	0.012196
3_0.28462	{'C': 3, 'gamma': 0.28462}	5	0.883962	0.012456
3_0.26274	{'C': 3, 'gamma': 0.26274}	6	0.883962	0.011869
6_0.27368	{'C': 6, 'gamma': 0.27368}	7	0.883562	0.012286
6_0.28462	{'C': 6, 'gamma': 0.28462}	8	0.883162	0.012167
6_0.26821	{'C': 6, 'gamma': 0.26821}	9	0.882762	0.012681
15_0.27368	{'C': 15, 'gamma': 0.27368}	10	0.882761	0.011147
10_0.27915	{'C': 10, 'gamma': 0.27915}	10	0.882761	0.011147
15_0.27915	{'C': 15, 'gamma': 0.27915}	10	0.882761	0.011147
10_0.27368	{'C': 10, 'gamma': 0.27368}	10	0.882761	0.011147
6_0.26274	{'C': 6, 'gamma': 0.26274}	14	0.881962	0.012129
15_0.26821	{'C': 15, 'gamma': 0.26821}	15	0.881961	0.011456
10_0.28462	{'C': 10, 'gamma': 0.28462}	15	0.881961	0.011029
15_0.28462	{'C': 15, 'gamma': 0.28462}	15	0.881961	0.011029
10_0.26821	{'C': 10, 'gamma': 0.26821}	15	0.881961	0.011456
10_0.26274	{'C': 10, 'gamma': 0.26274}	19	0.881561	0.011621
15_0.26274	{'C': 15, 'gamma': 0.26274}	20	0.881160	0.010930
1_0.28462	{'C': 1, 'gamma': 0.28462}	21	0.879162	0.014274
1_0.27368	{'C': 1, 'gamma': 0.27368}	21	0.879162	0.014274
1_0.27915	{'C': 1, 'gamma': 0.27915}	23	0.878762	0.014385
1_0.26821	{'C': 1, 'gamma': 0.26821}	23	0.878762	0.014550
1_0.26274	{'C': 1, 'gamma': 0.26274}	25	0.878362	0.014593

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	ca	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0
da	da	0	0	21	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
	dha	0	0	0	26	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
ga	ga	0	0	0	0	19	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	ha	0	0	0	0	0	14	0	0	3	0	0	0	1	0	0	0	0	0	0	1
ja	ja	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	1	0
	ka	0	0	0	0	0	0	0	25	0	0	3	0	0	0	0	0	0	0	0	0
la	la	0	0	0	0	0	3	0	0	22	0	0	0	0	0	0	0	0	0	0	0
	ma	0	1	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0
na	na	0	0	0	0	0	0	1	0	0	19	0	0	0	0	0	0	0	0	0	0
	nya	2	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0
pa	pa	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0
	ra	0	0	0	0	1	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0
sa	sa	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	23	0	0	0	0
	ta	0	0	0	0	0	4	0	0	2	0	0	0	0	0	0	0	19	0	0	0
tha	tha	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	21	0	0
	wa	0	0	0	0	0	1	0	0	0	0	0	2	0	0	0	0	0	18	0	0
ya	ya	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	18	0
	ba	ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya

Gambar 21. Hasil Confusion Matrix SVM-SIFT Skenario 11

		Classification report result:																		
		precision	recall	f1-score	support															
ba	ba	0.88	1.00	0.93	21															
ca	ca	0.92	0.85	0.88	26															
da	da	1.00	0.91	0.95	23															
dha	dha	1.00	0.93	0.96	28															
ga	ga	0.95	0.95	0.95	20															
ha	ha	0.67	0.74	0.70	19															
ja	ja	0.86	0.92	0.89	13															
ka	ka	0.96	0.89	0.93	28															
la	la	0.81	0.88	0.85	25															
ma	ma	0.91	0.95	0.93	22															
na	na	0.79	0.95	0.86	20															
nga	nga	1.00	0.90	0.95	21															
nya	nya	0.91	0.95	0.93	21															
pa	pa	0.82	1.00	0.90	18															
ra	ra	1.00	0.96	0.98	24															
sa	sa	0.85	0.92	0.88	25															
ta	ta	1.00	0.76	0.86	25															
tha	tha	1.00	0.95	0.98	22															
wa	wa	0.95	0.86	0.90	21															
ya	ya	0.95	0.95	0.95	19															
		accuracy		0.91	441															
		macro avg	0.91	0.91	441															
		weighted avg	0.92	0.91	441															

Gambar 22. Hasil Classification Report SVM-SIFT Skenario 11

**LAMPIRAN L. Pengujian Parameter Skenario 12**

**Tabel 12. Hasil Pelatihan SVM-SIFT Skenario 12**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.20183	{'C': 3, 'gamma': 0.20183}	1	0.909166	0.011960
3_0.21023	{'C': 3, 'gamma': 0.21023}	2	0.908766	0.011270
3_0.20603	{'C': 3, 'gamma': 0.20603}	2	0.908766	0.012092
3_0.21443	{'C': 3, 'gamma': 0.21443}	4	0.907966	0.011153
3_0.21863	{'C': 3, 'gamma': 0.21863}	4	0.907966	0.011153
6_0.20183	{'C': 6, 'gamma': 0.20183}	6	0.907565	0.015031
6_0.21863	{'C': 6, 'gamma': 0.21863}	7	0.907166	0.014495
6_0.21023	{'C': 6, 'gamma': 0.21023}	8	0.907165	0.014609
10_0.20183	{'C': 10, 'gamma': 0.20183}	9	0.907163	0.013889
6_0.20603	{'C': 6, 'gamma': 0.20603}	10	0.906765	0.015614
6_0.21443	{'C': 6, 'gamma': 0.21443}	10	0.906765	0.014332
10_0.20603	{'C': 10, 'gamma': 0.20603}	12	0.906363	0.014440
10_0.21023	{'C': 10, 'gamma': 0.21023}	12	0.906363	0.013701
15_0.20183	{'C': 15, 'gamma': 0.20183}	14	0.906362	0.014109
10_0.21443	{'C': 10, 'gamma': 0.21443}	15	0.905563	0.013582
10_0.21863	{'C': 10, 'gamma': 0.21863}	15	0.905563	0.013582
15_0.20603	{'C': 15, 'gamma': 0.20603}	17	0.905562	0.014609
15_0.21023	{'C': 15, 'gamma': 0.21023}	17	0.905562	0.013879
15_0.21863	{'C': 15, 'gamma': 0.21863}	19	0.905162	0.013539
15_0.21443	{'C': 15, 'gamma': 0.21443}	19	0.905162	0.013539
1_0.21443	{'C': 1, 'gamma': 0.21443}	21	0.900366	0.011916
1_0.21023	{'C': 1, 'gamma': 0.21023}	22	0.900366	0.011921
1_0.20183	{'C': 1, 'gamma': 0.20183}	23	0.900365	0.011447
1_0.21863	{'C': 1, 'gamma': 0.21863}	24	0.899966	0.012220
1_0.20603	{'C': 1, 'gamma': 0.20603}	25	0.899565	0.011856

