PENGENALAN POLA TULISAN TANGAN AKSARA SASAK MENGGUNAKAN METODE LINEAR DISCRIMINANT ANALYSIS DAN JARINGAN SYARAF TIRUAN JENIS BACKPROPAGATION

Tugas akhir

untuk memenuhi sebagian persyaratan

mencapai derajat Sarjana S-1 Program Studi Teknik Informatika

Oleh:

A.A.Sg. Mas Karunia Maharani

F1D 016 001

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MATARAM

2020

TUGAS AKHIR

PENGENALAN POLA TULISAN TANGAN AKSARA SASAK MENGGUNAKAN METODE LINEAR DISCRIMINANT ANALYSIS DAN JARINGAN SYARAF TIRUAN JENIS BACKPROPAGATION

Oleh :

A.A.SG. MAS KARUNIA MAHARANI
F1D016001

Telah diperiksa oleh Tim Pembimbing :

1. Pembimbing Utama

Tanggal: 15/07/2020

Prof. Dr. Eng. I Gede Pasek Suta Wijaya, ST., MT.
NIP. 197311302000031001

2. Pembimbing Pendamping

Tanggal: 15/07/2020

Fitri Bimantoro, ST.,M.Kom.
NIP. 198606222015041002

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Teknik
Universitas Mataram

Prof. Dr. Eng. I Gede Pasek Suta Wijaya, ST., MT.
NIP. 197311302000031001

TUGAS AKHIR

PENGENALAN POLA TULISAN TANGAN AKSARA SASAK MENGGUNAKAN METODE LINEAR DISCRIMINANT ANALYSIS DAN JARINGAN SYARAF TIRUAN JENIS BACKPROPAGATION

Oleh :

A.A.SG. MAS KARUNIA MAHARANI

F1D016001

Telah diujikan di depan penguji
Pada tanggal 6 Juli 2020
Dan dinyatakan telah memenuhi syarat mencapai derajat Sarjana S-1
Program Studi Teknik Informatika

Susunan Tim Penguji :

1. Penguji 1

Tanggal: 13/07/2020

Gibran Satya Nugraha, S.Kom., M.Eng.
NIP. 199203232019031012

2. Penguji 2

Tanggal: 13/07/2020

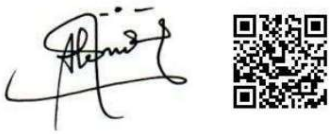
Ramaditia Dwiyansaputra, S.T., M.Eng.
NIP. -

3. Penguji 3

Tanggal: 15/07/2020

Dr. Eng. Budi Irmawati. S.Kom., MT.
NIP. 197210191999032001

Mataram, 21 Juli 2020
Dekan Fakultas Teknik
Universitas Mataram



Akmaluddin, ST., M.Sc Eng., Ph.D.
NIP. 196812311994121001

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Saya menyatakan bahwa tugas akhir ini dengan judul "Pengenalan Pola Tulisan Tangan Aksara Sasak Menggunakan Metode Linear Discriminant Analysis Dan Jaringan Syaraf Tiruan Jenis Backpropagation" sepenuhnya adalah karya sendiri. Tidak ada bagian di dalamnya yang merupakan plagiat dari karya orang lain dan saya tidak melakukan penjiplakan atau pengutipan dengan cara yang tidak sesuai dengan etika keilmuan yang berlaku. Atas pernyataan ini, saya siap menanggung resiko/sanksi yang dijatuhkan kepada saya apabila kemudian ditemukan adanya pelanggaran terhadap etika keilmuan dalam karya ini, atau ada klaim dari pihak lain terhadap keaslian karya saya ini.

Mataram, 21 Juli 2020

Yang membuat pernyataan,

A.A.Sg. Mas Karunia Maharani

PRAKATA

Om Swastiastu

Segala puji bagi Tuhan Yang Maha Esa yang telah memberikan karunia-NYA sehingga penulis dapat menyelesaikan Tugas Akhir dengan judul "Pengenalan Pola Tulisan Tangan Aksara Sasak Menggunakan Metode Linear Discriminant Analysis Dan Jaringan Syaraf Tiruan Jenis Backpropagation" tepat pada waktunya. Pada kesempatan ini pula, penulis menghaturkan terimakasih yang sebesar-besarnya kepada seluruh pihak yang telah mendukung agar terselesaikanya Tugas Akhir ini.

Penulis tentu menyadari Tugas Akhir ini masih jauh dari kata sempurna dan masih terdapat kesalahan dan kekurangan di dalamnya. Untuk itu, diharapkan kritik serta saran dari pembaca untuk Tugas Akhir ini, agar Tugas Akhir ini dapat menjadi lebih baik lagi. Penulis juga mengucapkan banyak-banyak terimakasih kepada semua pihak yang telah membantu penulis sehingga Tugas Akhir ini dapat selesai.

Demikian yang bisa disampaikan, sekali lagi terimakasih atas semua pihak yang telah membantu di dalam penyelesaian Tugas Akhir ini. Semoga Tugas Akhir ini dapat bermanfaat bagi para pembaca.

Om Shanti, shanti, shanti om.

Mataram, 21 Juli 2020

Penulis

UCAPAN TERIMAKASIH

Penulis menyadari bahwa terselesaikannya Tugas Akhir ini tentunya bukan hanya dari usaha penulis saja. Tugas Akhir ini bisa selesai tepat waktu tentunya berkat dukungan dari semua pihak yang terlibat juga. Oleh karena itu, pada kesempatan ini penulis menghaturkan terimakasih yang sebesar-besarnya kepada:

