IDENTIFIKASI AKSARA JAWA PADA NASKAH KUNO DENGAN METODE CNN

Disusun Oleh:

N a m a : Arif Sulaksana Putra

NIM : 16523228

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2020

HALAMAN PENGESAHAN DOSEN PEMBIMBING

IDENTIFIKASI AKSARA JAWA PADA NASKAH KUNO DENGAN METODE CNN

TUGAS AKHIR

Disusun Oleh:

N a m a : Arif Sulaksana Putra

NIM : 16523228

Yogyakarta, 13 Juli 2020

Pembimbing,

(Izzati Muhimmah , S.T., M.Sc. Ph.D.)

HALAMAN PENGESAHAN DOSEN PENGUJI

IDENTIFIKASI AKSARA JAWA PADA NASKAH KUNO DENGAN METODE CNN

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 22 Agustus 2020

Tim Penguji

Izzati Muhimmah, S.T., M.Sc., Ph.D.

Anggota 1

Taufiq Hidayat, S.T, M.C.S

Anggota 2

Arrie Kurniawardhani, S.Si., M.Kom.

Mengetahui,
Ketua Program Studi Informatika – Program Sarjana
Fakultas Teknologi Industri
Universitas Islam Indonesia

(Dr. Raden Teduh Dirgahayu, S.T., M.Sc.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Arif Sulaksana Putra

NIM : 16523228

Tugas akhir dengan judul:

IDENTIFIKASI AKSARA JAWA PADA NASKAH KUNO DENGAN METODE CNN

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apa pun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 13 Juli 2020

(Arif Sulaksana Putra)

HALAMAN PERSEMBAHAN

Alhamdulillahirobbil'alamin dengan mengucapkan rasa syukur ke hadirat Allah SWT yang telah melimpahkan seluruh rahmat dan karunia-Nya saya akhirnya dapat menyelesaikan Tugas Akhir Program Studi Informatika Fakultas Teknologi Industri Universitas Islam Indonesia dengan baik, maka saya persembahkan maha karya ini kepada :

1. Ibu Tuti Sulistyowati yang telah menjadi sosok Ibu yang menyayangi dan penyabar.
2. Bapak Sigit Triwahyudi yang telah menjadi sosok Ayah yang tak mengenal putus asa dan telah mendidik saya dengan sepenuh hati, tentu saja yang saya sayangi dan cintai.
3. Saudara Harma Putra Nugraha yang telah menjadi kakak laki-laki yang baik hati dan terus mendukung.
4. Keluarga Besar HS dan Mawardi yang saya sangat kagumi kepada mereka semua.
5. Ibu Izzati Muhimmah, S.T., M.Sc. Ph.D. yang telah menjadi dosen pembimbing pada tugas akhir ini dengan kesabaran dan sosok keibuan beliau tugas akhir ini bisa berakhir.
6. Semua mahasiswa, dosen, dan staff Informatika UII sebagai tempat saya bernaung selama kurang lebih 4 Tahun ini.

HALAMAN MOTO

“Kebahagiaan itu bergantung pada dirimu sendiri”
(Aristoteles)

“Bermimpilah seakan kau akan hidup selamanya.
Hiduplah seakan kau akan mati hari ini”
(James Dean)

“Yakinlah kau bisa dan kau sudah separuh jalan menuju ke sana”
(Theodore Roosevelt)

“Banyak kegagalan dalam hidup ini dikarenakan orang-orang tidak menyadari betapa dekatnya mereka dengan keberhasilan saat mereka menyerah.”
(Thomas Alva Edison)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakaatuh

Puja dan puji syukur saya ucapkan kepada Allah SWT yang telah memberikan rahmat serta hidayah-Nya sehingga saya dapat menyelesaikan tugas akhir ini yang berjudul "**IDENTIFIKASI AKSARA JAWA PADA NASKAH KUNO DENGAN METODE CNN**". Tugas akhir ini saya susun guna memenuhi syarat untuk memperoleh gelar sarjana di Program Studi Informatika Universitas Islam Indonesia dan juga untuk ke depannya supaya budaya yang telah diwarisi nenek moyang akan senantiasa dilestarikan dan mempermudah setiap orang untuk membaca aksara Jawa dalam bentuk digital.

Dalam penyusunan tugas akhir ini, saya sepenuhnya menyadari bahwa tugas akhir ini tidak dapat terselesaikan tanpa adanya bimbingan, dorongan dan bantuan dari berbagai pihak yang bersifat langsung maupun tidak langsung. Dalam kesempatan ini izinkan saya mengucapkan terima kasih kepada:

1. Allah SWT, berkat limpahan rahmat, karunia dan hidayah-Nya sehingga saya mampu menyusun tugas akhir ini dengan lancar.
2. Ibu yang sangat saya cintai yang sangat menjadi motivasi untuk menyelesaikan tugas akhir ini.
3. Bapak yang dengan kegigihannya telah mendidik dan selalu memeberi pelajaran kehidupan.
4. Kakak laki-laki yang sangat mendukung.
5. Keluarga besar Hartono Sutantini dan Mawardi yang saya banggakan.
6. Ibu Izzati Muhimmah , S.T., M.Sc. Ph.D. sebagai dosen pembimbing tugas akhir saya yang telah sabar membantu dalam penyusuanan tugas akhir ini.
7. Keluarga besar Informatika Universitas Islam Indonesia yang telah turut menyumbangkan pengalaman yang sangat berharga bagi saya.
8. Keluarga besar HexaDecima yang saya sayangi dan saya banggakan karena telah menjadi keluarga di lingkungan perantauan saya.
9. Grup 'NIJISANJI ID Fan Discord' selalu menemani saya.
10. Para Virtual Livers 'NIJISANJI ID' yang telah menemani saya dan selalu memotivasi.

11. Serta teman-teman semua yang tidak bisa saya ucapkan satu-satu persatu terima kasih atas dukungannya selama ini.

Saya menyadari bahwa dalam penyusunan tugas akhir ini masih banyak kekurangan dan kesalahan yang disengaja maupun tidak disengaja maka, izinkan saya mohon maaf sebesar-besarnya atas tindakan tersebut. Semoga dalam kekurangan atau pun kesalahan bisa menjadikan inspirasi bagi generasi-generasi berikutnya, sekian kata pengantar ini dibuat. Bagi siapa pun yang membaca tugas akhir ini semoga mendapatkan manfaat bagi semua pihak.

Wassalamu'alaikum warahmatullahi wabarakatuh.

Yogyakarta, 22 Agustus 2020

(Arif Sulaksana Putra)

SARI

Sudah banyak naskah kuno dari Yogyakarta yang telah didigitalisasikan. Kebanyakan dari naskah kuno tersebut ditulis menggunakan aksara Jawa, sehingga banyak orang yang tidak dapat membaca naskah tersebut, oleh karena itu penelitian ini akan berfokus pada pengenalan aksara Jawa. Penelitian ini bertujuan untuk mengidentifikasi tulisan aksara Jawa pada naskah kuno dan merepresentasikannya menjadi huruf Latin. Penelitian ini dimulai dengan mendapatkan data citra naskah kuno aksara Jawa melalui British Library lalu citra naskah tersebut akan diproses dengan menggunakan teknik pengolahan citra atau *image processing* sehingga tulisan aksara Jawa tersebut dapat dikenali oleh komputer dan mengubah aksara Jawa tersebut menjadi huruf Latin. Penelitian ini akan menggunakan model klasifikasi *Convolutional Neural Network* (CNN) dan bahasa Python untuk bahasa pemrograman. Hasil dari klasifikasi akan diuji dengan metode *Cohen's Kappa* dan mendapatkan hasil nilai *Kappa* sebesar 0.86 dan termasuk mendapat *level of agreement* yang kuat (*strong*).

Kata kunci : aksara, CNN, cohen's kappa, identifikasi, jawa, naskah

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Di Jawa, salah satunya di Daerah Istimewa Yogyakarta, terdapat banyak warisan-warisan budaya yang penting dalam bentuk dokumen atau naskah lama. Naskah-naskah tersebut merupakan warisan budaya yang perlu dilestarikan karena merupakan salah satu bukti nyata bahwa Indonesia mempunyai budaya.

Saat ini, sudah ada 75 naskah kuno dari Yogyakarta yang dimiliki oleh British Library dan 35 di antaranya sudah didigitalisasikan dan akan diserahkan kepada Badan Arsip dan Perpustakaan DIY serta Perpusnas di Jakarta. Naskah-naskah tersebut juga dapat diakses *online* secara gratis di situs British Library (Gallop, 2018). Isi suatu naskah atau dokumen tertulis seringkali juga memberikan informasi aspek budaya bangsa dari masyarakat yang bersangkutan. Naskah-naskah kuno di Yogyakarta kebanyakan ditulis menggunakan karakter atau aksara Jawa. Tujuan dari penelitian ini untuk merepresentasikan kembali dokumen tersebut dengan karakter Latin tanpa mengubah atau menghilangkan makna asli dari naskah tersebut.

Informasi yang dapat disampaikan dari naskah kuno meliputi bidang filsafat, kehidupan agama, kepercayaan, masalah-masalah teknis seperti pembangunan tempat tinggal, pengadaan tanah ladang, pengajaran berbagai jenis keahlian, dan keterampilan, serta hal hal lain yang menyangkut keperluan kehidupan bangsa bersangkutan secara menyeluruh (Saraswati, 2011).

Penelitian ini dilakukan dengan mengambil citra dokumen yang ditulis dengan aksara Jawa pada British Library. Citra dokumen tersebut akan diproses dengan menggunakan teknik pengolahan citra atau *image processing* sehingga tulisan aksara Jawa tersebut dapat dikenali oleh komputer dan mengubah aksara Jawa tersebut menjadi huruf Latin. Penelitian ini akan menggunakan model klasifikasi *Convolutional Neural Network* (CNN) dan bahasa Python untuk bahasa pemrograman.

1.2 Rumusan Masalah

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

1. Apakah sistem dapat mendeteksi tulisan aksara Jawa pada naskah lama dan bagaimana cara membangun sistem pendeteksiannya?
2. Apakah sistem dapat mengidentifikasi tulisan aksara Jawa yang telah terdeteksi pada naskah lama dan bagaimana cara membangun sistem identifikasi dengan CNN?
3. Bagaimana performa sistem ?

1.3 Batasan Masalah

Batasan masalah untuk penelitian ini adalah:

1. Hanya menggunakan satu naskah yang ditulis dengan aksara Jawa dan diambil dari British Library.
2. Hanya mengidentifikasi empat baris pertama perhalaman dari citra naskah.
3. Hanya mengklasifikasi 24 kategori aksara Jawa yaitu 20 huruf *Dentawyanjana* dan 4 huruf *sandhangan* yaitu *wulu, pepet, cecak,* dan *layar*.

1.4 Tujuan

Tujuan dari penelitian ini adalah untuk mengidentifikasi tulisan aksara Jawa pada naskah lama dan merepresentasikannya menjadi huruf Latin.

1.5 Manfaat

Manfaat dari penelitian ini adalah:

1. Mempermudah orang yang ingin melanjutkan penelitian tentang identifikasi aksara Jawa.
2. Melestarikan budaya.
3. Mengenalkan budaya.