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	20	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
	ca	0	18	1	2	1	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0
Actual	da	0	1	21	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	dha	0	0	0	27	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Actual	ga	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ha	0	0	0	0	0	14	0	0	4	0	0	0	0	1	0	0	0	0	0	0
Actual	ja	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
	ka	0	0	0	0	0	0	0	26	0	0	2	0	0	0	0	0	0	0	0	0
Actual	la	0	0	0	0	0	4	0	0	21	0	0	0	0	0	0	0	0	0	0	0
	ma	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0
Actual	na	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0
	nya	1	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
Actual	pa	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	1	0
	ra	0	0	0	0	1	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0
Actual	sa	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	23	0	0	0	0
	ta	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	22	0	0	0
Actual	wa	0	0	0	0	0	0	2	0	0	0	0	0	2	0	0	0	0	17	0	0
	ya	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	18	0
Predicted		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya

Gambar 23. Hasil Confusion Matrix SVM-SIFT Skenario 12

Classification report result:					
	precision	recall	f1-score	support	
ba	0.95	0.95	0.95	21	
ca	0.95	0.69	0.80	26	
da	0.91	0.91	0.91	23	
dha	0.90	0.96	0.93	28	
ga	0.91	1.00	0.95	20	
ha	0.70	0.74	0.72	19	
ja	0.76	1.00	0.87	13	
ka	0.96	0.93	0.95	28	
la	0.81	0.84	0.82	25	
ma	1.00	1.00	1.00	22	
na	0.91	1.00	0.95	20	
nga	0.95	0.95	0.95	21	
nya	1.00	0.95	0.98	21	
pa	0.85	0.94	0.89	18	
ra	1.00	0.96	0.98	24	
sa	0.85	0.92	0.88	25	
ta	1.00	0.88	0.94	25	
tha	1.00	1.00	1.00	22	
wa	0.89	0.81	0.85	21	
ya	1.00	0.95	0.97	19	
accuracy			0.92	441	
macro avg	0.92	0.92	0.92	441	
weighted avg	0.92	0.92	0.92	441	

Gambar 24. Hasil Classification Report SVM-SIFT Skenario 12

**LAMPIRAN M. Pengujian Parameter Skenario 13**

**Tabel 13. Hasil Pelatihan SVM-SIFT Skenario 13**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.13196	{'C': 3, 'gamma': 0.13196}	1	0.918374	0.010795
3_0.1346	{'C': 3, 'gamma': 0.1346}	1	0.918374	0.010795
6_0.13724	{'C': 6, 'gamma': 0.13724}	3	0.918374	0.010498
15_0.13724	{'C': 15, 'gamma': 0.13724}	4	0.917974	0.011134
10_0.13724	{'C': 10, 'gamma': 0.13724}	4	0.917974	0.011134
3_0.13724	{'C': 3, 'gamma': 0.13724}	4	0.917974	0.010313
6_0.1346	{'C': 6, 'gamma': 0.1346}	7	0.917574	0.010420
10_0.1346	{'C': 10, 'gamma': 0.1346}	8	0.917174	0.011031
15_0.1346	{'C': 15, 'gamma': 0.1346}	8	0.917174	0.011031
6_0.13196	{'C': 6, 'gamma': 0.13196}	8	0.917174	0.011031
10_0.13196	{'C': 10, 'gamma': 0.13196}	11	0.916774	0.011664
15_0.13196	{'C': 15, 'gamma': 0.13196}	11	0.916774	0.011664
6_0.12668	{'C': 6, 'gamma': 0.12668}	13	0.916774	0.010356
3_0.12932	{'C': 3, 'gamma': 0.12932}	13	0.916774	0.010884
6_0.12932	{'C': 6, 'gamma': 0.12932}	15	0.916374	0.010941
10_0.12668	{'C': 10, 'gamma': 0.12668}	15	0.916374	0.010941
15_0.12668	{'C': 15, 'gamma': 0.12668}	15	0.916374	0.010941
3_0.12668	{'C': 3, 'gamma': 0.12668}	18	0.916373	0.010266
10_0.12932	{'C': 10, 'gamma': 0.12932}	19	0.915974	0.011552
15_0.12932	{'C': 15, 'gamma': 0.12932}	19	0.915974	0.011552
1_0.13724	{'C': 1, 'gamma': 0.13724}	21	0.906370	0.010410
1_0.1346	{'C': 1, 'gamma': 0.1346}	22	0.906369	0.009448
1_0.13196	{'C': 1, 'gamma': 0.13196}	23	0.905969	0.009413
1_0.12668	{'C': 1, 'gamma': 0.12668}	24	0.905169	0.010115
1_0.12932	{'C': 1, 'gamma': 0.12932}	25	0.904769	0.010193

		Confusion Matrix Plot																				
		Predicted																				
Actual	ba	ba	18	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	
		ca	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
Actual	da	da	0	0	22	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
		dha	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Actual	ga	ga	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
		ha	0	0	0	0	0	17	0	0	2	0	0	0	0	0	0	1	0	0	0	0
Actual	ja	ja	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	1	0	0	0	0
		ka	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
Actual	la	la	0	0	0	0	0	4	0	0	21	0	0	0	0	0	0	0	0	0	0	0
		ma	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	0
Actual	na	na	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0
		nya	2	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0
Actual	nyanya	nyanya	1	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
		pa	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0
Actual	ra	ra	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0
		sa	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0	0
Actual	ta	ta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
		tha	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0
Actual	wa	wa	0	0	0	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0	17	0
		ya	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	18