1. Kedua Orang Tua dan keluarga yang selalu memberikan dukungan dari segala lini kehidupan selama perkuliahan.
2. Bapak Prof. I Gede Pasek Suta Wijaya, S.T, M.T, D.Eng. selaku dosen pembimbing utama yang telah memberikan bimbingan dan arahan kepada penulis selama penyusunan Tugas Akhir sehingga dapat selesai dengan baik.
3. Bapak Fitri Bimantoro,S.T.,M.Kom. selaku dosen pembimbing pendamping yang telah memberikan bimbingan dan arahan kepada penulis selama penyusunan Tugas Akhir sehingga dapat selesai dengan baik.
4. Reza Rismawandi, Muhammad Naufal, Medeline Widya Andani selaku teman diskusi selama pengerjaan Tugas Akhir.
5. Frizqa Ervina, I Kadek Adi Wirawan, I Putu Teguh Putrawan dan I Gede Ketut Sadhita Putra selaku teman seperjuangan selama pengerjaan Tugas Akhir.
6. Responden sumber pengambilan data Aksara Sasak yang telah meluangkan waktunya sehingga data yang dibutuhkan menjadi data yang rampung.
7. Semua pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan do’a dan dukungan baik moril maupun materil sehingga penulis dapat menyelesaikan pembuatan Tugas Akhir dengan baik

Semoga Tuhan Yang Maha Kuasa selalu memberikan rahmat dan hidayah-Nya dan memberikan imbalan yang setimpal atas bantuan yang diberikan kepada penulis.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Aksara Sasak merupakan salah satu warisan budaya Indonesia yang termasuk ke dalam kategori *endangered language* dan perlu dilestarikan agar tidak punah. Jika tidak ada upaya pelestarian, maka Aksara Sasak akan mengalami kepunahan. Mengingat betapa pentingnya upaya pelestarian, maka dilakukan penelitian untuk mengenali pola tulisan tangan aksara Sasak. Penelitian ini bertujuan untuk melihat model yang optimal untuk mengenali pola aksara Sasak serta untuk melihat seberapa akurat metode *Linear Discriminant Analysis* sebagai fitur ekstraksi dan *Backpropagation Neural Network* sebagai metode klasifikasi untuk mengenali pola aksara sasak. *Dataset* pada penelitian ini terdiri atas tiga variasi banyaknya data yaitu 13500, 10800 dan 2700. Menurut hasil akhir penelitian didapatkan akurasi model terbaik terbaik pada parameter uji koefisien DCT 64, *eigen value* bernilai 17, jumlah *neuron* 128, jumlah *hidden layer* yaitu 2 dan *learning rate* 0.003 pada *dataset* 10800 dengan akurasi 92.20% dengan presisi 92% dan recall 92% serta waktu komputasi 250.691s. Hal ini menunjukkan bahwa metode LDA dan JST-BP dapat bekerja dengan optimal pada pengenalan pola Aksara Sasak.

Kata kunci: pengenalan pola, tulisan tangan, aksara Sasak, LDA, JST-Backpropagation.

ABSTRACT

Sasak’s script is thr one of Indonesia's cultural heritages which is endangered language category and it needs to be preserved so its not become extinct script. If there is no preservation, Sasak’s script will be close to extinction. Given the importance of conservation efforts, a reserch was carried out to recognize the Sasak’s script handwriting patterns. This reserch aims to see the optimal model for recognizing Sasak script patterns and to see how accurate the Linear Discriminant Analysis method as extraction features and Backpropagation Neural Network as a classification method for recognizing Sasak’s script patterns. The dataset in this reserch consisted of three variations amount of data which are 13500, 10800 and 2700. According to the final results, the best accuracy of the proper model was obtained in DCT coefficient test parameter 64, eigen value are 17, number of neurons are 128, the number of hidden layers are 2 and learning rate 0.003 on the 10800 dataset with 92.20% accuracy, 92% precision and 92% recall with 250,691s computing time. This shows that the LDA and ANN-BP methods can work optimally to recognize Sasak’s script pattern.

Keywords: Pattern Recognition, Handwriting, Sasak Script, LDA, Backpropagation.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aksara Sasak merupakan salah satu warisan budaya Indonesia yang perlu dilestarikan agar tidak punah. Namun penggunaan aksara Sasak makin berkurang belakangan ini. Jika tidak ada upaya untuk mengenalkan kembali aksara ini, aksara Sasak akan masuk ke dalam kategori *endangered language. Endangered languages* adalah bahasa-bahasa yang terancam punah yang tidak memiliki generasi muda sebagai penuturnya dan penutur yang fasih hanyalah kelompok generasi menengah atau dewasa [1]. Saat ini peneliti dunia sedang menggalakkan penelitian mengenai *script* atau aksara. Penelitian tersebut bertujuan untuk menjaga aksara agar tidak punah dan dapat dilestarikan sebagai warisan budaya. Sistem pendidikan juga menjadi faktor yang berpengaruh terhadap pelesatarian budaya. Salah satu aksara di Indonesia yaitu aksara Sasak sudah jarang digunakan karena tidak diajarkan lagi di sekolah-sekolah. Sehingga dikhawatirkan aksara Sasak akan mengalami kepunahan. Berbeda dengan Bali, Bali merupakan salah satu provinsi di mana pemerintah daerahnya telah mengeluarkan kebijakan menggunakan aksara Bali pada fasilitas nama jalan, prasasti peresmian gedung, sarana pariwisata, dan beberapa fasilitas umum, sehingga warisan budayanya tetap terjaga [2]. Mengingat betapa pentingnya upaya pelestarian, maka dilakukan penelitian untuk mengenali aksara Sasak. Aplikasi ini selanjutnya dapat digunakan sebagai media pembelajaran untuk mendukung pengenalan aksara Sasak.

Pengenalan pola *atau Pattern recognition* dapat diartikan sebagai kegiatan yang dilakukan untuk mengambil keputusan atau kesimpulan berdasarkan pola kompleks objek atau informasi [3]. Tujuan dari pengenalan pola adalah untuk mengklasifikasi dan mendeskripsikan pola atau objek melalui pengetahuan sifat-sifat atau ciri-ciri objek tersebut. Penelitian sebelumnya mengenai aksara Sasak menggunakan metode *moment invariant* dan SVM memiliki tingkat akurasi 89.76% untuk 63 fitur dan 92.52% untuk 112 fitur [4]. Penelitian lainnya menggunakan metode *integral projection* untuk ekstraksi fitur dan metode PCA untuk reduksi dimensi dan *neural network* sebagai metode klasifikasi aksara dengan tingkat akurasi sistem sebesar 41,38% [5]. Penelitian [6] membandingkan kinerja metode

PCA dengan LDA, hasil meunjukkan bahwa akurasi metode LDA lebih tinggi. *Linear Discriminant Analysis* (LDA) merupakan metode yang digunakan untuk mengatasi kekurangan PCA.