1.6 Metodologi

Langkah-langkah yang diterapkan dalam mengembangkan sistem adalah:

a. Tahap pengumpulan data

   i. Studi Pustaka

      Studi Pustaka dilakukan untuk mencari data-data dan informasi-informasi dari buku, artikel, dan jurnal yang dapat menjadi referensi tentang pengolahan citra, segmentasi karakter dan klasifikasi karakter.

   ii. Pengambilan Citra Dokumen

      Citra dokumen diambil dari situs British Library.

b. Pembuatan Sistem

   Metode pembuatan sistem yang akan digunakan terdiri dari:

   i. Analisis Kebutuhan

      Pada tahap ini akan dilakukan pemodelan terhadap kebutuhan sistem untuk identifikasi tulisan aksara Jawa pada naskah lama mulai dari *input*, proses dan *output* dari sistem yang akan dibuat.

   ii. Perancangan

      Tahap ini akan menjelaskan tentang perancangan bentuk sistem yang akan dibuat berdasarkan analisis kebutuhan sistem, seperti *flowchart*.

   iii. Implementasi

      Tahap ini akan mengimplementasikan hasil pada sistem berdasarkan analisis kebutuhan dan perancangan yang sudah dilakukan sebelumnya.

   iv. Pengujian

      Tahap ini akan dilakukan pengujian terhadap aplikasi yang telah selesai dibuat, dengan cara menggunakan aplikasi dengan masukkan berupa citra dokumen naskah lama. Tahap ini dilakukan agar dapat mengetahui implementasi sistem sudah sesuai dengan tujuan yang diharapkan atau masih terdapat kesalahan.

BAB II

LANDASAN TEORI

2.1 Aksara Jawa

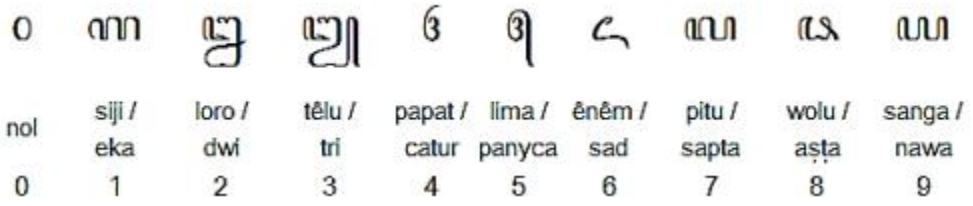
Aksara Jawa merupakan karakter Jawa kuno yang sudah digunakan sejak abad ke-17 selama masa kerajaan Mataram. Sejarah Indonesia kebanyakan ditulis menggunakan karakter ini yang tertulis pada batu-batuan. Beberapa orang lokal masih menggunakan karakter ini sebagai contoh untuk nama tempat, tempat turis,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

Dibandingkan dengan karakter Latin, aksara Jawa mempunyai struktur dan bentuk yang berbeda. Basis dari aksara Jawa disebut dengan *Carakan*, yang mana terdiri dari 20 suku kata yang dinamakan dengan *Dentawyanjana* yang dapat dilihat pada Gambar 2.1.

ha	na	ca	ra	ka
ꦲ	ꦤ	ꦕ	ꦫ	ꦏ
da	ta	sa	wa	la
ꦢ	ꦠ	ꦱ	ꦮ	ꦭ
pa	dha	ja	ya	nya
ꦥ	ꦝ	ꦗ	ꦪ	ꦚ
ma	ga	ba	tha	nga
ꦩ	ꦒ	ꦧ	ꦛ	ꦔ

Gambar 2.1 Aksara Jawa *Dentawyanjana*

Angka pada aksara Jawa dapat dilihat pada Gambar 2.2



Gambar 2.2 Aksara Jawa Angka

Karakter *Sandhangan* adalah karakter spesial yang biasanya digunakan sebagai karakter pelengkap, vokal dan konsonan yang biasanya digunakan dalam bahasa sehari-hari. *Sandhangan* dapat dilihat pada Gambar 2.3.

Sandhangan name	Java character	Description
Pepet		*Vowel ê*
Taling		*Vowel é*
Wulu		*Vowel i*
Taling tarung		*Vowel o*
Suku		*Vowel u*

Gambar 2.3 Aksara Jawa *Sandhangan*

2.2 Citra *Digital*

Secara umum, pengolahan citra *digital* menunjuk pada pemrosesan gambar dua dimensi menggunakan komputer. Citra *digital* merupakan sebuah larik (*array*) yang berisi nilai-nilai *real* maupun kompleks yang direpresentasikan dengan deretan *bit* tertentu (Putra, 2010).

Suatu citra dapat didefinisikan sebagai fungsi matriks *f(x,y)* berukuran M baris dan N kolom, dengan nilai *x* dan *y* adalah koordinat spasial, dan amplitudo *f* di titik koordinat *(x,y)* dinamakan intensitas atau tingkat keabuan dari citra tersebut. Apabila nilai *x* dan *y,* dan nilai amplitudo *f* secara keseluruhan berhingga (*finite)* dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra *digital* (Putra, 2010).

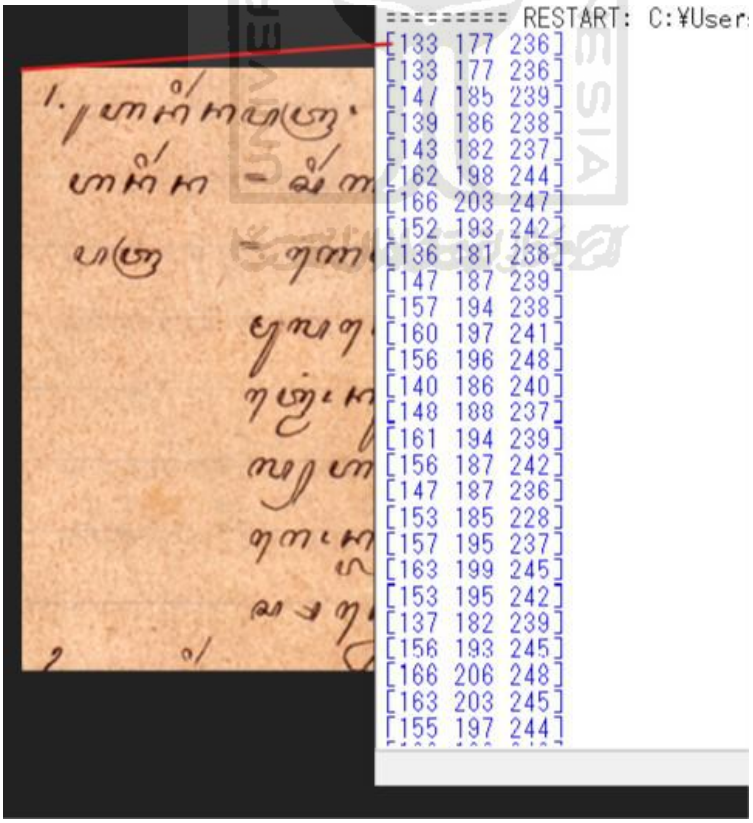
Citra *digital* dapat ditulis dalam bentuk matriks seperti Gambar 2.4.

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Gambar 2.4 Penggambaran Matriks Citra Digital

2.2.1 Citra Warna (Citra RGB)

Citra warna adalah citra yang masing-masing *pixel* memiliki warna tertentu, warna tersebut adalah warna merah (*Red*), hijau (*Green*) dan biru (*Blue*). Jika masing-masing warna memiliki *range* 0 - 255, maka totalnya adalah 255^3 = 16.581.375 (16 K) variasi warna berbeda pada citra di mana variasi warna ini cukup untuk gambar apa pun. Citra warna ini terdiri dari tiga matriks yang mewakili nilai-nilai merah, hijau dan biru untuk setiap *pixel*nya (Kusumanto dkk, 2011), seperti yang ditunjukkan pada Gambar 2.5.



Gambar 2.5 Citra Warna dan Intensitas Warnanya

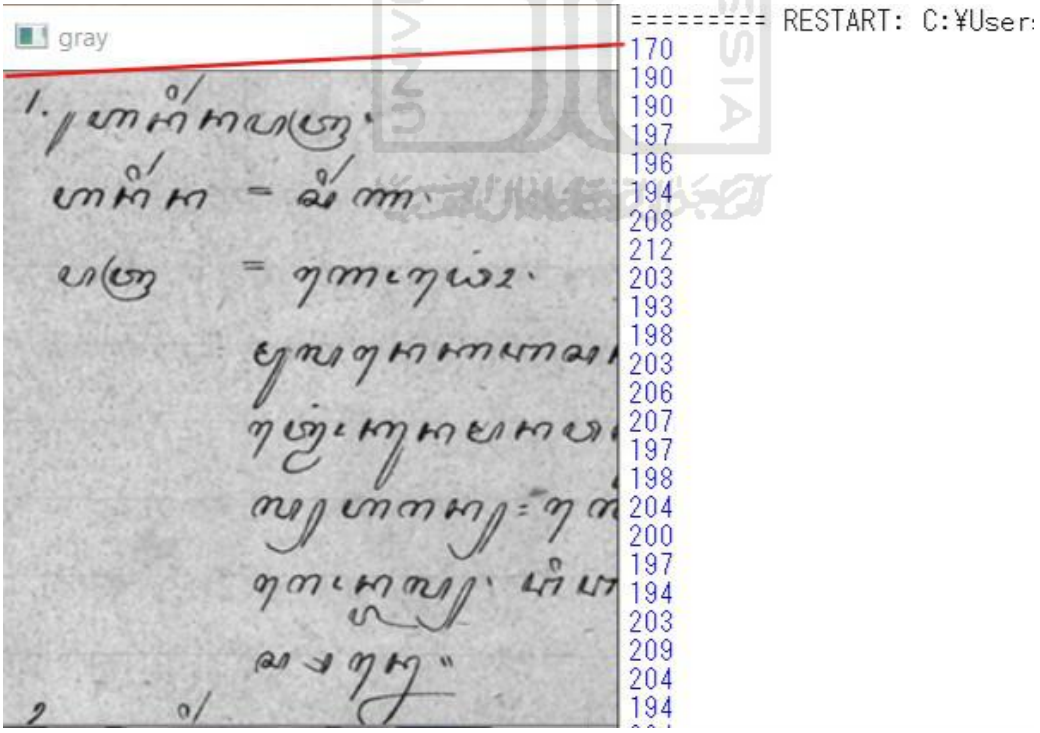
Gambar 2.5 menunjukkan citra berwarna. Terlihat bahwa titik pixel (1,1) mempunyai nilai RGB = [133,177, 236]. Kolom pertama adalah nilai warna merah, kolom kedua adalah nilai warna hijau, dan kolom ketiga adalah nilai warna biru.

2.2.2 Citra *Grayscale*

Citra *grayscale* atau citra keabuan untuk setiap *pixel*nya mempunyai warna gradasi mulai dari putih sampai hitam. Rentang tersebut berarti bahwa setiap *pixel* dapat diwakili oleh 8 bit atau 1 byte (Kusumanto dkk, 2011). Untuk mengubah citra warna menjadi citra *grayscale* dapat menggunakan Persamaan (1) :

$$\text{Gray}(i,j) = (0{,}2989 * \text{Red}(i,j)) + (0{,}5870 * \text{Green}(i,j)) + (0{,}1140 * \text{Blue}(i,j)) \qquad (1)$$

Untuk mendapatkan nilai *grayscale* dapat menggunakan Persamaan (1). Misal perhitungan untuk mendapatkan nilai *grayscale* untuk Gambar 2.5 pada *pixel* (1,1).
$f(1,1) = (0{,}2989 * 133) + (0{,}5870 * 177) + (0{,}1140 * 236) = 170$, hasil tersebut sama dengan nilai pada Gambar 2.6



Gambar 2.6 Citra *Grayscale* dan Intensitas Warnanya

Terlihat terdapat perbedaan antara Gambar 2.5 dan Gambar 2.6. Gambar 2.5 menunjukkan terdapatnya kombinasi intensitas warna merah, hijau, dan biru dari masing-masing *pixel.* Sedangkan Gambar 2.6 menunjukkan hasil *grayscale* dari intensitas abu-abu.