Gambar 25. Hasil Confusion Matrix SVM-SIFT Skenario 13

		Classification report result:				
		precision	recall	f1-score	support	
	ba	0.86	0.86	0.86	21	
	ca	0.92	0.88	0.90	26	
	da	0.96	0.96	0.96	23	
	dha	0.96	0.96	0.96	28	
	ga	0.95	0.95	0.95	20	
	ha	0.81	0.89	0.85	19	
	ja	1.00	0.92	0.96	13	
	ka	1.00	1.00	1.00	28	
	la	0.91	0.84	0.87	25	
	ma	1.00	1.00	1.00	22	
	na	0.95	1.00	0.98	20	
	nga	0.95	0.90	0.93	21	
	nya	0.91	0.95	0.93	21	
	pa	0.78	1.00	0.88	18	
	ra	0.92	0.96	0.94	24	
	sa	0.88	0.88	0.88	25	
	ta	1.00	0.96	0.98	25	
	tha	1.00	1.00	1.00	22	
	wa	0.94	0.81	0.87	21	
	ya	1.00	0.95	0.97	19	
		accuracy			0.93	
		macro avg			0.94	
		weighted avg			0.94	

Gambar 26. Hasil Classification Report SVM-SIFT Skenario 13

**LAMPIRAN N. Pengujian Parameter Skenario 14**

**Tabel 14. Hasil Pelatihan SVM-SIFT Skenario 14**

kernel	params	rank_test_score	mean_test_score	std_test_score
15_0.10234	{'C': 15, 'gamma': 0.10234}	1	0.921174	0.012012
10_0.10234	{'C': 10, 'gamma': 0.10234}	1	0.921174	0.012012
6_0.10234	{'C': 6, 'gamma': 0.10234}	1	0.921174	0.012012
6_0.1066	{'C': 6, 'gamma': 0.1066}	4	0.920774	0.011605
15_0.1066	{'C': 15, 'gamma': 0.1066}	4	0.920774	0.011605
15_0.10447	{'C': 15, 'gamma': 0.10447}	4	0.920774	0.011605
10_0.1066	{'C': 10, 'gamma': 0.1066}	4	0.920774	0.011605
10_0.10447	{'C': 10, 'gamma': 0.10447}	4	0.920774	0.011605
6_0.10447	{'C': 6, 'gamma': 0.10447}	4	0.920774	0.011605
10_0.10873	{'C': 10, 'gamma': 0.10873}	10	0.920373	0.011172
15_0.10873	{'C': 15, 'gamma': 0.10873}	10	0.920373	0.011172
6_0.10873	{'C': 6, 'gamma': 0.10873}	10	0.920373	0.011172
10_0.11086	{'C': 10, 'gamma': 0.11086}	13	0.919573	0.011999
6_0.11086	{'C': 6, 'gamma': 0.11086}	13	0.919573	0.011999
15_0.11086	{'C': 15, 'gamma': 0.11086}	13	0.919573	0.011999
3_0.1066	{'C': 3, 'gamma': 0.1066}	16	0.919174	0.012968
3_0.10234	{'C': 3, 'gamma': 0.10234}	16	0.919174	0.013273
3_0.10447	{'C': 3, 'gamma': 0.10447}	18	0.918774	0.012906
3_0.10873	{'C': 3, 'gamma': 0.10873}	19	0.918773	0.012532
3_0.11086	{'C': 3, 'gamma': 0.11086}	20	0.917973	0.013418
1_0.1066	{'C': 1, 'gamma': 0.1066}	21	0.906366	0.016346
1_0.10447	{'C': 1, 'gamma': 0.10447}	21	0.906366	0.016346
1_0.10234	{'C': 1, 'gamma': 0.10234}	23	0.905566	0.016636
1_0.10873	{'C': 1, 'gamma': 0.10873}	24	0.905166	0.016792
1_0.11086	{'C': 1, 'gamma': 0.11086}	25	0.904766	0.016980

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	19	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
	ca	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0
da	0	0	21	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
	dha	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ga	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ha	0	0	0	0	0	17	0	0	2	0	0	0	0	0	0	0	0	0	0	0
ja	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
	ka	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
la	0	0	0	0	0	2	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0
	ma	0	0	0	0	0	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0
nya	0	0	0	0	0	0	0	1	0	0	0	19	0	0	0	0	0	0	0	0	0
	na	1	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0
pa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	1
	ra	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0
sa	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0
	ta	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	23	0	0
tha	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
	wa	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	18
ya	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	18
	ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya	

Gambar 27. Hasil Confusion Matrix SVM-SIFT Skenario 14

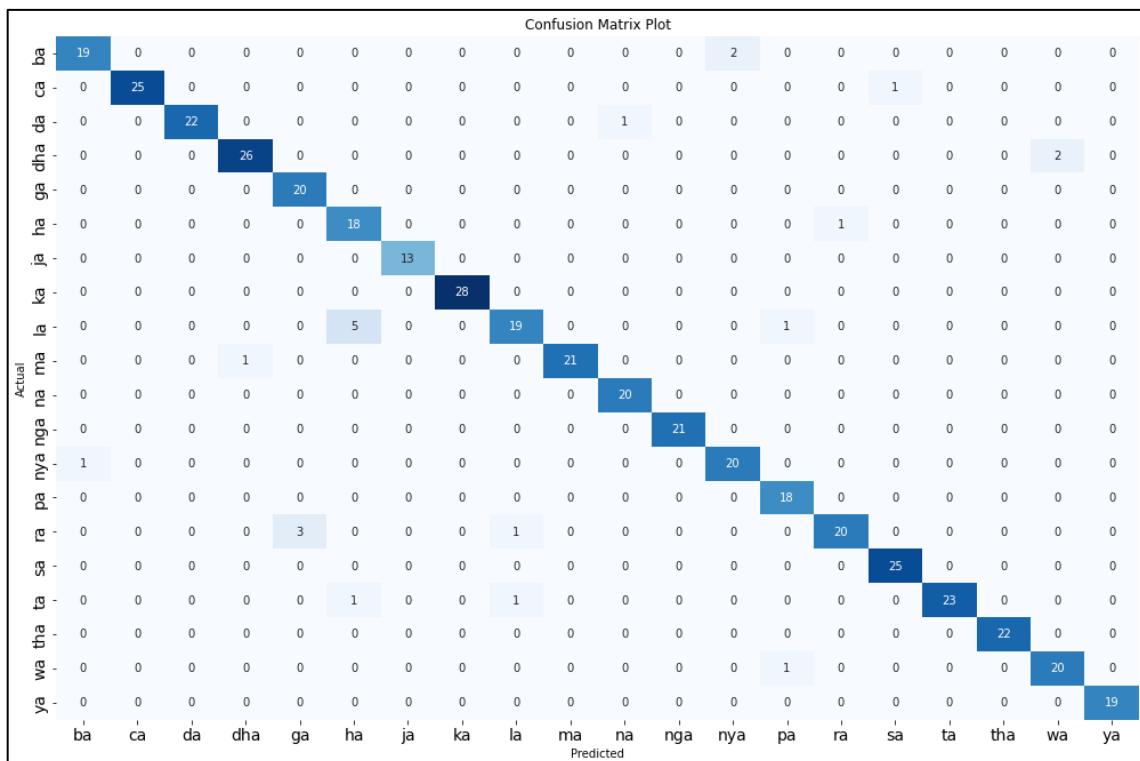
Classification report result:				
	precision	recall	f1-score	support
ba	0.86	0.90	0.88	21
ca	1.00	0.88	0.94	26
da	0.95	0.91	0.93	23
dha	1.00	1.00	1.00	28
ga	0.95	1.00	0.98	20
ha	0.85	0.89	0.87	19
ja	1.00	1.00	1.00	13
ka	0.97	1.00	0.98	28
la	0.85	0.92	0.88	25
ma	1.00	1.00	1.00	22
na	0.95	0.95	0.95	20
nga	1.00	0.95	0.98	21
nya	0.91	0.95	0.93	21
pa	0.85	0.94	0.89	18
ra	1.00	0.96	0.98	24
sa	0.89	0.96	0.92	25
ta	1.00	0.92	0.96	25
tha	1.00	0.95	0.98	22
wa	1.00	0.86	0.92	21
ya	0.90	0.95	0.92	19
accuracy				0.95
macro avg				0.95
weighted avg				0.95
				441