Penelitian pengenalan cacat pada kertas, menyatakan bahwa terdapat metode yang menghasilkan nilai *error* lebih kecil dibandingkan dengan metode PCA. Metode tersebut adalah *Linear Discriminant* Analysis (LDA) [7]. PCA memberlakukan properti statistik yang sama bagi seluruh *image training* dari berbagai obyek atau kelas, sementara LDA memberlakukan properti statistik yang terpisah untuk tiap-tiap obyek [8]. LDA mengelompokkan vektor data dari kelas yang sama dan memisahkan kelas yang berbeda sehingga hasil yang didapatkan lebih spesifik dibandingkan dengan PCA. Penelitian pengenalan pola wajah menggunakan metode LDA memiliki tingkat akurasi sebasar 80% [8]. Penelitian lainnya mengenai pengenalan garis telapak tangan dengan menggunakan metode LDA memiliki akurasi sebasar 93% [9]. LDA juga digunakan sebagai metode ekstraksi fitur pada penelitian membaca angka pada meteran air secara otomatis [10]. Selain metode LDA, penelitian ini juga menggunakan metode *neural network* dalam pengklasifikasiannya.

Neural network merupakan suatu metode yang dapat digunakan untuk mengenali pola tulisan tangan [3]. *Neural network* dimisalkan sebagai analogi sistem kerja otak manusia. Terdiri atas sebuah unit pemroses yang disebut *neuron* yang berisi penambah dan fungsi aktivasi, sejumlah bobot dan sejumlah vektor masukan. Fungsi aktivasi berguna untuk mengatur keluaran yang diberikan *neuron*. Teknik penelitian *neural network* dapat menggunakan algoritme *backpropagation*, metode ini disebut dengan *neural network backpropogation* [3].

Backpropagation memiliki keunggulan dalam menghitung pola keluaran. Jika terdapat *error*, maka bobot dalam jaringan akan diperbaharui untuk mengurangi *error* tersebut [3]. *Backpropagation* telah dikembangkan dalam beberapa penelitian, mengenai pengenalan pola aksara Jawa dengan menggunakan metode *backpropogation* memiliki tingkat akurasi 99.8% untuk data latih dan 95.81% untuk data uji [11]. Penelitian lainnya terkait pengenalan pola aksara Jawa dengan menggunakan metode *backpropogation* memiliki tingkat keakuratan yaitu sebesar 99.56% untuk data sampel berupa data pelatihan, 61.359% untuk data

sampel diluar data pelatihan dan 75% untuk data sampel data pelatihan dan di luar data pelatihan [12]. Penelitian pengenalan pola aksara Jawa menerapkan metode *thinning* serta pembagian citra menjadi 16 subcitra dengan metode perambatan-balik memiliki akurasi 75% [13].

Untuk itu, penulis mengajukan sebuah penelitian untuk merancang sebuah model pembelajaran mesin (*machine learning*) untuk mengenali pola tulisan tangan aksara Sasak dengan menerapkan metode *neural network backpropagation*. Ekstraksi ciri yang digunakan adalah *linear discriminant analysis* (LDA). Tingkat keberhasilan ditentukan oleh akurasi dari algoritme *backpropagation* yang mampu mengenali pola dari tulisan tangan aksara Sasak. Berdasarkan kegunaannya diharapkan kedua metode ini dapat diterapkan untuk mengenali pola tulisan tangan aksara Sasak sehingga ke depannya kedua metode ini dapat dikembangkan untuk sistem pembelajaran aksara Sasak.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan sebelumnya, rumusan masalah pada penelitian ini adalah sebagai berikut.

1. Bagaimana merancang metode LDA dan *backpropagation* untuk klasifikasi aksara Sasak.
2. Berapakah persentase keberhasilan pengenalan pola aksara Sasak dengan metode LDA dan *backpropagation*.

1.3 Batasan Masalah

Penelitian ini memiliki batasan-batasan masalah untuk memberikan lingkup penelitian agar lebih terfokus ketika pengerjaan. Adapun batasan masalah yang diberikan adalah sebagai berikut.

1. Objek penelitian ini hanya aksara Sasak yang berjumlah 18 karakter tanpa atribut (pasangan) nya.
2. Data citra yang akan digunakan dalam penelitian bersumber dari tulisan tangan. Tulisan tangan dibedakan menjadi dua jenis yaitu berdasarkan orang yang pernah belajar aksara Sasak dan orang yang belum pernah belajar aksara Sasak.

1.4 Tujuan

Tujuan yang diharapkan dari penelitian ini adalah sebagai berikut.

1. Untuk membuat model pengenalan pola berbasis metode LDA dan *backpropagation* untuk klasifikasi pola aksara Sasak.
2. Mengetahui akurasi keberhasilan dari metode LDA dan *backpropagation* untuk mengenali pola tulisan tangan aksara Sasak.

1.5 Manfaat

Manfaat dari penelitian ini secara umum dapat diperoleh oleh dua subjek antara lain.

1. Bagi penulis
   a. Dapat menerapkan pengetahuan selama di perkuliahan terutama pengetahuan tentang pengenalan pola.
   b. Dapat menambah pengetahuan di bidang *machine learning.*
2. Bagi pembaca
   a. Dapat mengetahui performa dari metode LDA dan *backpropagation* untuk klasifikasi pola Aksara Sasak.
   b. Dapat dijadikan sebagai rujukan untuk mengembangkan model *machine learning* agar mendapat performa yang lebih baik.
   c. Model yang dihasilkan dari penelitian ini dapat digunakan untuk pembuatan sistem pembelajaran aksara melalui *handwriting digital*.

1.6 Sistematika Penulisan

Sistematika penulisan dari penelitian ini disajikan dalam beberapa bab, antara lain sebagai berikut.

1. Bab I Pendahuluan

   Bab ini menjelaskan dasar-dasar dari penulisan laporan tugas akhir, yang terdiri dari latar belakang, rumusan masalah, batasan masalah, tujuan, serta sistematika penulisan laporan tugas akhir.
2. Bab II Tinjauan Pustaka dan Dasar Teori

   Bab ini membahas tentang penelitian-penelitian terdahulu yang mengimplementasikan metode LDA, JST-BP dan DCT serta teori-teori sebagai

referensi penulis ketika melakukan penelitian.