2.2.3 Citra Biner

Citra biner adalah citra yang memiliki dua nilai tingkat keabuan yaitu hitam dan putih. Secara umum proses binerisasi citra *grayscale* untuk menghasilkan citra biner dapat dilihat pada Persamaan (2). (Putra, 2004)

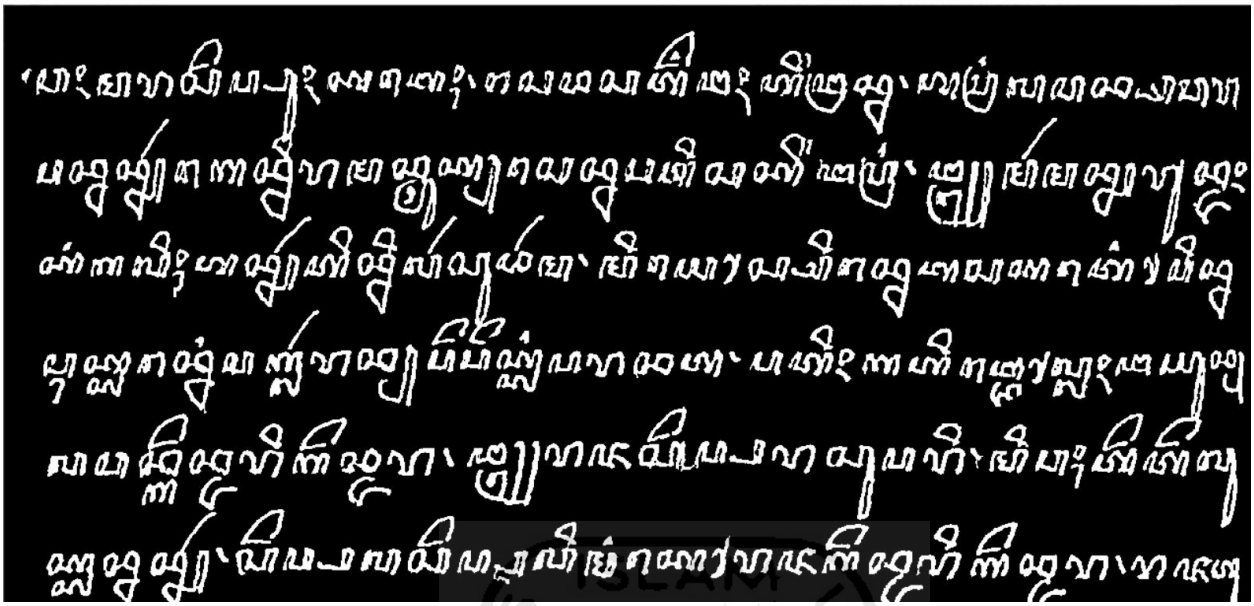
$$g(x,y) = \begin{cases} 1 if \ \ f(x,y) \geq T \\ 0 if \ f(x,y) < T \end{cases} \qquad (2)$$

dengan g(x,y) adalah citra biner dari citra *grayscale* f(x,y) dan T menyatakan nilai ambang.

Nilai T dapat ditentukan dengan salah satu dari 3 cara berikut.

1. Nilai Ambang Global (*Global Threshold*)
   T = T{f(x,y)}
   dengan T tergantung pada nilai *gray level* dari *pixel* pada posisi x,y.
2. Nilai Ambang Lokal (*Local Threshold*)
   T = T{A(x,y), f(x,y)}
   dengan T tergantung pada properti *pixel* tetangga. A(x,y) menyatakan nilai *pixel* tetangga.
3. Nilai Ambang Dinamis (*Dynamic Threshold*)
   T = T{x,y, A(x,y), f(x,y)}
   dengan T tergantung pada koordinat-koordinat *pixel*.

Contoh citra biner dapat dilihat pada Gambar 2.7.



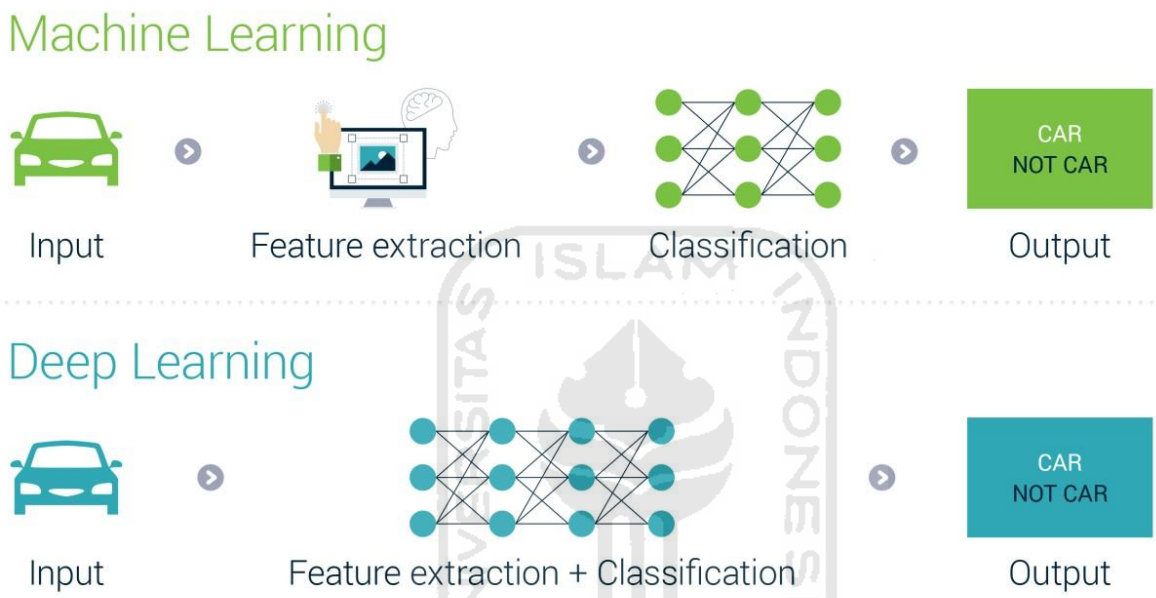
Gambar 2.7 Citra Biner

2.3 *Deep Learning*

Deep Learning (Pembelajaran Dalam) atau sering dikenal dengan istilah Pembelajaran Struktural Mendalam (*Deep Structured Learning)* atau Pembelajaran Hierarki (*Hierarchical learning)* adalah salah satu cabang dari ilmu Pembelajaran Mesin (*Machine Learning*) yang terdiri algoritma pemodelan abstraksi tingkat tinggi pada data menggunakan sekumpulan fungsi transformasi *non-linear* yang ditata berlapis-lapis dan mendalam. Teknik dan algoritma dalam Pembelajaran Dalam dapat digunakan baik untuk kebutuhan Pembelajaran Terarah (*Supervised Learning*), Pembelajaran Tak Terarah (*Unsupervised Learning*) dan Semi-terarah (*Semi-supervised Learning*) dalam berbagai aplikasi seperti pengenalan citra, pengenalan suara, klasifikasi teks, dan sebagainya. *Deep Learning* disebut sebagai *Deep* (dalam) karena struktur dan jumlah jaringan saraf pada algoritmanya sangat banyak bisa mencapai hingga ratusan lapisan(Dadang, 2018).

Deep Learning adalah salah satu jenis algoritma jaringan syaraf tiruan yang menggunakan *metadata* sebagai *input* dan mengolahnya menggunakan sejumlah lapisan tersembunyi (*hidden layer*) transformasi *non linier* dari data masukan untuk menghitung nilai *output*. Algoritma pada *Deep Learning* memiliki fitur yang unik yaitu sebuah fitur yang mampu mengekstraksi secara otomatis. Hal ini berarti algoritma yang dimilikinya secara otomatis dapat

menangkap fitur yang relevan sebagai keperluan dalam pemecahan suatu masalah. Algoritma semacam ini sangat penting dalam sebuah kecerdasan buatan karena mampu mengurangi beban pemrograman dalam memilih fitur yang eksplisit. Dan, algoritma ini dapat digunakan untuk memecahkan permasalahan yang perlu pengawasan (*supervised*), tanpa pengawasan (*unsupervised*), dan semi terawasi (*semi supervised*) (Dadang, 2018). Perbedaan *Machine Learning* dan *Deep Learning* dapat dilihat pada Gambar 2.8.



Gambar 2.8 Perbedaan Machine Learning dan Deep Learning

2.4 *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah salah satu jenis *neural network* yang biasa digunakan pada data *image*. CNN bisa digunakan untuk mendeteksi dan mengenali objek pada sebuah citra. Secara garis besar CNN tidak jauh beda dengan *neural network* biasanya. CNN terdiri dari *neuron* yang memiliki *weight*, *bias* dan *activation function*. Proses yang dilakukan oleh *Convolutional Neural Network*, yaitu: *Convolutional Layer*, *Non-Linearity Layer* (ReLU *Layer*), *Pooling Layer*, dan *Fully-Connected Layer*(Paoletti dkk, 2017).

2.4.1 *Convolutional Layer*

Convolutional Layer adalah *layer* yang memiliki tujuan untuk mengambil informasi posisi dan nilai dari *layer* sebelumnya dengan cara melakukan konvolusi, di mana *neuron* melakukan operasi *dot product* antara *weights* dan *region* kecil dari *layer* sebelumnya.
Rumus yang digunakan untuk perhitungan *Convolutional Layer* dapat dilihat pada Persamaan (3) (Paoletti dkk, 2017) :

$$Z_i^l = B^l + \sum_{j=1}^{k^{i-1}} W_{ij}^l * Z_j^{l-1}, where \quad i \in \left[1, k^l\right], B^l \tag{3}$$

2.4.2 *Non-Linearity Layer*

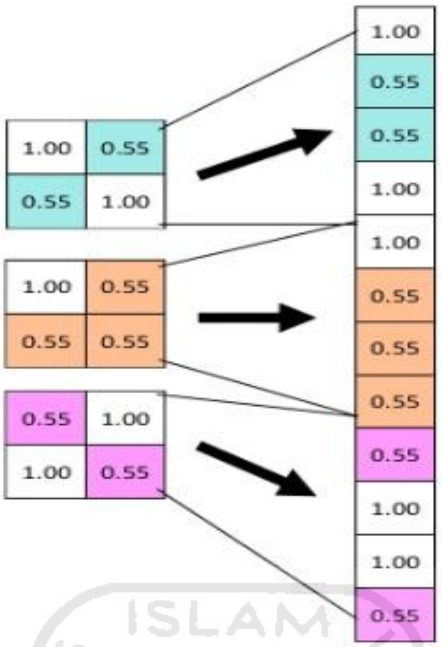
Non-Linearity layer ini berguna untuk membuat hasil dari *Convolutional Layer* menjadi tidak *linear*. Dengan cara memasukkan hasil dari *Convolutional Layer* ke dalam suatu fungsi.

Rumus untuk *Non-Linearity Layer* dapat dilihat pada Persamaan (4) (Paoletti dkk, 2017) :

$$a^l = f(z^l) \tag{4}$$

2.4.3 *Fully-Connected Layer*

Fully-Connected Layer adalah *layer* terakhir di mana data berupa matriks n-dimensi diubah menjadi sebuah matriks *linear* atau 1-dimensi, sehingga dapat dilakukan klasifikasi dengan lebih mudah. Contoh dapat dilihat pada Gambar 2.9(Paoletti dkk, 2017) :



Gambar 2.9 Contoh *Fully-Connected Layer*

2.5 Cohen's Kappa

Untuk mengukur tingkat kesepakatan antara sistem dengan naskah asli dapat menggunakan Koefisien Cohen's Kappa. Secara umum koefisien Cohen's Kappa dapat digunakan untuk mengukur tingkat kesepakatan (*degree of agreement*) dari dua penilai dalam mengklasifikasikan objek ke dalam kelompok dan mengukur kesepakatan alternatif metode baru dengan metode yang sudah ada(Nugraheni, 2017).

Rumus dari Koefisien Cohen's Kappa dapat dilihat pada Persamaan (5) :

$$K = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \tag{5}$$

2.6 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar(Kuhlman, 2012).

Python dapat digunakan untuk pemrograman yang memerlukan dinamisme tinggi, waktu pengembangan yang cepat, aplikasi skala besar yang memerlukan orientasi objek, dan fleksibilitas tinggi. Python juga dapat digunakan untuk membuat banyak aplikasi mulai dari aplikasi perkantoran, aplikasi web, simulasi yang memerlukan perhitungan tingkat tinggi, administrasi, hingga sistem operasi(Noprianto, 2002).

Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi(Noprianto, 2002).

2.7 Tensor Flow

Tensorflow adalah suatu *open-source software library* untuk *dataflow programming*. Tensorflow merupakan suatu *library* matematika simbolik yang juga digunakan untuk aplikasi *Machine Learning* seperti *Neural Network*. Tensorflow dikembangkan oleh Google Brain Team dan juga digunakan *research* oleh Google sendiri (Metz, 2015)

Tensorflow dirilis pertama kali pada 11 Februari 2017. Keunggulan tensorflow adalah dapat dijalankan di beberapa CPU dan GPU (*Graphic Processing Unit*). Tensorflow dapat digunakan pada 64-bit Linux, macOS, Windows, dan Mobile Computing Platforms termasuk Android dan iOS.

2.8 Keras

Keras adalah sebuah *high-level neural networks* API yang ditulis dalam bahasa pemrograman Python dan mempunyai kemampuan untuk jalan diatas TensorFlow, CNTK, maupun Theano. Keras dikembangkan dengan tujuan untuk menjadikan penelitian lebih cepat. Keras juga merupakan sebuah library open source. Keras berfokus pada kemampuannya yang User Friendly, Modular, dan Extensible(2019).

BAB III

METODOLOGI PENELITIAN

3.1 Data

Untuk memulai penelitian ini maka diperlukan data yang akan diproses, untuk mendapatkannya maka diperlukan akuisisi data. Akuisisi data merupakan tahap awal dari penelitian ini, yaitu untuk mendapatkan citra digital. Proses ini bertujuan untuk mendapatkan citra digital naskah Kuno yang menggunakan aksara Jawa. Data-data citra atau gambar yang digunakan dalam penelitian ini diperoleh dari British Library. Naskah yang digunakan adalah naskah yang berjudul "Kocap wulangira Sri Bupati Sinuhun Swarga Ngayogyakarta ḍumatĕng putra wayahe prituwin mring wadya gung, i.e. the teachings of Sultan Hamĕngkubuwana I" koleksi tahun 1812 halaman f5.r, f5.v, f6.r, dan f6.v. Naskah tersebut dapat diakses pada http://www.bl.uk/manuscripts/FullDisplay.aspx?ref=Add_MS_12337. Penelitian ini hanya akan mengambil empat baris pertama dari setiap halaman. Gambar naskah yang digunakan dapat dilihat pada Lampiran 3.1.

Data-data ini nantinya akan disegmentasi dan akan dibagi menjadi dua bagian, yaitu data latih dan data uji.

3.1.1 Data Latih

Data latih merupakan sekumpulan data-data yang akan digunakan sebagai acuan dan juga tolak ukur dari data uji. Data latih ini diperoleh dari segmentasi beberapa citra naskah kuno aksara Jawa lalu melakukan beberapa modifikasi supaya mendapatkan hasil yang sedikit berbeda-beda. Tahap ini membutuhkan data yang beraneka ragam supaya data yang nanti akan diuji terdapat kemiripan dengan data latih. Pemilihan data latih ini dilakukan dengan cara manual dengan melihat jenis dari aksara Jawa. Data latih ini akan digunakan untuk keperluan *training* dan *testing.* Pada penelitian ini dibuat 24 kategori aksara Jawa yang akan diklasifikasi yaitu 20 huruf *dentawyanjana* dan 4 *sandhangan yaitu wulu, pepet, cecak,* dan *layar*. Data latih ini diperoleh dari hasil segmentasi citra naskah dengan nama file 'Naskah-f5v.jpg', 'Naskah-f6r.jpg', dan 'Naskah-f6v.jpg'. Dari total 2880 citra yang tersegmentasi, akan digunakan proporsi 75% untuk data latih dan 25% untuk testing, Sehingga 2160 citra (masing-masing 90 citra per-kategori) untuk *training* dan 720 citra (masing-masing 30 citra per-kategori) untuk *testing.*

3.1.2 Data Uji

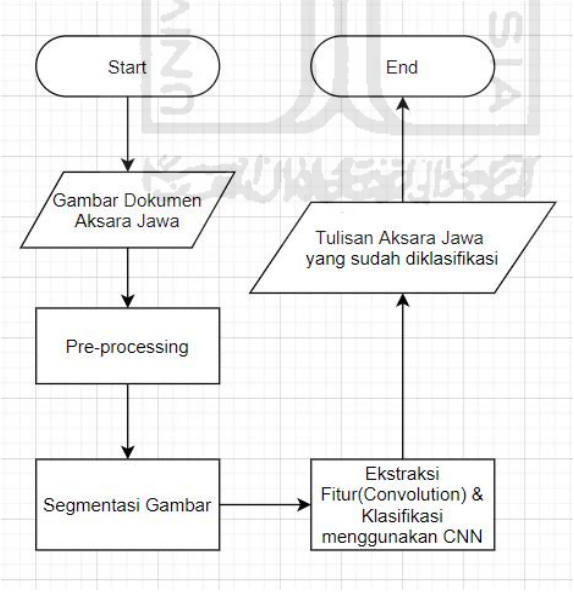
Data uji adalah data yang akan digunakan untuk percobaan terhadap sistem. Dengan menggunakan data uji ini, sistem akan mendeteksi tingkat kemiripan data yang diuji dengan data latih dan hasil dari proses pencocokan ini akan ditemukan nilai akurasi. Data uji ini didapat dari mengambil segmentasi citra naskah aksara Jawa. Data yang diuji adalah citra dengan nama file ’Naskah-f5r.jpg’

3.2 Perancangan Sistem

Tahap perancangan sistem ini dilakukan dengan melihat hasil dari analisis kebutuhan yang telah dilakukan. Tahap ini akan dibagi menjadi dua bagian yaitu *flowchart* gambaran umum dan *flowchart* sistem secara detail.

3.2.1 *Flowchart* Gambaran Umum

Flowchart gambaran umum ini merupakan penggambaran umum atau sederhana dari sistem yang akan dibuat secara keseluruhan. *Flowchart* gambaran umum dapat dilihat pada Gambar 3.1.



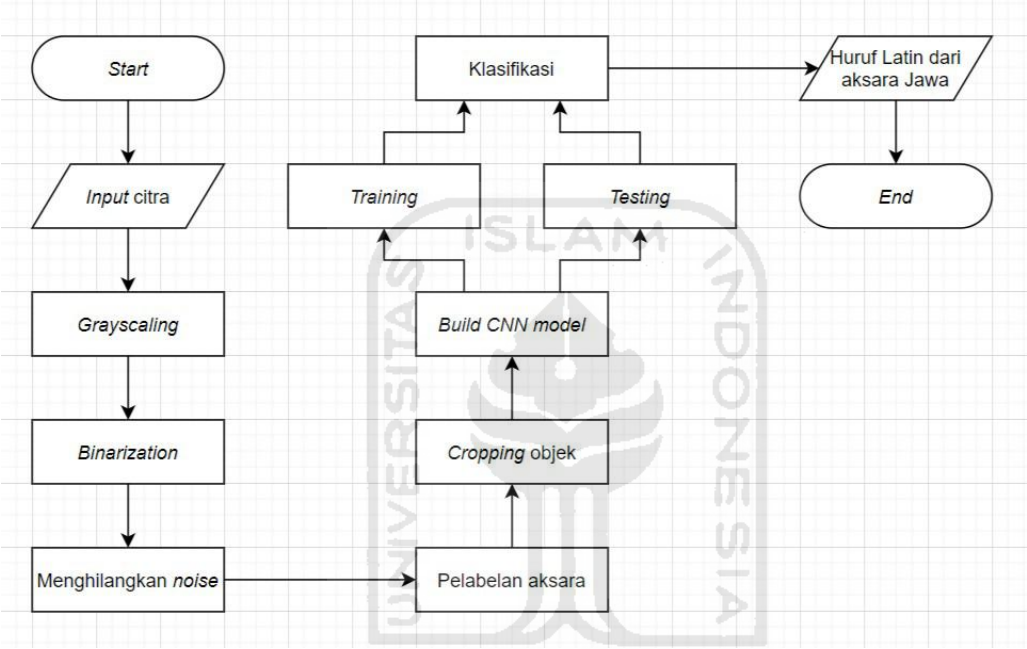
Gambar 3.1 Flowchart Garis Besar Sistem

Diawali dengan proses akuisisi data. Lalu dilanjutkan dengan proses *preprocessing* yang dapat terdiri dari *grayscale*, binerisasi, *resize* dsb . Selanjutnya adalah proses segmentasi citra untuk memotong huruf-huruf Jawa dari suatu dokumen menjadi gambar satu huruf Jawa. Setelah

itu dilakukan Ekstraksi Fitur(*Convolution*) yang bertujuan untuk mengekstraksi fitur dari setiap huruf Jawa yang telah disegmentasi. Yang terakhir adalah Klasifikasi menggunakan *Convolutional Neural Network*(CNN) yang digunakan untuk mengenali huruf Jawa.

3.2.2 *Flowchart* Sistem

Flowchart pada bagian ini akan menjelaskan secara detail proses-proses yang terjadi pada sistem yang dibuat. *Flowchart* sistem dapat dilihat pada Gambar 3.2.



Gambar 3.2 Flowchart Sistem

Tahapan sistem akan dimulai dari menginput citra warna dari naskah aksara Jawa lalu melakukan 11 tahapan yang nanti akan menghasilkan output huruf Latin dari huruf Jawa yang terdeteksi pada naskah aksara Jawa. Berikut adalah penjelasan dari 11 tahapan tersebut:

1. Menginput citra yang akan diklasifikasi ke dalam sistem. Jenis citra yang diinput adalah citra warna yang bertipe ‘jpg’.
2. Tahapan untuk mengubah citra warna menjadi citra *gray* yang nanti akan dilanjutkan ke proses untuk mendapatkan citra biner.
3. Proses untuk mengubah citra *gray* menjadi citra biner. Proses ini bertujuan untuk memisahkan objek huruf dengan latar belakangnya.

4. Proses menghilangkan *noise* atau derau ini bertujuan untuk menghilangkan objek-objek kecil yang bukan termasuk dari objek huruf.
5. Pelabelan aksara bertujuan untuk memberikan tanda pada objek-objek yang terdeteksi sebagai huruf.
6. *Cropping* objek dilakukan untuk memotong citra yang sudah dilabelkan sehingga menjadi citra yang terpisah.
7. Build Models CNN bertujuan untuk membuat model CNN yang digunakan untuk klasifikasi huruf Jawa.
8. Proses *training* dilakukan untuk melatih model yang sudah dibuat sebelumnya dengan menggunakan sejumlah data latih yang sudah disiapkan.
9. Tahap *testing* ini adalah proses yang dilakukan untuk menguji atau mengevaluasi model yang sudah dilatih sebelumnya dengan menggunakan data latih yang sudah disiapkan.
10. Klasifikasi huruf Jawa dilakukan berdasarkan hasil CNN dari *training* dan *testing* sebagai acuan.
11. Hasil dari klasifikasi adalah huruf Latin sesuai huruf Jawa yang terdeteksi.

3.3 Implementasi

Proses implementasi menggunakan bahasa pemrograman python dan beberapa *tools* seperti OpenCV, Tensorflow, Keras, dsb.