Gambar 28 Hasil Classification Report SVM-SIFT Skenario 14

## LAMPIRAN O. Pengujian Parameter Skenario 15

**Tabel 15. Hasil Pelatihan SVM-SIFT Skenario 15**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.08487	{'C': 3, 'gamma': 0.08487}	1	0.926778	0.015290
3_0.08664	{'C': 3, 'gamma': 0.08664}	2	0.926378	0.015945
3_0.09195	{'C': 3, 'gamma': 0.09195}	3	0.925978	0.015362
3_0.08841	{'C': 3, 'gamma': 0.08841}	3	0.925978	0.016272
15_0.09195	{'C': 15, 'gamma': 0.09195}	5	0.925578	0.015742
10_0.08487	{'C': 10, 'gamma': 0.08487}	5	0.925578	0.016144
6_0.09195	{'C': 6, 'gamma': 0.09195}	5	0.925578	0.015742
6_0.08487	{'C': 6, 'gamma': 0.08487}	5	0.925578	0.016144
3_0.09018	{'C': 3, 'gamma': 0.09018}	5	0.925578	0.015226
10_0.09195	{'C': 10, 'gamma': 0.09195}	5	0.925578	0.015742
15_0.08487	{'C': 15, 'gamma': 0.08487}	5	0.925578	0.016144
15_0.09018	{'C': 15, 'gamma': 0.09018}	12	0.925178	0.016153
6_0.09018	{'C': 6, 'gamma': 0.09018}	12	0.925178	0.016153
10_0.09018	{'C': 10, 'gamma': 0.09018}	12	0.925178	0.016153
6_0.08664	{'C': 6, 'gamma': 0.08664}	15	0.925178	0.016496
15_0.08664	{'C': 15, 'gamma': 0.08664}	15	0.925178	0.016496
10_0.08664	{'C': 10, 'gamma': 0.08664}	15	0.925178	0.016496
15_0.08841	{'C': 15, 'gamma': 0.08841}	18	0.924778	0.016879
6_0.08841	{'C': 6, 'gamma': 0.08841}	18	0.924778	0.016879
10_0.08841	{'C': 10, 'gamma': 0.08841}	18	0.924778	0.016879
1_0.09195	{'C': 1, 'gamma': 0.09195}	21	0.914372	0.012449
1_0.08487	{'C': 1, 'gamma': 0.08487}	22	0.913973	0.014164
1_0.08841	{'C': 1, 'gamma': 0.08841}	23	0.913573	0.013192
1_0.08664	{'C': 1, 'gamma': 0.08664}	23	0.913573	0.014016
1_0.09018	{'C': 1, 'gamma': 0.09018}	25	0.913572	0.012055



Gambar 29. Hasil Confusion Matrix SVM-SIFT Skenario 15

Classification report result:				
	precision	recall	f1-score	support
ba	0.95	0.90	0.93	21
ca	1.00	0.96	0.98	26
da	1.00	0.96	0.98	23
dha	0.96	0.93	0.95	28
ga	0.87	1.00	0.93	20
ha	0.75	0.95	0.84	19
ja	1.00	1.00	1.00	13
ka	1.00	1.00	1.00	28
la	0.90	0.76	0.83	25
ma	1.00	0.95	0.98	22
na	0.95	1.00	0.98	20
nga	1.00	1.00	1.00	21
nya	0.91	0.95	0.93	21
pa	0.90	1.00	0.95	18
ra	0.95	0.83	0.89	24
sa	0.96	1.00	0.98	25
ta	1.00	0.92	0.96	25
tha	1.00	1.00	1.00	22
wa	0.91	0.95	0.93	21
ya	1.00	1.00	1.00	19
accuracy			0.95	441
macro avg	0.95	0.95	0.95	441
weighted avg	0.95	0.95	0.95	441