3. Bab III Metodologi Penelitian

   Bab ini membahas tentang dataset yang digunakan pada penelitian dan penjelasan mengenai metodologi yang digunakan untuk membangun model *machine learning* pengenalan pola tulisan aksara sasak

4. Bab IV Analisis dan Perancangan

   Pada bab ini merupakan pembahasan tentang perlakuan data, proses *preprocessing*, analisis metode, analisis model pengujian, analisis hasil terbaik dari parameter-parameter pengujian dan analisa perbandingan penggunaan *dataset.*

5. Bab V Implementasi dan Pengujian Metode

   Bab ini membahas kesimpulan- kesimpulan yang didapatkan saat proses pengujian dengan metode LDA dan *JST-BP* serta menyisipkan saran-saran yang bisa dikembangkan pada penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian mengenai pengklasifikasian suatu citra menggunakan metode *backpropagation neural network* sudah pernah dilakukan oleh beberapa peneliti dalam kurun waktu 5 tahun belakangan ini. Penelitian-penelitian sebelumnya akan dijadikan sebagai rujukan ketika pelaksanaan penelitian ini, antara lain klasifikasi aksara menggunakan metode *support vector machine* (SVM) [4] dan pengenalan pola aksara pada citra menggunakan ekstraksi fitur *integral projection* dan klasifikasi NN [5].

Penelitian tentang klasifikasi menggunakan metode *Backpropagation* sudah pernah dilakukan beberapa kali pada interval tahun 2015-2019. Penelitian-penelitian tersebut antara lain tentang klasifikasi wajah [14], klasifikasi huruf vokal [15], identifikasi otentifikasi citra tanda tangan [16], klasifikasi pola sidik jari [17], klasifikasi aksara jawa [18].

Penelitian terkait metode LDA untuk ekstraksi fitur sebelumnya telah dilakukan yaitu untuk pengenalan wajah [8], angka meteran air otomatis [10] dan pengenalan telapak tangan [9]. Metode LDA memiliki beberapa keunggulan dibandingkan metode yang lain seperti relatif mudah diimplementasikan karena hanya memiliki *co-occurrent* dan nilai rata-rata global (μ) dari *Eigen Analysis* [19]. Pada ekstraksi fitur dengan LDA, *dataset* lokasinya tetap, namun kelas yang dibentuk menjadi lebih terpisah, kondisi ini disebabkan oleh jarak antar data pelatihan dalam satu kelas menjadi lebih kecil [20].

Dari referensi yang diperoleh tersebut, semua penelitian berhasil melakukan pengenalan atau klasifikasi dengan baik. Akurasi dari tiap penelitian dapat sajikan dalam Tabel 2.1.

Tabel 2.1. Referensi penelitian sebelumnya

No.	Penulis	Judul	Keterangan	Akurasi Pengujian
1	Eka Dina Juliani Utari	Pengenalan Pola Tulisan Tangan Huruf	- *Integral projection* (*feature extraction*)	41,38%

No.	Penulis	Judul	Keterangan	Akurasi Pengujian
	(2019)	Sasak Menggunakan Metode Integral Projection dan Neural Network	dan *neural network* (*classification*) adalah dua metode yang digunakan pada penelitian ini - Terdapat 18 kelas - Skenario data latih sebanyak 900 data dan data uji 360 data	
2	Riska Yulianti (2018)	Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode Moment Invariant dan Support Vector Machine	- Terdapat 18 kelas - Data set berjumlah 2700	89.76%-92.52%
3	Bambang Widoyono (2013)	Implementasi Linear Discriminant Analysisi Dan Jaringan Syaraf Tiruan Backprogation Untuk Membaca Angkat Pada Meteran Air Secara Otomatis	- Ada 10 kelas - Data set berjumlah 900 citra - Scenario *training* dan *testing* sebesar 100% dan 98%	98%
4	Pijush Chakraborty (2013)	An Approach to Handwriting Recognition using Back-Propagation Neural Network	- Ada 26 kelas - Data set berjumlah 910 citra - Scenario *training* dan *testing* sebesar 70%-100% dan 79-91%	91%

No.	Penulis	Judul	Keterangan	Akurasi Pengujian
5	Gregorius Satia Budhi (2015)	Handwritten Javanese Character Recognition Using Several Artificial Neural Network Method	- *Dataset* terdiri atas 620 data - Menggunakan beberapa metode klasifikasi diantaranya yaitu CPN, ENN, BPNN dan kombinasi dari Chi2 dan BPPN.	CPN 71.45% BPNN 79.03% ENN 1 layer, 94.19% ENN 2 layer, 92.26% Chi2 dan BPNN 98.71%
6	Dhita Azzahra Pancorowati dan M. Arief Bustomi (2016)	Klasifikasi Pola Huruf Vokal dengan Menggunakan Jaringan Saraf Tiruan	- Ada 5 kelas - Data set berjumlah 450 citra - Scenario *training* dan *testing* sebesar 84% dan 76%	84%
7	Reza gharoie ahangar dan Mohammad Farajpoor Ahangar (2019)	Handwritten Farsi Character Recognition using Artificial Neural Network	- Ada 32 kelas - Klasifikasi dengan ANN - Data set berjumlah 250 citra	85%

No.	Penulis	Judul	Keterangan	Akurasi Pengujian
8	Shyla Afroge, Boshir Ahmed, Firoz Mahmud (2016)	Optical Character Recognition using Back Propagation Neural Network	- Data set berjumlah 558 citra - Terdapat 62 karakter	Digit numerik (0~9) 99% Huruf kapital (A~Z) 97% Huruf kecil (a~z) 96% alphanumeric characters 93%