3.4 Pengujian

Proses dari pengujian ini dilakukan supaya dapat mengetahui hasil dari klasifikasi sistem terhadap huruf Jawa yang terdeteksi. Pengujian ini dilakukan dengan cara manual yaitu dengan cara menerjemahkan huruf secara satu-persatu. Metode pengujian yang digunakan adalah metode *Cohen's Kappa*. Hasil dari pengujian ini digunakan untuk mengetahui seberapa besar keberhasilan dari sistem dalam klasifikasi huruf Jawa.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Input Citra

Pada Tahap ini, citra yang akan digunakan untuk dikenali diinput ke dalam sistem. Citra yang digunakan adalah citra warna dengan tipe 'jpg'. Setelah citra diinput, citra tersebut akan ditampilkan di layar. Contoh *Code* dari input citra dapat dilihat pada Gambar 4.1.

```
#import image
image = cv2.imread('naskah-f5r.jpg') #menginput file 'naskah-f5r.jpg'
cv2.imshow('color',image)
```

Gambar 4.1 Code Program Input Citra

4.2 Preprocessing

Proses dari preprocessing ini diawali dengan mengubah citra yang berwarna menjadi citra abu-abu atau grayscale. Setelah proses grayscale selesai, maka dilakukan proses menghilangkan *noise*.

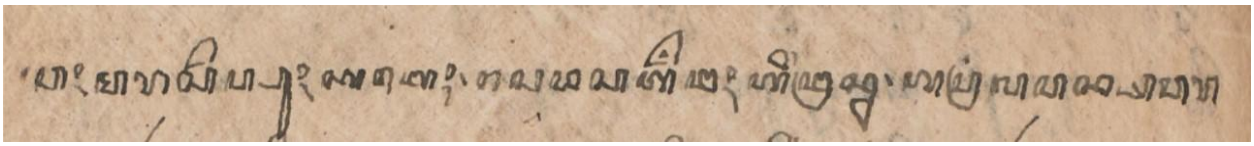
4.2.1 Grayscale

Citra warna yang diinput ke dalam sistem sebelumnya akan diubah menjadi citra *grayscale* atau skala keabuan. Contoh *code* program untuk mengubah citra warna menjadi citra *grayscale* dapat dilihat pada Gambar 4.2.

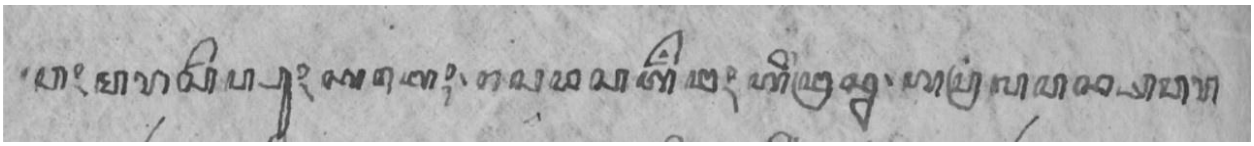
```
#grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Gambar 4.2 Code Program Grayscale

Hasil dari proses *grayscale* ini dapat dilihat pada Gambar 4.3.



(a)



(b)

Gambar 4.3 (a) Citra Warna, (b) Citra Grayscale

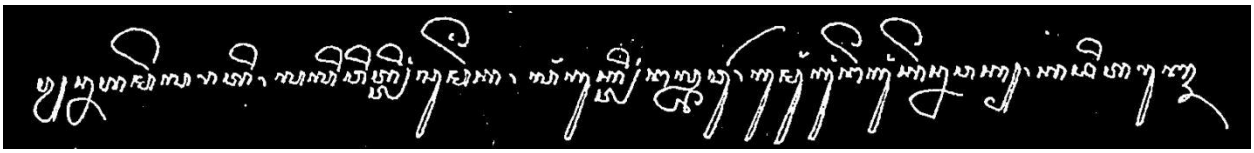
4.2.2 *Binarization*

Setelah selesai proses *grayscale*, dilanjutkan dengan proses segmentasi. Segmentasi citra itu adalah salah satu bagian penting dari *image processing* untuk membedakan objek citra dengan yang bukan objek. Metode segmentasi yang digunakan pada penelitian kali ini adalah *binarization* atau binerisasi. Dalam proses ini citra akan diubah menjadi citra biner atau citra hitam dan putih. Dalam sistem ini, metode yang digunakan adalah *Adaptive Threshold*. Contoh *Code* program yang digunakan dapat dilihat pada Gambar 4.4.

```
#binarization
thresh = cv2.adaptiveThreshold(gray,255,
     cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
     cv2.THRESH_BINARY_INV,21,10)
thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, (5,5))
```

Gambar 4.4 Code Program Binarization

Hasil dari tahap binerisasi ini dapat dilihat pada Gambar 4.5.



Gambar 4.5 Hasil dari Proses *Binarization*

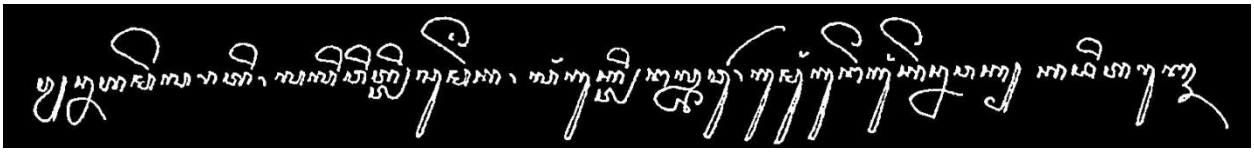
4.2.3 Menghilangkan *Noise*

Setelah mendapatkan hasil dari binerisasi, hasil tersebut masih terdapat *noise* atau derau. Untuk mendapatkan hasil yang optimal maka *noise* tersebut harus dihilangkan. Contoh *code* program untuk mengilangkan *noise* tersebut dapat dilihat pada Gambar 4.6.

```
thresh = cv2.morphologyEx(thresh, cv2.MORPH_CLOSE, (5,5))
nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(thresh, connectivity=8)
sizes = stats[1:, -1]; nb_components = nb_components - 1
min_size = 50

thresh = np.zeros((output.shape))
for i in range(0, nb_components):
    if sizes[i] >= min_size:
        thresh[output == i + 1] = 255
thresh = cv2.convertScaleAbs(thresh)
```

Gambar 4.6 Code Program Menghilangkan Noise

Hasil dari proses menghilangkan *noise* dapat dilihat pada Gambar 4.7.



Gambar 4.7 Citra Biner tanpa Noise

Hasil proses ini pada semua naskah dapat dilihat pada Lampiran 4.1.

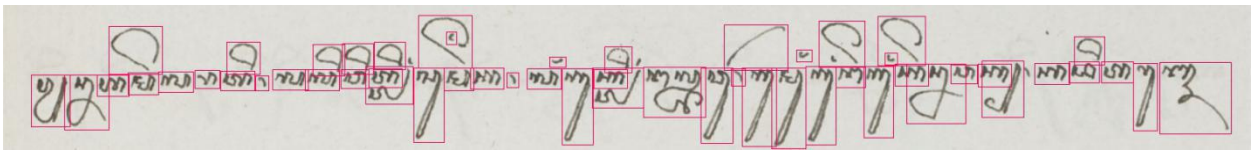
4.3 Pelabelan Aksara

Setelah mendapatkan hasil dari binerisasi yang tidak ada *noise*, maka dilanjutkan dengan proses untuk melabelkan aksara. Proses ini digunakan untuk memberi label yang berbeda pada setiap karakter atau huruf sehingga yang satu dengan yang lain dapat dipisahkan. Pada sistem ini langkah pertama yang digunakan adalah mencari *contour* pada citra, langkah ini menggunakan fungsi 'cv2.findContours'. Setelah mendapatkan *contour* citra, *contour* tersebut kita urutkan menggunakan fungsi 'sorted'. Setelah diurutkan, tahap selanjutnya adalah mendapatkan *bounding box* dengan fungsi 'cv2.boundingRect'. Setelah itu mendapatkan *Region of Image* (ROI). Contoh *code* program yang digunakan dapat dilihat pada Gambar 4.8.

```
#find contours
ctrs, hier = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
      cv2.CHAIN_APPROX_SIMPLE)

#sort contours
sorted_ctrs = sorted(ctrs, key=lambda ctr:
      cv2.boundingRect(ctr)[0])

for i, ctr in enumerate(sorted_ctrs):
    # Get bounding box
    x, y, w, h = cv2.boundingRect(ctr)

    # Getting ROI
    roi1 = thresh[y-3:y+h+3, x-3:x+w+3]

    # show ROI
     cv2.rectangle(image,(x-3,y-3),( x + w + 3, y + h + 3),(90,0,255),1)
```

Gambar 4.8 Code Program Pelabelan Aksara

Hasil dari proses pelabelan dapat dilihat pada Gambar 4.9.



Gambar 4.9 Hasil dari Pelabelan Aksara

Hasil proses ini pada semua naskah dapat dilihat pada Lampiran 4.2.

4.4 *Cropping* Objek

Tahap ini bertujuan untuk *cropping* atau pemotongan objek pada citra sehingga mendapatkan objek-objek secara terpisah. Cara yang digunakan untuk memotong objek adalah dengan mendapatkan *bounding box* dan *ROI*, lalu melakukan *resize* pada hasil potongan menjadi ukuran 32x32 *pixel* dan membalikkan warna hitam dan putih lalu menyimpannya di *directory* "Predict". Contoh *code* program yang digunakan dapat dilihat pada Gambar 4.10.

```
for i, ctr in enumerate(sorted_ctrs):
    # Get bounding box
    x, y, w, h = cv2.boundingRect(ctr)


    # Getting ROI
    roi = thresh[y-3:y+h+3, x-3:x+w+3]
    roi = util.invert(roi)
    roi = cv2.resize(roi, (32,32))
    path = 'Predict/'
    cv2.imwrite(path+'cropped-'+str(i)+'.jpg', roi)
```

Gambar 4.10 Code Program Cropping Objek

Contoh hasil dari pemotongan dapat dilihat pada Gambar 4.11.

(a)

(b)

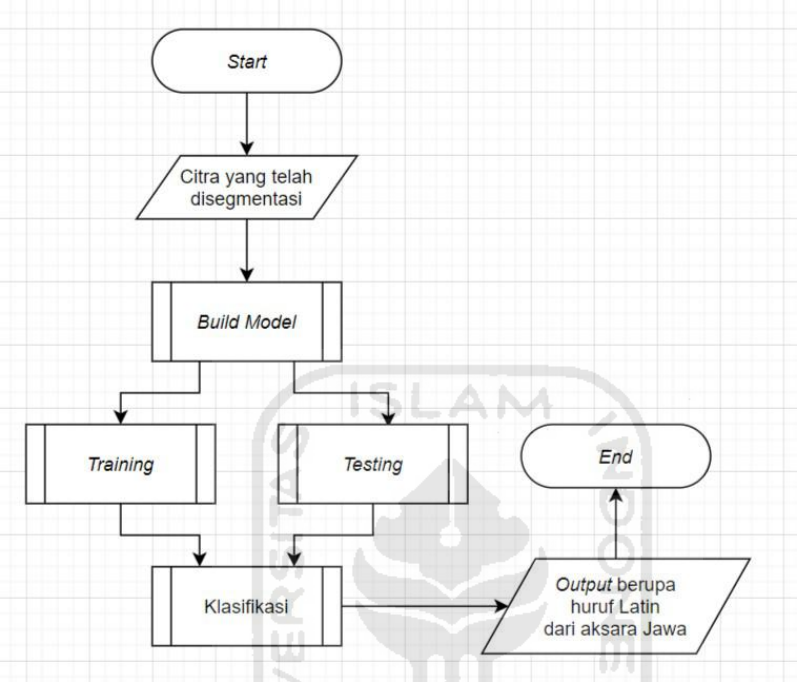
(c)

(d)

Gambar 4.11 Hasil dari Cropping Objek

4.5 *Convolutional Neural Network* (CNN)

Metode yang digunakan untuk klasifikasi huruf Jawa adalah *Convolutional Neural Networ*k. Secara garis besar cara kerja dari CNN dapat dilihat pada Gambar 4.12.



Gambar 4.12 Flowchart Convolutional Neural Network

4.5.1 *Build Models*

Tahap ini adalah proses untuk membuat dan mendeklarasikan seluruh *variable* untuk model yang akan digunakan pada proses *training, testing,* dan klasifikasi. Sumber model patokan yang digunakan untuk penelitian ini adalah model VGG-16 dari (VGG - ILSVRC Winner 2014) dengan struktur model sebagai berikut:

Model 1:

- Kernel size = 3x3
- 13 Convolution + 5 Max Pool + 3 Fully Connected
- Architecture *variable*:
  - Conv64 - Conv64 - Maxpool2 - Conv128 - Conv128 - Maxpool2- Conv256 - Conv256 - Conv256 - Maxpool2 - Conv512 - Conv512 - Conv512 - Maxpool2 - Conv512 - Conv512 - Conv512 - Maxpool2 - FC4096 - FC4096 - FC24

Pada penelitian ini, kita akan juga akan membuat model dengan struktur yang diminimalkan lalu nanti akan dibandingkan dengan model sebelumnya yang menggunakan VGG-16, model yang minimal tersebut adalah :

Model 2:

- Kernel size = 3x3
- 8 Convolution + 4 Max Pool + 3 Fully Connected
- Architecture *variable*:
  - Conv32 - Conv32 - Maxpool2 - Conv64 - Conv64 - Maxpool2- Conv128 - Conv128 - Maxpool2 - Conv256 - Conv256 - Maxpool2 - FC4096 - FC4096 - FC24

Keterangan:

- Conv*N* = *Convolutional Layer* dengan jumlah filter *N*
- Maxpool2 = *Max Pooling Layer* dengan *size* 2x2
- FC*m* = *Fully Connected Layer* dengan jumlah *m*

Contoh *code* yang digunakan untuk *build models* dengan Model 1 dapat dilihat pada Gambar 4.13.