Gambar 30. Hasil Classification Report SVM-SIFT Skenario 15

**LAMPIRAN P. Pengujian Parameter Skenario 16**

**Tabel 16. Hasil Pelatihan SVM-SIFT Skenario 16**

kernel	params	rank_test_score	mean_test_score	std_test_score
15_0.24383	{'C': 15, 'gamma': 0.24383}	1	0.858346	0.019011
10_0.24383	{'C': 10, 'gamma': 0.24383}	1	0.858346	0.019011
15_0.23885	{'C': 15, 'gamma': 0.23885}	3	0.858346	0.017884
10_0.23885	{'C': 10, 'gamma': 0.23885}	3	0.858346	0.017884
6_0.23885	{'C': 6, 'gamma': 0.23885}	5	0.857946	0.016320
6_0.24383	{'C': 6, 'gamma': 0.24383}	5	0.857946	0.017457
15_0.25877	{'C': 15, 'gamma': 0.25877}	7	0.857945	0.017279
10_0.25877	{'C': 10, 'gamma': 0.25877}	7	0.857945	0.017279
15_0.24881	{'C': 15, 'gamma': 0.24881}	9	0.857545	0.018110
10_0.24881	{'C': 10, 'gamma': 0.24881}	9	0.857545	0.018110
10_0.25379	{'C': 10, 'gamma': 0.25379}	11	0.857145	0.017805
6_0.25877	{'C': 6, 'gamma': 0.25877}	11	0.857145	0.016007
15_0.25379	{'C': 15, 'gamma': 0.25379}	11	0.857145	0.017805
6_0.24881	{'C': 6, 'gamma': 0.24881}	14	0.856745	0.016882
3_0.24383	{'C': 3, 'gamma': 0.24383}	14	0.856745	0.016105
3_0.23885	{'C': 3, 'gamma': 0.23885}	16	0.856745	0.015401
3_0.25877	{'C': 3, 'gamma': 0.25877}	17	0.855945	0.015412
3_0.24881	{'C': 3, 'gamma': 0.24881}	17	0.855945	0.015721
6_0.25379	{'C': 6, 'gamma': 0.25379}	17	0.855945	0.017086
3_0.25379	{'C': 3, 'gamma': 0.25379}	20	0.855545	0.015638
1_0.24383	{'C': 1, 'gamma': 0.24383}	21	0.843940	0.021487
1_0.24881	{'C': 1, 'gamma': 0.24881}	22	0.843140	0.020904
1_0.23885	{'C': 1, 'gamma': 0.23885}	22	0.843140	0.021692
1_0.25877	{'C': 1, 'gamma': 0.25877}	24	0.843139	0.020248
1_0.25379	{'C': 1, 'gamma': 0.25379}	24	0.843139	0.020248

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ca	0	21	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0
Actual	da	0	1	21	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	dha	1	0	0	26	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Actual	ga	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ha	0	0	0	0	0	15	0	0	4	0	0	0	0	0	0	0	0	0	0	0
Actual	ja	0	0	0	0	0	0	12	0	1	0	0	0	0	0	0	0	0	0	0	0
	ka	0	0	0	0	0	0	0	26	0	0	2	0	0	0	0	0	0	0	0	0
Actual	la	0	0	0	0	0	3	0	0	19	0	0	0	1	0	0	2	0	0	0	0
	ma	0	0	0	0	0	1	0	0	21	0	0	0	0	0	0	0	0	0	0	0
Actual	nya	0	0	1	0	0	0	3	0	0	16	0	0	0	0	0	0	0	0	0	0
	na	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0
Actual	pa	1	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	1	0	0	0
	ra	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Actual	sa	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0
	ta	0	0	0	0	0	2	1	0	1	1	0	0	0	0	0	0	19	0	1	0
Actual	tha	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0
	wa	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0
Actual	ya	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	18	0
Predicted		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya

Gambar 31. Hasil Confusion Matrix SVM-SIFT Skenario 16

		Classification report result:				
		precision	recall	f1-score	support	
	ba	0.91	1.00	0.95	21	
	ca	0.91	0.81	0.86	26	
	da	0.91	0.91	0.91	23	
	dha	0.96	0.93	0.95	28	
	ga	0.95	1.00	0.98	20	
	ha	0.75	0.79	0.77	19	
	ja	0.75	0.92	0.83	13	
	ka	0.90	0.93	0.91	28	
	la	0.76	0.76	0.76	25	
	ma	0.91	0.95	0.93	22	
	na	0.84	0.80	0.82	20	
	nga	1.00	1.00	1.00	21	
	nya	1.00	0.95	0.98	21	
	pa	0.81	0.94	0.87	18	
	ra	1.00	0.96	0.98	24	
	sa	0.88	0.92	0.90	25	
	ta	0.90	0.76	0.83	25	
	tha	1.00	1.00	1.00	22	
	wa	0.90	0.86	0.88	21	
	ya	1.00	0.95	0.97	19	
		accuracy		0.90	441	
		macro avg	0.90	0.91	441	
		weighted avg	0.91	0.90	441	

Gambar 32. Hasil Classification Report SVM-SIFT Skenario 16

**LAMPIRAN Q. Pengujian Parameter Skenario 17**

**Tabel 17. Hasil Pelatihan SVM-SIFT Skenario 17**

kernel	params	rank_test_score	mean_test_score	std_test_score
6_0.19929	{'C': 6, 'gamma': 0.19929}	1	0.879959	0.013028
15_0.19929	{'C': 15, 'gamma': 0.19929}	1	0.879959	0.013028
15_0.19131	{'C': 15, 'gamma': 0.19131}	1	0.879959	0.012399
10_0.19929	{'C': 10, 'gamma': 0.19929}	1	0.879959	0.013028
10_0.19131	{'C': 10, 'gamma': 0.19131}	1	0.879959	0.012399
6_0.19131	{'C': 6, 'gamma': 0.19131}	1	0.879959	0.012399
3_0.19131	{'C': 3, 'gamma': 0.19131}	7	0.879958	0.011190
15_0.1953	{'C': 15, 'gamma': 0.1953}	8	0.879559	0.013533
10_0.20727	{'C': 10, 'gamma': 0.20727}	8	0.879559	0.013533
10_0.1953	{'C': 10, 'gamma': 0.1953}	8	0.879559	0.013533
6_0.20727	{'C': 6, 'gamma': 0.20727}	8	0.879559	0.013533
15_0.20727	{'C': 15, 'gamma': 0.20727}	8	0.879559	0.013533
6_0.1953	{'C': 6, 'gamma': 0.1953}	8	0.879559	0.013533
15_0.20328	{'C': 15, 'gamma': 0.20328}	14	0.879159	0.013951
6_0.20328	{'C': 6, 'gamma': 0.20328}	14	0.879159	0.013951
10_0.20328	{'C': 10, 'gamma': 0.20328}	14	0.879159	0.013951
3_0.20727	{'C': 3, 'gamma': 0.20727}	17	0.879158	0.013608
3_0.19929	{'C': 3, 'gamma': 0.19929}	17	0.879158	0.013006
3_0.1953	{'C': 3, 'gamma': 0.1953}	19	0.879158	0.012121
3_0.20328	{'C': 3, 'gamma': 0.20328}	20	0.878758	0.014069
1_0.20727	{'C': 1, 'gamma': 0.20727}	21	0.868759	0.016204
1_0.20328	{'C': 1, 'gamma': 0.20328}	22	0.867959	0.016222
1_0.1953	{'C': 1, 'gamma': 0.1953}	23	0.867958	0.015776
1_0.19131	{'C': 1, 'gamma': 0.19131}	24	0.867558	0.015693
1_0.19929	{'C': 1, 'gamma': 0.19929}	25	0.867158	0.016055