Berdasarkan penelitian-penelitian yang sudah dilakukan sebelumnya, dapat dilihat bahwa *backpropagation* dapat bekerja dengan baik untuk pengklasifikasian citra. Meskipun memiliki beberapa kelemahan seperti hasil pelatihan yang tidak konstan dan tidak diketahui secara detail bagaimana hasil prediksi diperoleh, karena metode ini tidak dapat memberikan informasi mengenai bobot yang paling berpengaruh diantara pola inputannya, namun metode ini juga memiliki kelebihan. Kelebihan metode ini mampu memformulasikan pengalaman dan sangat fleksibel dalam perubahan aturan perkiraan [21]. LDA juga baik digunakan untuk ekstraksi fitur karena meminimalkan persebaran dalam kelas (*within class*) dan memaksimumkan persebaran antar kelas (*between class*), hal ini dapat

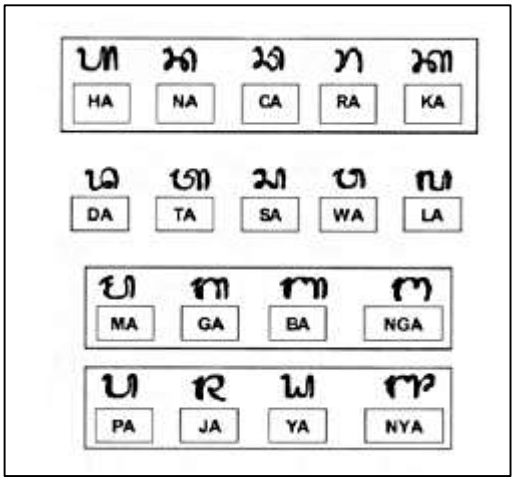
memudahkan persebaran data yang nantinya akan diolah [8]. Oleh karena itu, penulis bermaksud untuk menggunakan metode LDA sebagai metode ekstraksi fitur dan BPP sebagai metode klasifikasi citra aksara sasak.

2.2 Dasar Teori

2.2.1 Aksara Sasak

Aksara merupakan suatu simbol visual yang tertera pada suatu media (kertas, kain) untuk mengungkapkan unsur-unsur yang ekspresif dalam suatu bahasa. Aksara digunakan untuk secara khusus menuliskan bahasa daerah tertentu. Salah satu bahasa daerah nusantara yang digunakan di Lombok adalah bahasa Sasak dan ditulis dengan menggunakan aksara Sasak [22].

Aksara berfungsi sebagai media penulisan untuk membaca karya sastra kuno berbahasa daerah. Huruf sasak merupakan salah satu aksara tradisional nusantara yang digunakan oleh masyarakat suku Sasak di Lombok, Indonesia untuk menulis bahasa daerah yaitu bahasa sasak, yang terdiri dari 18 karakter dasar [23]. Semua karakter huruf sasak dapat dilihat pada *Gambar* 2.1



Gambar 2.1 Karakter aksara Sasak [4]

2.2.2 Pengenalan Pola

Pengenalan pola *(pattern recognition*) dapat diartikan sebagai proses klasifikasi dari objek atau pola menjadi beberapa kategori atau kelas dan bertujuan untuk pengambilan keputusan [24]. Tujuan dari pengenalan pola ini adalah mengklasifikasi dan mendeskripsikan pola atau obyek kompleks melalui

pengetahuan sifat-sifat atau ciri-ciri obyek tersebut. Pola adalah entitas yang terdefinisi dan dapat diberikan suatu identifikasi atau nama misalkan aksara.

Pengenalan pola pada dasarnya terdiri dari 3 langkah utama, yaitu *preprocessi*ng, ekstraksi ciri dan pengenalan. *Preprocessing* merupakan langkah awal yang dilakukan pada keseluruhan data objek yang ada agar mendapatkan hasil ciri atau fitur yang lebih baik pada tahap berikutnya. Pada tahap ini informasi yang dianggap penting akan lebih ditonjolkan, informasi tersebut adalah fitur atau ciri pada tiap objek. Tahap selanjutnya adalah ekstraksi ciri, tahap ini berfungsi untuk menemukan karakteristik pembeda yang mewakili sifat utama suatu data obyek, sekaligus mengurangi jumlah data tersebut menjadi lebih sedikit tetapi representatif. Tahap akhir yaitu pengenalan, pada tahap ini data yang ada akan dikelompokkan menjadi kelas yang sesuai [25].

2.2.3 Ekstraksi Ciri

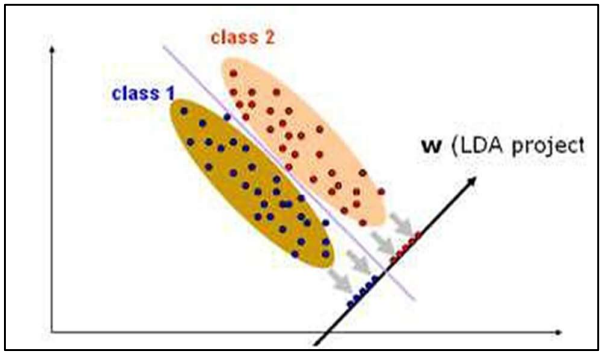
Ekstraksi ciri adalah bagian paling penting dari suatu aplikasi pengenalan pola. Proses ekstraksi ciri dilakukan untuk mendapatkan nilai yang merupakan ciri dari citra aksara. Nilai yang merupakan ciri setiap citra disebut nilai *eigen*. Pada proses ekstraksi ciri dibutuhkan nilai *eigen* dari masing-masing citra. Nilai *eigen* pada proses training disimpan pada *database,* nilai *eigen* pada proses training nantinya dibandingkan dengan nilai *eigen* pada citra masukan selanjutnya [26].

Ekstraksi fitur terkait erat dengan masalah pengurangan dimensi di mana tujuannya adalah untuk mengidentifikasi fitur dalam kumpulan data serta membuang fitur lain seperti informasi yang tidak relevan dan berlebihan [27]. Ekstraksi fitur adalah salah satu faktor yang paling penting yang dapat mempengaruhi tingkat akurasi klasifikasi karena jika *dataset* berisi fitur yang tidak penting, maka ruang dimensi akan menjadi besar serta mempengaruhi tingkat akurasi dari proses klasifikasi [28].