```
trdata = ImageDataGenerator()
traindata =
trdata.flow_from_directory(directory="train",target_size=(32,32))
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory(directory="test",
target_size=(32,32))

model = Sequential()
model.add(Conv2D(input_shape=(32,32,3),filters=64,kernel_size=(3,3),paddin
g="same", activation="relu"))
model.add(Conv2D(filters=64,kernel_size=(3,3),padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=512, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=24, activation="softmax"))

from keras.optimizers import Adam
opt = Adam(lr=0.0001)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])

model.summary()
```

Gambar 4.13 Code Program Build Models dengan Model 1

Contoh *code* yang digunakan untuk *build models* dengan Model 2 dapat dilihat pada Gambar 4.14.

```
trdata = ImageDataGenerator()
traindata =
trdata.flow_from_directory(directory="train",target_size=(32,32))
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory(directory="test",
target_size=(32,32))

model = Sequential()
model.add(Conv2D(input_shape=(32,32,3),filters=32,kernel_size=(3,3),pad
ding="same", activation="relu"))
model.add(Conv2D(filters=32,kernel_size=(3,3),padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding="same",
activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))

model.add(Flatten())
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=4096,activation="relu"))
model.add(Dense(units=24, activation="softmax"))

from keras.optimizers import Adam
opt = Adam(lr=0.0001)
model.compile(optimizer=opt, loss=keras.losses.categorical_crossentropy,
metrics=['accuracy'])

model.summary()
```

Gambar 4.14 Code Program Build Models dengan Model 2

4.5.2 *Training*

Tahap ini adalah proses untuk melatih model yang sudah dibuat sebelumnya dengan menggunakan sejumlah data latih untuk *training* yang sudah disiapkan. Contoh *code* yang digunakan akan ditunjukkan bersama dengan *code* untuk *testing*.

4.5.3 *Testing*

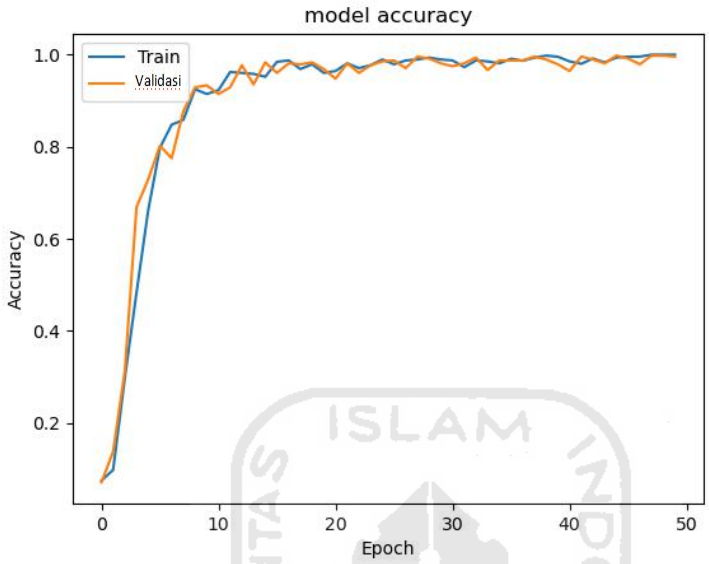
Tahap *testing* ini adalah proses yang dilakukan untuk menguji atau mengevaluasi model yang sudah dilatih sebelumnya dengan menggunakan data latih untuk *testing* yang sudah disiapkan. Penelitian ini menggunakan 50 *epochs* (iterasi) yang artinya *training* dan *testing* akan dilakukan dengan 50 kali putaran. Contoh *Code* program *training* dan *testing* dapat dilihat pada Gambar 4.15.

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint = ModelCheckpoint("vgg16.h5", monitor='val_accuracy',
verbose=1, save_best_only=True, save_weights_only=False, mode='auto',
period=1)
early = EarlyStopping(monitor='val_accuracy', min_delta=0,
patience=20, verbose=1, mode='auto')
hist = model.fit_generator(steps_per_epoch=15,generator=traindata,
validation_data= testdata,
validation_steps=15,epochs=50,callbacks=[checkpoint,early])

import matplotlib.pyplot as plt
plt.plot(hist.history["accuracy"])
plt.plot(hist.history['val_accuracy'])
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation
Loss"])
plt.show()
```

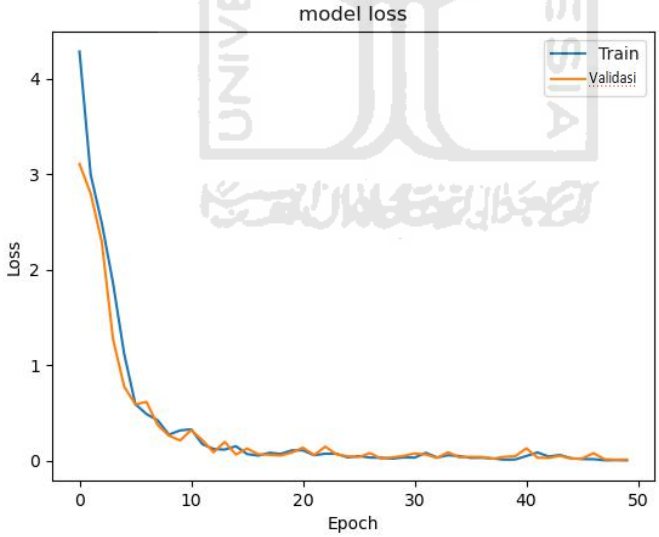
Gambar 4.15 Code Program Training dan Testing

Setelah melakukan training dan testing dengan kedua model yang dibuat, didapatkan hasil sebagai berikut.

Hasil dari *training* dan *testing* dengan Model 1 dapat dilihat pada Gambar 4.16.



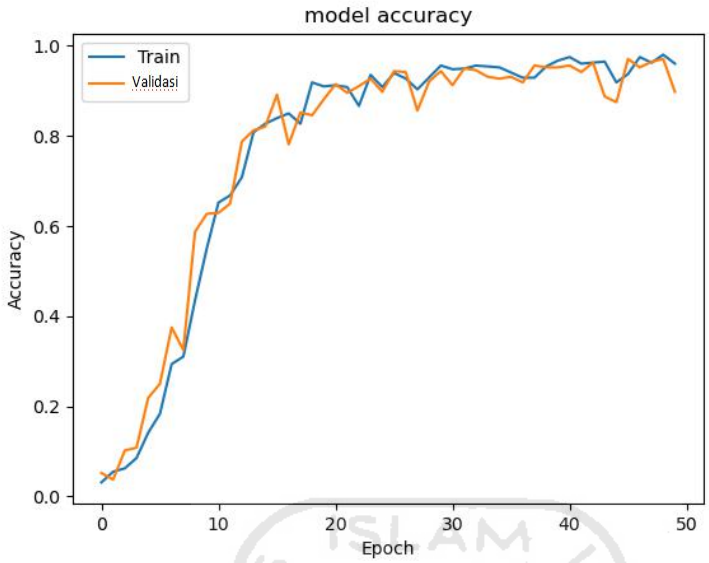
(a)



(b)

Gambar 4.16 Hasil dari Training dan Testing (a) grafik tingkat akurasi (b) grafik loss dengan Model 1

Hasil dari *training* dan *testing* dengan Model 2 dapat dilihat pada Gambar 4.17.



(a)

(b)

Gambar 4.17 Hasil dari Training dan Testing (a) grafik tingkat akurasi (b) grafik loss dengan Model 2

Gambar diatas menunjukan grafik pergerakan nilai akurasi dan nilai loss untuk *training* dan *testing* yang dihasilkan pada setiap iterasi (*epoch*).

Dari hasil pada Gambar 4.16 hasil menggunakan Model 1 dapat dilihat bahwa *accuracy training* pada *epochs* terakhir dapat mencapai 97% dengan nilai *loss* sebesar 0.2708 dan

accuracy validasi mencapai 95% dengan nilai *loss* sebesar 0.2330. Sedangkan dari hasil pada Gambar 4.17 hasil menggunakan Model 2 dapat dilihat bahwa *accuracy training* pada *epochs* terakhir mencapai 95% dengan nilai *loss* sebesar 0.3542 dan *accuracy* validasi mencapai 87% dengan nilai *loss* sebesar 0.4723. Hasil Model 1 lebih baik dari Model 2, oleh karena itu pada penelitian ini untuk klasifikasi akan menggunakan Model 1 yaitu model VGG-16.

4.5.4 Klasifikasi

Klasifikasi huruf Jawa dilakukan berdasarkan hasil CNN dari *training* dan *testing* sebagai acuan. Data uji yang hampir mendekati hasil dari CNN akan ditulis dengan huruf Latin. Pada penelitian ini apabila ‘pos’ karakter yang diuji bernilai 0 maka akan ditulis sebagai huruf ‘ba’, ketika ‘pos’ karakter yang diuji bernilai 1 maka akan ditulis sebagai huruf ‘ca’ dan begitu seterusnya hingga semua huruf Jawa yang terdeteksi sudah terklasifikasi.

Contoh *code* program dapat dilihat pada Gambar 4.18.

```
saved_model = load_model("vgg16.h5")

_nsre = re.compile('([0-9]+)')
def natural_sort_key(s):
    return [int(text) if text.isdigit() else text.lower()
            for text in re.split(_nsre, s)]

arrPos = []
arrPos.append('ba')
arrPos.append('ca')
arrPos.append('cecak')
arrPos.append('da')
arrPos.append('dha')
arrPos.append('ga')
arrPos.append('ha')
arrPos.append('ja')
arrPos.append('ka')
arrPos.append('la')
arrPos.append('layar')
arrPos.append('ma')
arrPos.append('na')
arrPos.append('nga')
arrPos.append('nya')
arrPos.append('pa')
arrPos.append('pepet')
arrPos.append('ra')
arrPos.append('sa')
arrPos.append('ta')
arrPos.append('tha')
arrPos.append('wa')
arrPos.append('wulu')
arrPos.append('ya')

path = 'Predict'
for filename in sorted(os.listdir(path), key = natural_sort_key):
        img = image.load_img(os.path.join(path, filename),target_size=(32,32))
        img = np.asarray(img)
        img = np.expand_dims(img, axis=0)
        output = saved_model.predict(img)

        max = output[0][0]
        pos = 0
        for i in range(1, 23):
            if output[0][i] > max:
                max = output[0][i]
                pos = I

        #print aksara
        print(arrPos[pos])
        print('\n')
```

Gambar 4.18 Code Program Klasifikasi

Waktu yang diperlukan sistem untuk mengklasifikasi file ‘Naskah-f5r.jpg’ adalah selama 14.48 detik.