		Confusion Matrix Plot																			
		Confusion Matrix Plot																			
Actual	Predicted	ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
		19	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
ba	ba	19	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0
ca	ba	0	23	0	0	0	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0
da	ca	0	0	21	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
dha	da	1	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ga	da	0	0	0	0	19	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
ha	ga	0	0	0	0	0	17	0	0	1	0	0	0	0	0	0	0	1	0	0	0
ja	ha	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	1
ka	ja	0	0	0	0	0	0	0	26	0	0	2	0	0	0	0	0	0	0	0	0
la	ka	0	0	0	0	0	9	0	0	13	0	0	0	1	0	0	2	0	0	0	0
ma	la	0	1	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	1	0
nya	ma	0	0	1	0	0	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0
pa	nya	0	0	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0
ra	pa	0	0	0	0	1	0	0	0	0	0	0	0	16	0	0	0	0	0	1	0
sa	ra	0	0	0	0	1	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0
ta	sa	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	21	0	1	0
tha	ta	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0	0
wa	tha	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	0	18	0
ya	wa	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	18

Gambar 33. Hasil Confusion Matrix SVM-SIFT Skenario 17

Classification report result:				
	precision	recall	f1-score	support
ba	0.95	0.90	0.93	21
ca	0.96	0.88	0.92	26
da	0.95	0.91	0.93	23
dha	0.96	0.96	0.96	28
ga	0.95	0.95	0.95	20
ha	0.59	0.89	0.71	19
ja	0.80	0.92	0.86	13
ka	0.96	0.93	0.95	28
la	0.87	0.52	0.65	25
ma	1.00	0.91	0.95	22
na	0.86	0.95	0.90	20
nga	0.91	1.00	0.95	21
nya	1.00	1.00	1.00	21
pa	0.84	0.89	0.86	18
ra	1.00	0.96	0.98	24
sa	0.93	1.00	0.96	25
ta	0.88	0.84	0.86	25
tha	1.00	1.00	1.00	22
wa	0.86	0.86	0.86	21
ya	0.95	0.95	0.95	19
accuracy			0.91	441
macro avg	0.91	0.91	0.91	441
weighted avg	0.92	0.91	0.91	441

Gambar 34. Hasil Classification Report SVM-SIFT Skenario 17

**LAMPIRAN R. Pengujian Parameter Skenario 18**

**Tabel 18. Hasil Pelatihan SVM-SIFT Skenario 18**

kernel	params	rank_test_score	mean_test_score	std_test_score
3_0.11659	{'C': 3, 'gamma': 0.11659}	1	0.891962	0.007634
15_0.11659	{'C': 15, 'gamma': 0.11659}	2	0.891562	0.010018
10_0.11659	{'C': 10, 'gamma': 0.11659}	2	0.891562	0.010018
3_0.12135	{'C': 3, 'gamma': 0.12135}	4	0.891562	0.007674
3_0.11421	{'C': 3, 'gamma': 0.11421}	4	0.891562	0.008080
3_0.11897	{'C': 3, 'gamma': 0.11897}	4	0.891562	0.007674
10_0.11421	{'C': 10, 'gamma': 0.11421}	7	0.891162	0.010800
15_0.11421	{'C': 15, 'gamma': 0.11421}	7	0.891162	0.010800
6_0.11659	{'C': 6, 'gamma': 0.11659}	7	0.891162	0.009872
10_0.12135	{'C': 10, 'gamma': 0.12135}	10	0.890762	0.010499
6_0.12135	{'C': 6, 'gamma': 0.12135}	10	0.890762	0.010499
15_0.12135	{'C': 15, 'gamma': 0.12135}	10	0.890762	0.010499
6_0.11897	{'C': 6, 'gamma': 0.11897}	10	0.890762	0.009789
15_0.11897	{'C': 15, 'gamma': 0.11897}	10	0.890762	0.010499
6_0.11421	{'C': 6, 'gamma': 0.11421}	10	0.890762	0.010575
10_0.11897	{'C': 10, 'gamma': 0.11897}	10	0.890762	0.010499
3_0.12373	{'C': 3, 'gamma': 0.12373}	17	0.890762	0.007997
10_0.12373	{'C': 10, 'gamma': 0.12373}	18	0.889962	0.010373
15_0.12373	{'C': 15, 'gamma': 0.12373}	18	0.889962	0.010373
6_0.12373	{'C': 6, 'gamma': 0.12373}	18	0.889962	0.010373
1_0.11659	{'C': 1, 'gamma': 0.11659}	21	0.885158	0.011037
1_0.12135	{'C': 1, 'gamma': 0.12135}	22	0.884758	0.011456
1_0.11421	{'C': 1, 'gamma': 0.11421}	22	0.884758	0.012134
1_0.11897	{'C': 1, 'gamma': 0.11897}	24	0.884758	0.010742
1_0.12373	{'C': 1, 'gamma': 0.12373}	25	0.884358	0.011157

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	20	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	ca	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
Actual	da	0	0	22	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	dha	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Actual	ga	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	ha	0	0	0	0	0	17	0	0	2	0	0	0	0	0	0	0	0	0	0	0
Actual	ja	0	1	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	1	0
	ka	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
Actual	la	0	0	0	0	0	4	0	0	18	0	0	0	0	1	0	0	2	0	0	0
	ma	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	1	0
Actual	na	0	0	1	0	0	0	1	0	0	17	0	0	0	0	0	0	0	1	0	0
	nya	0	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0
Actual	pa	1	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0
	ra	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0
Actual	sa	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0
	ta	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	22	0	1
Actual	tha	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	0	0
	wa	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	18	0
Actual	ya	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	17
Predicted		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya

Gambar 35. Hasil Confusion Matrix SVM-SIFT Skenario 18

		Classification report result:				
		precision	recall	f1-score	support	
	ba	0.91	0.95	0.93	21	
	ca	0.92	0.92	0.92	26	
	da	0.96	0.96	0.96	23	
	dha	1.00	1.00	1.00	28	
	ga	1.00	1.00	1.00	20	
	ha	0.74	0.89	0.81	19	
	ja	1.00	0.85	0.92	13	
	ka	0.97	1.00	0.98	28	
	la	0.86	0.72	0.78	25	
	ma	1.00	0.95	0.98	22	
	na	1.00	0.85	0.92	20	
	nga	1.00	1.00	1.00	21	
	nya	0.95	0.95	0.95	21	
	pa	0.78	1.00	0.88	18	
	ra	1.00	1.00	1.00	24	
	sa	0.92	0.96	0.94	25	
	ta	0.92	0.88	0.90	25	
	tha	0.95	0.95	0.95	22	
	wa	0.82	0.86	0.84	21	
	ya	1.00	0.89	0.94	19	
		accuracy		0.93	441	
		macro avg	0.93	0.93	441	
		weighted avg	0.94	0.93	441	

Gambar 36. Hasil Classification Report SVM-SIFT Skenario 18

**LAMPIRAN S. Pengujian Parameter Skenario 19**

**Tabel 19. Hasil Pelatihan SVM-SIFT Skenario 19**

kernel	params	rank_test_score	mean_test_score	std_test_score
10_0.09026	{'C': 10, 'gamma': 0.09026}	1	0.909172	0.017152
15_0.09026	{'C': 15, 'gamma': 0.09026}	1	0.909172	0.017152
6_0.09026	{'C': 6, 'gamma': 0.09026}	1	0.909172	0.017152
15_0.09214	{'C': 15, 'gamma': 0.09214}	4	0.908372	0.018006
6_0.09214	{'C': 6, 'gamma': 0.09214}	4	0.908372	0.018006
3_0.09026	{'C': 3, 'gamma': 0.09026}	4	0.908372	0.016812
10_0.09214	{'C': 10, 'gamma': 0.09214}	4	0.908372	0.018006
3_0.09214	{'C': 3, 'gamma': 0.09214}	8	0.907572	0.017646
6_0.09402	{'C': 6, 'gamma': 0.09402}	9	0.907172	0.018146
10_0.09402	{'C': 10, 'gamma': 0.09402}	9	0.907172	0.018146
15_0.09402	{'C': 15, 'gamma': 0.09402}	9	0.907172	0.018146
3_0.09402	{'C': 3, 'gamma': 0.09402}	12	0.906772	0.018234
3_0.0959	{'C': 3, 'gamma': 0.0959}	12	0.906772	0.018234
15_0.0959	{'C': 15, 'gamma': 0.0959}	14	0.906771	0.017656
6_0.0959	{'C': 6, 'gamma': 0.0959}	14	0.906771	0.017656
10_0.0959	{'C': 10, 'gamma': 0.0959}	14	0.906771	0.017656
3_0.09778	{'C': 3, 'gamma': 0.09778}	17	0.905973	0.019188
10_0.09778	{'C': 10, 'gamma': 0.09778}	18	0.905972	0.018597
15_0.09778	{'C': 15, 'gamma': 0.09778}	18	0.905972	0.018597
6_0.09778	{'C': 6, 'gamma': 0.09778}	18	0.905972	0.018597
1_0.09402	{'C': 1, 'gamma': 0.09402}	21	0.896364	0.017744
1_0.09214	{'C': 1, 'gamma': 0.09214}	21	0.896364	0.016866
1_0.09026	{'C': 1, 'gamma': 0.09026}	23	0.895964	0.017315
1_0.09778	{'C': 1, 'gamma': 0.09778}	24	0.895164	0.017570
1_0.0959	{'C': 1, 'gamma': 0.0959}	25	0.895164	0.018501

		Confusion Matrix Plot																			
		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya
Actual	ba	20	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	ca	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0
Actual	da	0	0	21	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0
	dha	1	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Actual	ga	0	0	0	0	19	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	ha	0	0	0	0	0	15	0	0	3	0	0	0	0	0	0	0	1	0	0	0
Actual	ja	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
	ka	0	0	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	0	0
Actual	la	0	0	0	0	0	3	0	0	19	0	0	0	0	2	0	0	0	0	1	0
	ma	0	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	1	0
Actual	nya	0	0	0	0	0	0	0	0	2	0	0	0	0	18	0	0	0	0	0	0
	na	0	2	0	0	0	0	0	0	0	0	0	0	0	19	0	0	0	0	0	0
Actual	pa	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	1
	ra	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	0	0	0	0	0
Actual	sa	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0
	ta	0	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	22	0	0	0
Actual	tha	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	22	0	0
	wa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	1	0
Actual	ya	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	17
Predicted		ba	ca	da	dha	ga	ha	ja	ka	la	ma	na	nga	nya	pa	ra	sa	ta	tha	wa	ya