2.2.4 *Linear Discriminant Analysis*

LDA pertama kali diterapkan pada proses pengenalan wajah oleh Etemad dan Chellapa. Metode ini merupakan salah satu pengenalan wajah yang lebih dikenal sebagai *Fisher's Linear Discriminant*. LDA dikenal masyarakat setelah

Ronald A. Fisher sebagai penemu metode ini mempublikasikannya melalui *paper The Use of Multiple Measures in Taxonomic Problems* pada tahun 1936. LDA bekerja berdasarkan analisa matrik penyebaran (*scatter matrix analysis*) yang bertujuan menemukan suatu proyeksi optimal yang dapat memaksimumkan penyebaran antar kelas dan meminimumkan penyebaran dalam kelas data . Pada LDA terdapat perbedaan yang minimum dari citra dalam kelas. Perbedaan antar kelas direpresentasikan oleh matriks *Sb* (*scatter between class*) dan perbedaan dalam kelas direpresentasikan oleh matriks *Sw* (*scatter within class*). *Matriks covariance* didapatkan dari perhitungan kedua matriks Perhitungan matriks *Matriks covariance* berfungsi umtuk memaksimalkan jarak antar kelas dan meminimumkan jarak dalam kelas [29]. Ide dasar dari LDA adalah menemukan sebuah transformasi linear sehingga pengklasteran dapat dipisahkan setelah transformasi. Pada LDA vektor data diproyeksikan ke dalam sub ruang. Demikian pula apabila ada data uji maka akan diproyeksikan ke sub ruang yang sama dengan melakukan perkalian *eigenvector* hasil *training* dengan vektor data uji. LDA mengelompokkan vektor data dari kelas yang sama dan memisahkan kelas yang berbeda. Vektor data diproyeksi dari ruang *N* dimensi (di mana *N* ada jumlah citra aksara yang diproses) ke ruang C-1 dimensi (di mana C adalah jumlah kelas dalam vektor data). Proyeksi data dua kelas dari metode LDA dapat dilihat pada *Gambar* 2.2 .



Gambar 2.2 proyeksi data dua kelas dari metode LDA

Berikut merupakan langkah-langkah ekstraksi ciri menggunakan LDA [30]:

1. Mengubah matriks dua dimensi yang didapat dari *pixel* setiap *dataset* menjadi matriks satu dimensi atau mengubahnya ke dalam bentuk vektor baris atau kolom.

2. Data *training* yang didapat kemudian dikelompokkan ke dalam matriks sejumlah kelas (x_i).
3. Menghitung nilai rata – rata (*mean*) dari tiap – tiap kelas (μ_i) dengan Persamaan (2-1) dan jika tiap data *training* diubah ke bentuk vektor baris, maka perhitungan *mean* dimensi dihitung berdasarkan kolom. Jika data *training* diubah ke bentuk vektor kolom, maka perhitungan *mean* dimensi dihitung berdasarkan baris, sehingga nantinya jumlah *mean* dimensi yang dihasilkan akan sama dengan jumlah dimensi satu data *training* dan bukan dimensi jumlah dari *dataset*.

$$\mu_i = \frac{1}{N_i} \sum_{x \epsilon \omega_i} x \tag{2-1}$$

Dimana:

N_i : Jumlah sampel pada kelas ke-i

i : Mewakili jumlah kelas dari seluruh kelas

μ_i : Vektor rata – rata kelas ke-i

x : Sampel pada kelas

4. Menghitung nilai rata – rata (*mean*) total dari semua kelas (μ) dengan Persamaan (2-2):

$$\mu = \frac{1}{N_i + \cdots + N_c} \sum_i^c \sum_{x \in \omega_i} x_i \tag{2-2}$$

Dimana:

x_i : Sampel pada kelas ke-i

x_c : Sampel pada kelas ke-c

5. Menghitung matriks sebaran antar kelas (S_b) dengan Persamaan (2-3) dan matriks sebaran dalam kelas (S_w) dengan Persamaan (2-4):

$$S_b = \sum_{i=1}^{c} N_i \ (\mu_i - \mu)(\mu_i - \mu)^T \tag{2-3}$$

$$S_w = \sum_{i=1}^{c} \sum_{j=1}^{N_i} \left((x_j - \mu_i)(x_j - \mu_i)^T \right) \tag{2-4}$$

Dimana:

S_b : Penyebaran antar kelas

S_w : Penyebaran dalam kelas

c : Jumlah seluruh kelas

x_j : Sampel pada kelas ke-j

μ : Vektor rata – rata jumlah sampel

T : *Transpose*

6. Menghitung nilai *covariance* matriks (C) menggunakan nilai S_b dan S_w yang telah didapat dengan Persamaan (2-5):

$$C = (S_w)^{-1} * S_b \quad (2\text{-}5)$$

Dimana:

C : Matrik kovarian

7. Menghitung *eigen value* (λ) dan *eigen vecto*r (v) dengan Persamaan (2-6):

$$S_b v = \lambda S_w v \quad (2\text{-}6)$$

Dimana:

λ : *Eigen value*

v : *Eigen vector*

8. Pilih m *eigen vektor* yang memiliki *eigen value* terbesar ($m \ll n$) dan nyatakan dengan matrik W.
9. Memproyeksi citra asal dengan *eigen vector* terpilih menggunakan Persamaan (2-7):

$$p = w^T x^i \quad (2\text{-}7)$$

Dimana:

p : Proyeksi

v^T : *Vector eigen transpose*

x^i : Citra masukan

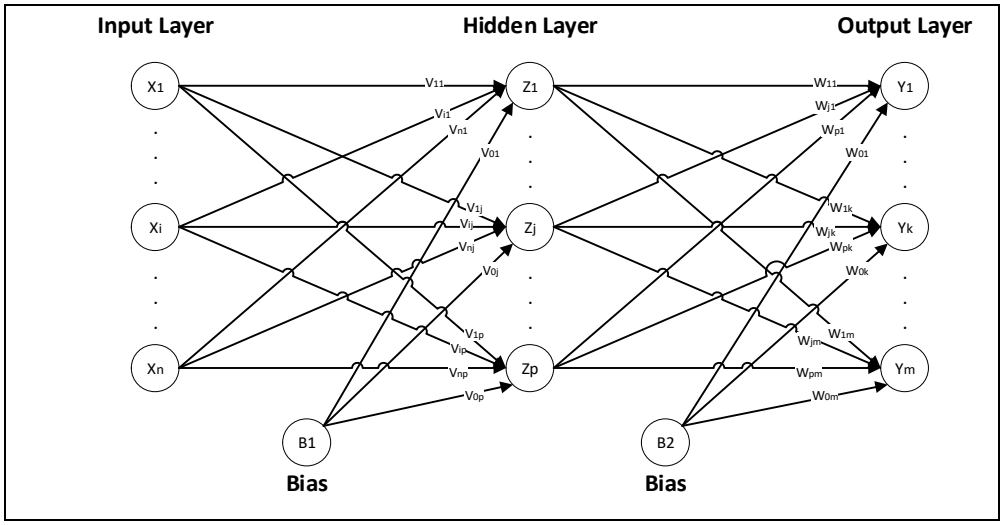
2.2.5 Jaringan Syaraf Tiruan *Backpropagation*