4.6 Pengujian

Tahap pengujian ini dilakukan untuk mengetahui tingkat akurasi dari sistem pada setiap data citra naskah aksara Jawa. Pada penelitian ini akan menggunakan metode C*ohen's Kappa.*

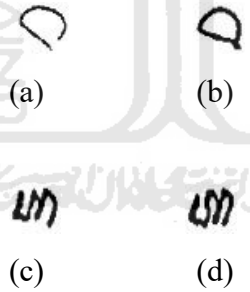
4.6.1 *Cohen's Kappa*

Pengujian C*ohen's Kappa* pada penelitian ini akan menggunakan hasil dari sistem dan hasil yang sebenarnya. Data yang diuji adalah data citra dengan filename 'Naskah-f5r.jpg'. Hasil dari pengujian ini dapat dilihat pada Lampiran 4.3.

Berdasarkan hasil pada Lampiran 4.3, nilai *Kappa* sistem adalah sebesar 0.86 dan nilai error 0.031. Dengan tingkat akurasi 0.86 maka sistem ini termasuk mendapat *level of agreement* yang kuat (*strong*) sehingga sistem sudah bagus.

Hasil pengujian tidak mendapat nilai yang sempurna karena terdapat kesalahan identifikasi. Kesalahan identifikasi terjadi karena terdapat bentuk huruf yang mirip.

Contoh kesalahan dapat dilihat pada Gambar 4.19.



Gambar 4.19 (a) Huruf Pepet (b) Huruf Wulu (c) Huruf Ha (d) Huruf Ta

Contoh kesalahan yang terjadi adalah huruf '*Pepet*' diidentifikasi menjadi '*Wulu*' dan begitu sebaliknya dan juga huruf 'Ha' diidentifikasi menjadi huruf 'Ta' dan sebaliknya.

4.7 Kelebihan dan Kekurangan Sistem

4.7.1 Kelebihan Sistem

- Sistem dapat mengenali huruf Jawa dengan baik dengan tingkat akurasi model sebesar 97% dan nilai pengujian *Kappa* sebesar 0.86.
- Waktu untuk klasifikasi sistem terhitung cepat dengan waktu selama 14.48 detik.

4.7.1 Kekurangan Sistem

- Masih terdapat dua huruf atau lebih yang berdekatan/menempel terdeteksi dan tersegmentasi menjadi satu huruf.
- Masih terdapat kesalahan klasifikasi karena terdapat huruf yang mirip.
- Belum menggunakan UI.

Masih terdapat dua huruf atau lebih yang berdekatan atau menempel terdeteksi dan tersegmentasi menjadi satu huruf karena saat proses deteksi huruf, sistem masih belum mengenali termasuk kategori apa huruf yang terdeteksi sehingga sistem belum bisa membedakan dua huruf atau lebih yang berdekatan atau menempel itu dan menganggapnya sebagai satu huruf oleh karena itu sistem belum dapat memisahkannya menjadi dua atau lebih huruf.

Masih tedapat kesalahan klasifikasi karena terdapat huruf yang mirip, ini dapat terjadi karena huruf yang mirip tersebut memiliki fitur atau ciri yang hampir sama karena saat ekstraksi fitur huruf-huruf tersebut memiliki nilai fitur yang hampir sama sehingga sistem masih salah untuk mengklasifikasikan huruf-huruf tersebut.

Belum menggunakan UI, bahasa pemrograman python sendiri belum mempunyai fitur atau *code* untuk membuat sebuah *interface.* Diperlukan aplikasi *3rd party* lainnya untuk membuat *interface* untuk program atau aplikasi yang menggunakan bahasa pemrograman python.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan semua proses penelitian, maka dapat ditarik kesimpulan sebagai berikut:

1. Sistem mampu mendeteksi tulisan aksara Jawa pada naskah lama dengan cara mengubah citra warna menjadi citra biner untuk memisahkan objek huruf dengan background lalu menghilangkan *noise* sehingga hanya tersisa objek huruf lalu melakukan pelabelan huruf-huruf yang terdeteksi.
2. Sistem sudah mampu mengidentifikasi tulisan aksara Jawa yang telah terdeteksi dengan baik dengan menggunakan metode klasifikasi CNN dengan cara membuat model menggunakan beberapa *Convolutional Layer*, *Max Pooling Layer* dan *Fully Connected Layer* lalu melakukan *training* dan *testing* pada model lalu melakukan klasifikasi terhadap huruf.
3. Performa sistem sudah memuaskan dengan mendapat hasil *training* dan *testing* model yang memuaskan dengan *accuracy* sebesar 95% dan *validation accuracy* sebesar 93%, mendapat hasil nilai *Kappa* 0.86 di mana sudah termasuk mendapatkan *level of agreement* yang kuat serta kecepatan klasifikasi 14.48 detik.

5.2 Saran

Pada bagian akhir ini, penulis akan mengajukan saran-saran untuk selanjutnya apabila terdapat peneliti lain ingin mengembangkan sistem ini supaya lebih baik lagi. Saran yang diajukan sebagai berikut:

1. Membuat semua kategori huruf Jawa untuk model klasifikasi supaya dapat sistem dapat mengenali semua huruf Jawa.
2. Perbanyak lagi data *training* dan *testing* supaya tingkat akurasi dapat lebih tinggi.
3. Perbanyak jenis data *training dan testing* yang berbeda supaya tingkat akurasi dapat lebih tinggi.
4. Menambahkan data yang diuji supaya menambahkan kredibilitas sistem.
5. Membuat UI sehingga orang awam dapat menggunakannya.

DAFTAR PUSTAKA

Gallop, Annabel. (2018). 15,000 images of Javanese Manuscripts from Yogyakarta now online. Diakses dari https://blogs.bl.uk/asian-and-african/2018/09/15000-images-of-javanese-manuscripts-from-yogyakarta-now-online.html

Gonzalez. R. C. dan Woods. R. E. (2001). Digital Image Processing. Addison-Wesley Publishing.

Hartanto, S., Sugiharto, A., & Endah, S. N. (2012). Optical Character Recognition Menggunakan Algoritma Template Matching Correlation. Journal of Informatics and Technology, 1(1), 11-20.

Putri, F. Z., Irawan, B., & Ahmad, U. A. (2016). Perancangan dan Implementasi Directional Feature Extraction dan Support Vector Machines untuk Menerjemah Kata dengan Pengenalan Huruf Hiragana dalam Bahasa Jepang ke Bahasa Indonesia Berbasis Android. eProceedings of Engineering, 3(2).

Saraswati, Ufi. (2011, Maret). Arti dan Fungsi Naskah Kuno Bagi Pengembangan Budaya dan Karakter Bangsa melalui Pengajaran Sejarah. Dipresentasikan pada Seminar Nasional Pendidikan Sejarah, APPS di Bandung.

Widiarti, A. R. (2006). *Pengenalan Citra dokumen Sastra Jawa konsep dan implementasinya*. Doctoral dissertation, Universitas Gadjah Mada.

Aksara Jawa Hanacaraka, Diakses dari http://nusantaranger.com/referensi/buku-elang/chapter-4merah/aksara-jawa-hanacaraka/

Putra, Darma. (2004). Binerisasi Citra Tangan Dengan Metode Otsu. Majalah Ilmiah Teknologi Elektro Vol 3 No 2, Universitas Udayana.

Mardianto S., (2015). Aplikasi Segmentasi Huruf Jawa. Teknik Informatika - Universitas Kristen Petra, Surabaya.

Dadang, W. (2018). Memahami Kecerdasan Buatan berupa Deep Learning dan Machine Learning. Diakses dari https://warstek.com/2018/02/06/deepmachinelearning/

Paoletti, M. E., Haut, J.M., Plaza J., & Plaza A. (2017). "A new deep convolutional neural network for fast hyperspectral image classification", ISPRS Journal of Photogrammetry and Remote Sensing, 16

Kuhlman, D. (2012, Juni) . Python Book: Beginning Python, Advanced Python, and Python Exercises. Section 1.1

Noprianto. (2002). Python & Pemrograman Linux. Yogyakarta: ANDI.

Metz, C (2015, November). TensorFlow, Google's Open Source AI, Points to a Fast-Changing Hardware World. Wired.

Keras Documentation. Keras: The Python Deep Learning Library. Retrieved January 4, 2019, dari https://keras.io/

Afrianto, T. (2017) .Segmentasi Aksara Pada Tulisan Aksara Jawa Menggunakan Adaptive Threshold. STIKI Infromatika Jurnal, 7(01)1-2

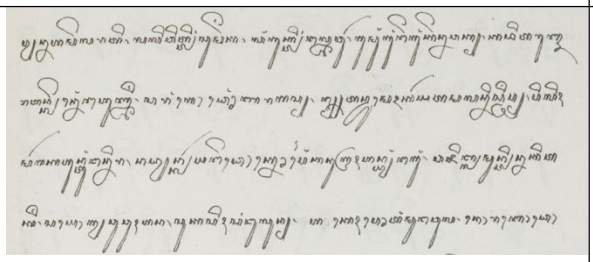
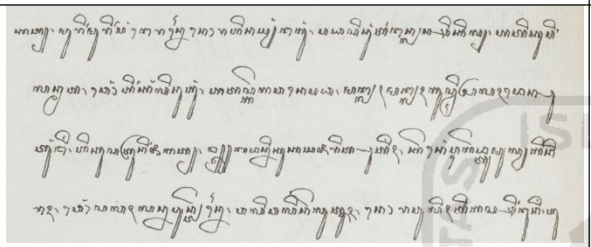
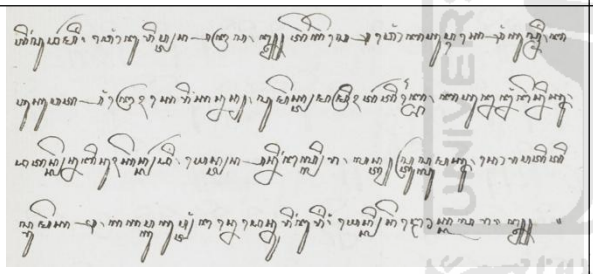
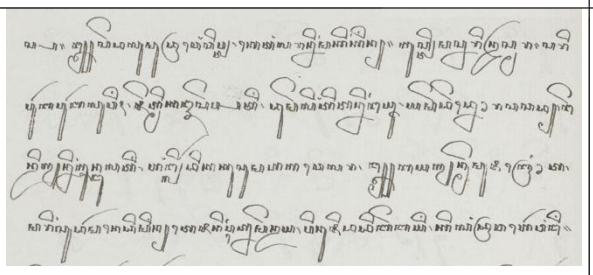
Putra, D. (2010). Pengolahan citra digital. Penerbit Andi.

Kusumanto, R. D., & Tompunu, A. N. (2011). pengolahan citra digital untuk mendeteksi obyek menggunakan pengolahan warna model normalisasi RGB. Semantik, 1(1).