Gambar 37. Hasil Confusion Matrix SVM-SIFT Skenario 19

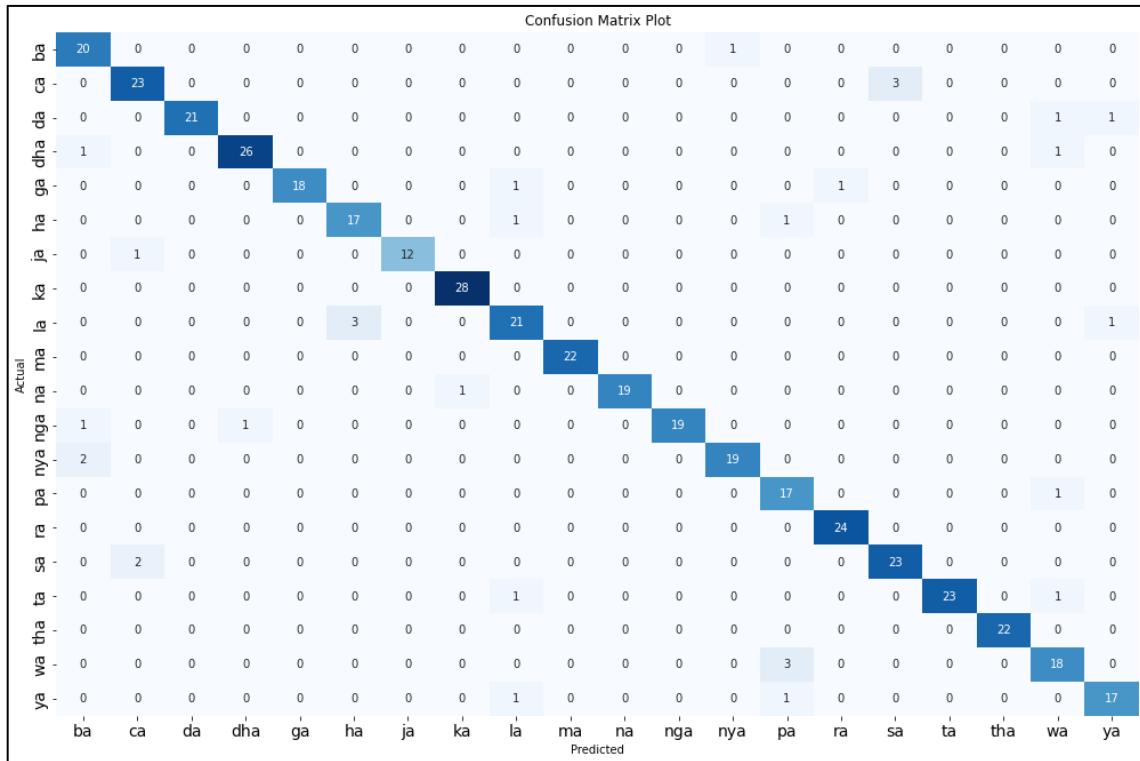
Classification report result:				
	precision	recall	f1-score	support
ba	0.87	0.95	0.91	21
ca	0.92	0.92	0.92	26
da	1.00	0.91	0.95	23
dha	1.00	0.96	0.98	28
ga	1.00	0.95	0.97	20
ha	0.75	0.79	0.77	19
ja	1.00	1.00	1.00	13
ka	0.93	1.00	0.97	28
la	0.76	0.76	0.76	25
ma	1.00	0.95	0.98	22
na	0.95	0.90	0.92	20
nga	1.00	1.00	1.00	21
nya	0.95	0.90	0.93	21
pa	0.77	0.94	0.85	18
ra	0.96	1.00	0.98	24
sa	0.92	0.92	0.92	25
ta	0.96	0.88	0.92	25
tha	1.00	1.00	1.00	22
wa	0.90	0.86	0.88	21
ya	0.89	0.89	0.89	19
accuracy			0.93	441
macro avg	0.93	0.93	0.93	441
weighted avg	0.93	0.93	0.93	441

Gambar 38. Hasil Classification Report SVM-SIFT Skenario 19

**LAMPIRAN T. Pengujian Parameter Skenario 20**

**Tabel 20. Hasil Pelatihan SVM-SIFT Skenario 20**

kernel	params	rank_test_score	mean_test_score	std_test_score
15_0.07609	{'C': 15, 'gamma': 0.07609}	1	0.904770	0.012054
6_0.07609	{'C': 6, 'gamma': 0.07609}	1	0.904770	0.012054
10_0.07609	{'C': 10, 'gamma': 0.07609}	1	0.904770	0.012054
10_0.07767	{'C': 10, 'gamma': 0.07767}	4	0.904770	0.011305
15_0.07767	{'C': 15, 'gamma': 0.07767}	4	0.904770	0.011305
6_0.07767	{'C': 6, 'gamma': 0.07767}	4	0.904770	0.011305
10_0.07925	{'C': 10, 'gamma': 0.07925}	7	0.903970	0.012079
6_0.07925	{'C': 6, 'gamma': 0.07925}	7	0.903970	0.012079
15_0.07925	{'C': 15, 'gamma': 0.07925}	7	0.903970	0.012079
3_0.07609	{'C': 3, 'gamma': 0.07609}	10	0.903970	0.011882
3_0.07767	{'C': 3, 'gamma': 0.07767}	11	0.903969	0.011193
3_0.07925	{'C': 3, 'gamma': 0.07925}	12	0.903170	0.011853
10_0.08083	{'C': 10, 'gamma': 0.08083}	13	0.901969	0.011813
15_0.08083	{'C': 15, 'gamma': 0.08083}	13	0.901969	0.011813
6_0.08083	{'C': 6, 'gamma': 0.08083}	13	0.901969	0.011813
3_0.08083	{'C': 3, 'gamma': 0.08083}	16	0.901568	0.011147
10_0.08241	{'C': 10, 'gamma': 0.08241}	17	0.900368	0.012041
15_0.08241	{'C': 15, 'gamma': 0.08241}	17	0.900368	0.012041
3_0.08241	{'C': 3, 'gamma': 0.08241}	17	0.900368	0.012041
6_0.08241	{'C': 6, 'gamma': 0.08241}	17	0.900368	0.012041
1_0.07609	{'C': 1, 'gamma': 0.07609}	21	0.893165	0.010180
1_0.07925	{'C': 1, 'gamma': 0.07925}	22	0.892362	0.008833
1_0.07767	{'C': 1, 'gamma': 0.07767}	23	0.891963	0.009145
1_0.08083	{'C': 1, 'gamma': 0.08083}	24	0.890762	0.009870
1_0.08241	{'C': 1, 'gamma': 0.08241}	25	0.889562	0.010555



Gambar 39. Hasil Confusion Matrix SVM-SIFT Skenario 20

Classification report result:				
	precision	recall	f1-score	support
ba	0.83	0.95	0.89	21
ca	0.88	0.88	0.88	26
da	1.00	0.91	0.95	23
dha	0.96	0.93	0.95	28
ga	1.00	0.90	0.95	20
ha	0.85	0.89	0.87	19
ja	1.00	0.92	0.96	13
ka	0.97	1.00	0.98	28
la	0.84	0.84	0.84	25
ma	1.00	1.00	1.00	22
na	1.00	0.95	0.97	20
nga	1.00	0.90	0.95	21
nya	0.95	0.90	0.93	21
pa	0.77	0.94	0.85	18
ra	0.96	1.00	0.98	24
sa	0.88	0.92	0.90	25
ta	1.00	0.92	0.96	25
tha	1.00	1.00	1.00	22
wa	0.82	0.86	0.84	21
ya	0.89	0.89	0.89	19
accuracy			0.93	441
macro avg		0.93	0.93	441
weighted avg		0.93	0.93	441

Gambar 40. Hasil Classification Report SVM-SIFT Skenario 20