JST dengan *layer* tunggal memiliki keterbatasan dalam pengenalan pola. Kelemahan ini bisa ditanggulangi dengan menambahkan satu atau beberapa layar tersembunyi di antara *layer* masukan dan *layer* keluaran. Jaringan syaraf tiruan *backpropagation* (JST-BP) melatih jaringan mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola yang digunakan selama pelatihan serta kemampuan untuk memberikan respon yang benar terhadap pola masukan yang serupa dengan pola yang dipakai selama pelatihan [31].

2.2.5.1 Arsitektur *Backpropagation*

Backpropagation memiliki beberapa unit (*neuron*) yang ada dalam satu atau lebih *layer* tersembunyi. *Gambar* 2.3 adalah arsitektur *backpropagation multilayer*

dengan 1 *hidden layer*. Pada *Gambar*, unit *input* dilambangkan dengan X, unit *hidden* dilambangkan dengan Z, dan unit *output* dilambangkan dengan Y. Bobot antara unit *input* (X) dan unit *hidden* (Z) dilambangkan dengan V, sedangkan bobot antara unit *hidden* (Z) dan unit *output* (Y) dilambangkan dengan W.



Gambar 2.3 Arsitektur jaringan *backpropagation*

2.2.5.2 Fungsi Aktivasi

Dalam *backpropagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu: kontinu, terdiferensial dengan mudah, dan merupakan fungsi yang tidak turun. Salah satu fungsi yang memenuhi ketiga syarat tersebut sehingga sering dipakai adalah fungsi *sigmoid biner* yang memiliki range (0, 1) [32]. Persamaan fungsi aktivasi *sigmoid biner* yaitu sebagai berikut:

$$f(\mathrm{x}) = \frac{1}{1+e^{-x}} \tag{2-8}$$

$$f'(\mathrm{x}) = f(\mathrm{x})(1 - f(\mathrm{x})) \tag{2-9}$$

2.2.5.3 Algoritma *Backpropagation*

Algoritma pelatihan *backpropagation* terdiri dari proses *feedforward* dan *backpropagation*. Algoritma tersebut yaitu sebagai berikut [33]:

Langkah 0: Inisialisasi bobot (ambil bobot awal dengan nilai acak yang cukup kecil)

Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 sampai 9

Langkah 2: Untuk setiap pasang data pelatihan, lakukan langkah 3 sampai 8

Fase I: *Feedforward*

Langkah 3: Tiap unit masukan (x_i, i = 1,2,…, n) menerima sinyal dan

meneruskannya ke unit selanjutnya, yaitu lapisan tersembunyi

Langkah 4 : Hitung semua keluaran pada lapisan tersembunyi (Z_j, j = 1, 2,…, p)

$$Z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (2\text{-}10)$$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya:

$$Z_j = (Z\_net_j) \quad (2\text{-}11)$$

Dan kirimkan sinyal tersebut ke semua unit lapisan atasnya (unit-unit *output*). Langkah ini dilakukan sebanyak jumlah lapisan tersembunyi.

Langkah 5 : Hitung semua keluaran jaringan di lapisan *output* (Y_k, k = 1, 2,…, m)

$$Y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad (2\text{-}12)$$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya:

$$Y_k = f(y\_net_k) \quad (2\text{-}13)$$

Fase II: *Backpropagation*

Langkah 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran (y_k, k = 1, 2,…, m)

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) \quad (2\text{-}14)$$

δ merupakan unit kesalahan yang akan dipakai dalam perubahan bobot *layer* di bawahnya (langkah 7). $f'(y\_net_k)$ merupakan fungsi turunan dari fungsi aktivasi *sigmoid biner.*

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki w_{jk}) dengan laju percepatan α

$$\Delta w_{jk} = \alpha.\delta\, .z_j \quad (2\text{-}15)$$

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k})

$$\Delta w_{0k} = \alpha.\delta_k \quad (2\text{-}16)$$

Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi (z_j, j = 1, 2,…, p)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k . w_{jk} \quad (2\text{-}17)$$

Faktor δ unit tersembunyi:

$$\delta_j = \delta\_net_j\, f'(z\_net_j) \quad (2\text{-}18)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij})

$$\Delta v_{ij} = \alpha.\delta_j.x_i \quad (2\text{-}19)$$

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j})

$$\Delta v_{0j} = \alpha . \delta_j \quad (2\text{-}20)$$

Fase III: Perubahan Bobot

Langkah 8: Tiap-tiap unit *output* (Y , k = 1, 2,…, m) memperbaiki bobotnya (j = 0, 1, 2,..., p)

$$w_{jk}\ (baru) = w_{jk}\ (lama) + \Delta w_{jk} \quad (2\text{-}21)$$

Tiap-tiap unit tersembunyi (Z_j, j = 1, 2, 3,..., p) memperbaiki bobotnya (j = 0, 1, 2, 3,..., n)

$$v_{ij}\ (baru) = v_{ij}\ (lama) + \Delta v_{ij} \quad (2\text{-}22)$$

Langkah 9: Kondisi pelatihan berhenti

Ketiga fase tersebut diulang terus menerus hingga kondisi penghentian dipenuhi. Umumnya kondisi penghentian yang sering dipakai adalah jumlah iterasi atau kesalahan. Iterasi akan dihentikan jika jumlah iterasi yang dilakukan sudah melebihi jumlah maksimum iterasi yang ditetapkan, atau jika kesalahan yang terjadi sudah lebih kecil dari batas toleransi yang diijinkan [22].