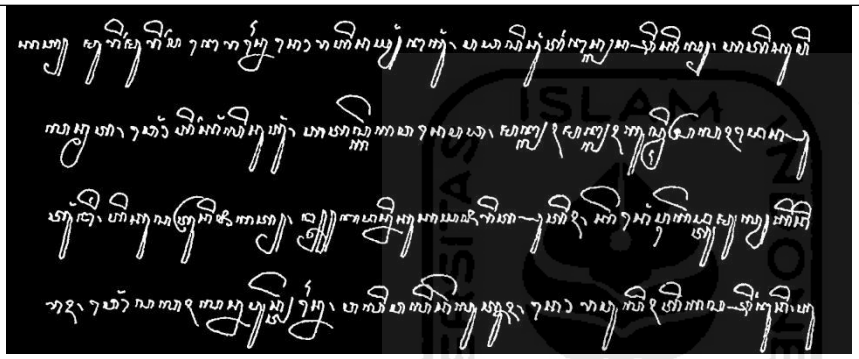
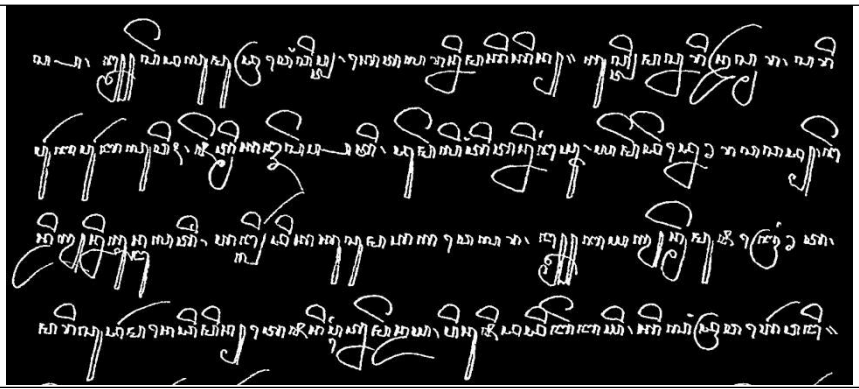
Nugraheni, R. A. (2017). Identifikasi Morfologi Telur dan Larva Nyamuk Pembawa Vektor Penyakit Zoonosis Berbasis Citra Mikroskopis. Informatics Engineering, UII

LAMPIRAN

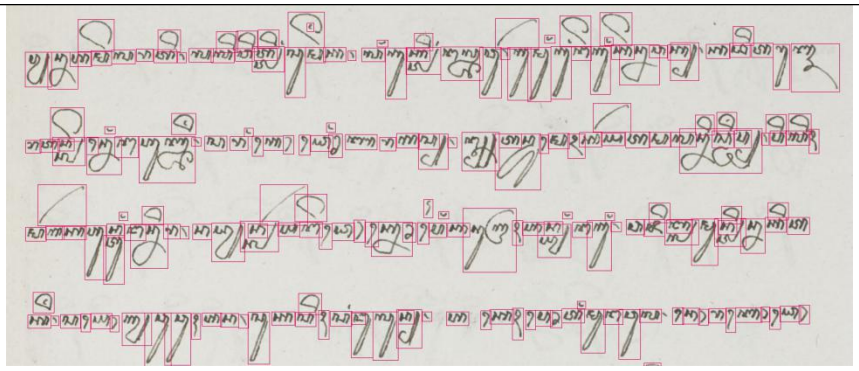
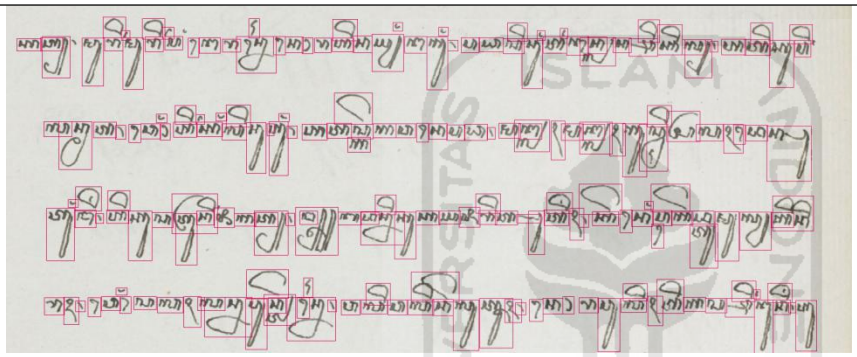
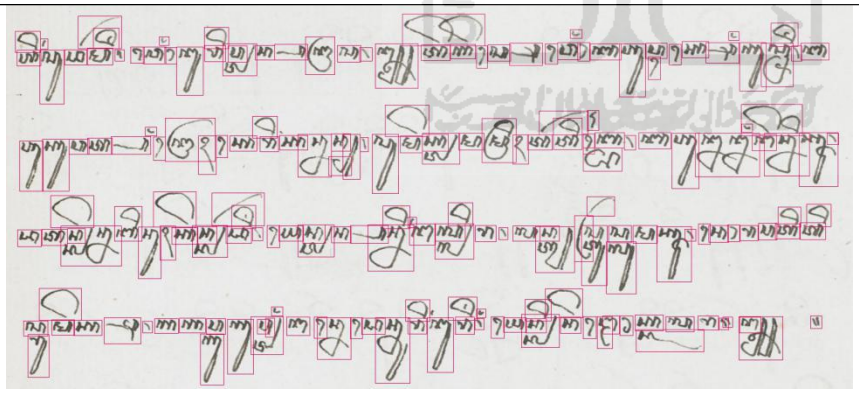
Lampiran 3.1 Data

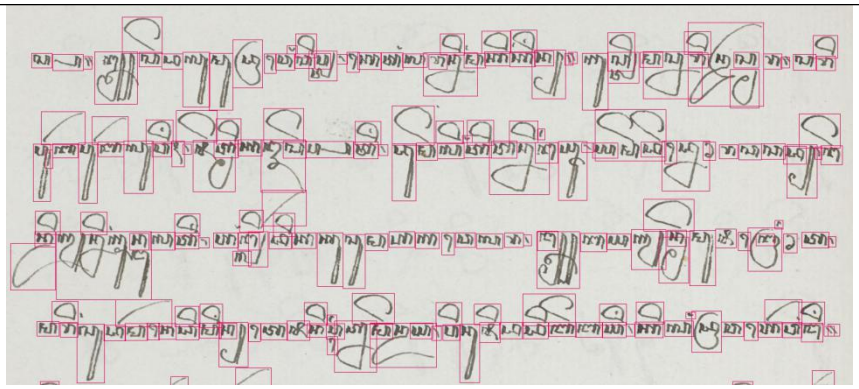
No	Gambar	Nama File	Properties
1		Naskah-f5r.jpg	Dimensions 1608 x 684 Width 1608 pixels Height 684 pixels Horizontal resolution 120 dpi Vertical resolution 120 dpi Bit depth 24
2		Naskah-f5v.jpg	Dimensions 1573 x 646 Width 1573 pixels Height 646 pixels Horizontal resolution 120 dpi Vertical resolution 120 dpi Bit depth 24
3		Naskah-f6r.jpg	Dimensions 1565 x 708 Width 1565 pixels Height 708 pixels Horizontal resolution 120 dpi Vertical resolution 120 dpi Bit depth 24
4		Naskah-f6v.jpg	Dimensions 1569 x 703 Width 1569 pixels Height 703 pixels Horizontal resolution 120 dpi Vertical resolution 120 dpi Bit depth 24

Lampiran 4.1 Citra Biner tanpa *Noise*

No	Gambar	Nama File
1		Naskah-f5r-biner.jpg
2		Naskah-f5v-biner.jpg
3		Naskah-f6r-biner.jpg
4		Naskah-f6v-biner.jpg

Lampiran 4.2 Hasil Pelabelan

No	Gambar	Nama File
1		Naskah-f5r-marked.jpg
2		Naskah-f5v-marked.jpg
3		Naskah-f6r-marked.jpg

4		Naskah-f6v-marked.jpg

Lampiran 4.3 Pengujian *Cohen's Kappa pada 'Naskah-f5r.jpg'*

		Naskah Asli					
		Ha	Na	Ca	Ra	Ka	Da
Sistem	Ha	6	0	0	0	0	0
	Na	0	4	0	0	0	0
	Ca	0	0	5	0	0	0
	Ra	0	0	0	4	0	0
	Ka	0	0	0	0	6	0
	Da	0	0	0	0	0	3
	Ta	0	0	0	0	0	0
	Sa	0	0	0	0	0	0
	Wa	0	0	0	0	0	0
	La	0	0	1	0	0	0
	Pa	0	0	0	0	0	2
	Dha	0	0	0	0	0	0
	Ja	0	0	0	0	0	0
	Ya	0	0	0	0	0	0
	Nya	0	0	0	0	0	0
	Ma	0	0	0	0	2	0
	Ga	0	0	0	0	0	0
	Ba	0	0	0	0	1	0
	Tha	0	0	0	0	0	0
	Nga	0	0	0	0	0	0
	Cecak	0	0	0	0	0	0
	Layar	0	0	0	0	0	0
	Pepet	0	0	0	0	0	0
	Wulu	0	0	0	0	0	0
	Jumlah	6	4	6	4	9	5

Ta	Sa	Wa	La	Pa	Dha
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	1	0	0
0	0	0	0	0	0
2	0	0	0	0	0
0	0	0	0	0	0
4	0	0	0	0	0
0	4	0	0	0	0

0	0	6	0	0	0
0	0	0	4	0	0
0	0	0	0	5	0
0	0	0	0	0	6
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
6	4	6	5	5	6

Ja	Ya	Nya	Ma	Ga	Ba
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	2	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
3	0	0	0	0	0
0	4	0	0	0	0
0	0	2	0	0	0
0	0	0	4	0	0
0	0	0	0	4	0
0	0	0	0	0	5
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
3	4	2	4	6	5

Tha	Nga	Cecak	Layar	Pepet	Wulu	Jumlah
0	0	0	0	0	0	6
0	0	0	0	0	0	4
0	0	0	0	0	0	6
0	0	0	0	0	0	4
0	0	0	0	0	0	10
0	0	0	0	0	0	3
0	0	0	0	0	0	4
0	0	0	0	0	0	4
0	0	0	0	0	0	6
0	0	0	0	0	0	5
0	0	0	0	0	0	7
0	0	0	0	0	0	6
0	0	0	0	0	0	3
0	0	0	0	0	0	4
0	0	0	0	0	0	2
0	0	0	0	0	0	6
0	0	0	0	0	0	4
0	1	2	0	0	0	9
2	0	0	0	0	0	2
0	2	0	0	0	0	2
0	0	12	0	0	0	12
0	0	3	4	0	0	7
0	0	0	0	4	0	4
0	0	0	0	2	17	19
2	3	17	4	6	17	139

Keterangan :

- Baris pada tabel menjelaskan huruf Jawa yang terdeteksi oleh sistem
- Kolom pada tabel menjelaskan huruf Jawa yang terdapat pada naskah

Contoh perhitungan akurasi *Cohen's Kappa*:

		Truth/nilai sebenarnya		Row Marginals	
		Tepat	Tidak tepat		
Sistem	Tepat	147	3	150	rm^1
	Tidak tepat	10	62	72	rm^2
Column Marginals		157	65	222	n
		cm^1	cm^2		

Raw % Agreement

$$\frac{147+62}{222} = .94$$

$$\text{Expected Agreement} = \frac{\left(\frac{cm^1 \times rm^1}{n}\right) + \left(\frac{cm^2 \times rm^2}{n}\right)}{n}$$

KAPPA CALCULATION

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$

Keterangan :

Raw % Agreement = Pr(a)

Expected Agreement = Pr(e)

K = nilai *Kappa*

Dengan rumus tersebut maka tingkat akurasi sistem ini:

$$\Pr(a) = \frac{120}{139} = 0.86$$

Pr(e) = 0.06

n = 139

$$k = \frac{0.86 - 0.06}{1 - 0.06} = 0.86$$

Contoh perhitungan tingkat error *Cohen's Kappa*:

$$SE_k = \sqrt{\frac{\Pr(a)(1-\Pr(a)}{n(1-\Pr(e))^2}}$$

Dengan rumus tersebut maka tingkat error sistem ini:

$$SE_k = \sqrt{\frac{0.86(1-0.86)}{139\,(1-0.06)^2}} = \sqrt{\frac{0.1204}{122.82}}$$

$$SE_k = \sqrt{0.00098029\ \ 3} = 0.031$$

Interpretation of Cohen's kappa.

Value of Kappa	Level of Agreement	% of Data that are Reliable
0–.20	None	0–4%
.21–.39	Minimal	4–15%
.40–.59	Weak	15–35%
.60–.79	Moderate	35–63%
.80–.90	Strong	64–81%
Above.90	Almost Perfect	82–100%