2.2.6 *Discrate Cosine Transform* (DCT)

Discrete Cosine Transform (DCT) adalah sebuah teknik untuk mengubah sebuah sinyal kedalam komponen frekuensi dasar. DCT merepresentasikan sebuah citra dari penjumlahan *cosinus* dari *magnitude* dan frekuensi yang berubah-ubah. Sifat dari DCT adalah mengubah informasi citra yang signifikan dikonsentrasikan hanya pada beberapa koefisien DCT. DCT adalah sebuah skema *lossy compression* dimana NxN blok di transformasikan dari domain spasial ke domain DCT. DCT menyusun sinyal tersebut ke frekuensi spasial yang disebut dengan koefisien. Frekuensi koefisien DCT yang lebih rendah muncul pada kiri atas dari sebuah matriks DCT, dan frekuensi koefisien DCT yang lebih tinggi berada pada kanan bawah dari matriks DCT. Kelebihan dari metode DCT kemampuan menghitung kuantitas bit-bit data gambar dimana pesan tersebut disembunyikan didalamnya. Walaupun gambar yang dikompresi dengan *lossy compression* akan menimbulkan kecurigaan karena perubahan gambar terlihat jelas, pada metode ini hal ini tidak

akan terjadi karena metode ini terjadi di domain frekuensi di dalam *image*, bukan pada domain spasial, sehingga tidak akan ada perubahan yang terlihat pada *cover* gambar [34].

Penelitian ini mengaplikasikan DCT untuk mengambil fitur berdasarkan *low frequency* citra dimana *backgorund* putih pada citra aksara mendominasi matriks *input*an sehingga menghasilkan determinan bernilai 0 dan hal tersebut berakibat tidak dapat dilakukannya *eigen analysis* pada proses ekstraksi LDA. Pada penelitian [35] metode DCT dipilih karena kemampuannya yang untuk membawa informasi penting dari sinyal menuju *low frequency components*, jadi fitur yang ada pada citra dapat dibangun berdasarkan sebagian kecil persentase koefisien dominan DCT dengan rentang pengujian koefisien DCT 16~256. Berikut adalah persamaan pada proses DCT (2-23) [36].

$$C(u,v) = a(u)a(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)cos\left[\frac{(2x+1)\pi\mu}{2N}\right]cos\left[\frac{(2y+1)\pi\mu}{2N}\right] \quad (2\text{-}23)$$

Selama proses DCT berlangsung, dihitung pula invers dari 2D-DCT dengan menggunakan persamaan (2-24).

$$f(x,y) = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)cos\left[\frac{(2x+1)\pi\mu}{2N}\right]cos\left[\frac{(2y+1)\pi\mu}{2N}\right] \quad (2\text{-}24)$$

Matriks hasil transformasi DCT yang telah terisi dengan koefisien DCT selanjutnya diproses pada tahap kuantisasi, dimana data yang terletak pada kiri atas merupakan korelasi dari frekuensi-frekuensi rendah dari data asli. Sedangkan yang terletak pada kanan bawah merupakan korelasi dari frekuensi-frekuensi tinggi dari data asli. Pada tahap kuantitasi, digunakan algoritma *Huffman* yaitu dengan Menyusun bilangan menggunakan fungsi zigzag *scanning* [37]. Proses ini bertujuan untuk menentukan fitur terbaik dari citra yang selanjutnya akan diproses pada tahap klasifikasi, dimana jumlah fitur yang diambil sejumlah nilai koefisien yang digunakan.

BAB III
METODOLOGI PENELITIAN

3.1 Alat dan Bahan Penelitian

Alat dan bahan pada penelitian yang dilakukan berupa *software* dan *hardware* serta data-data yang dibutuhkan selama kegiatan.

1. Alat Penelitian

   Alat-alat yang akan digunakan dalam melakukan penelitian ini adalah sebagai berikut.

   a. Laptop ASUS ROG Intel® Core™ i7-7700HQ CPU @ 2.80GHz 2.81 GHz dengan RAM 8 GB dan GPU NVIDIA GEFORCE 1050,
   b. Sistem operasi Windows 10 PRO,
   c. Bahasa pemrograman *python*,
   d. Anaconda Jupyter Lab,
   e. Microsoft office word 2019,

2. Bahan Penelitian

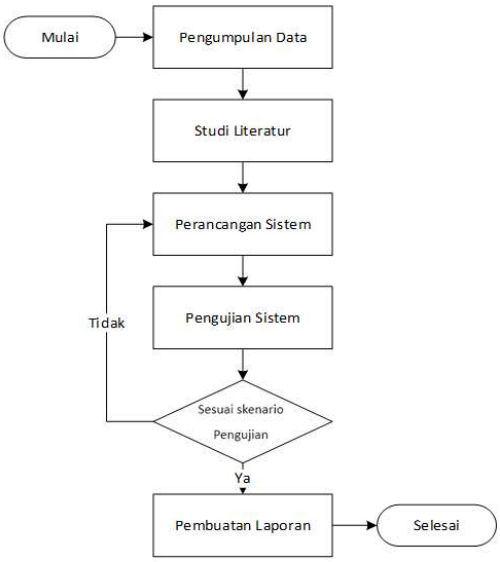
   a. Literatur-literatur dari jurnal, buku, serta penelitian-penelitian yang berkaitan dengan aksara, machine learning, *neural network*, dan metode LDA serta *backpropagation.*
   b. Data yang akan digunakan dalam penelitian ini adalah citra aksara Sasak dari *dataset* yang ditulis oleh siswa SD, SMP, SMA dan Perguruan Tinggi dengan masing-masing kategori melibatkan sebanyak 10 orang serta dibedakan berdasarkan dua jenis yaitu yang pernah mempelajari aksara sasak sebelumnya dan yang belum pernah mempelajari Aksara Sasak. Untuk proses pengambilan data satu orang memberikan kontribusi sebanyak 15 kali penulisan. Jenis *file* citra *input* adalah JPG, citra di*scan* dengan resolusi 600 dpi menggunakan *scanner Canon MP* 287. Proses *cropping* pada citra menggunakan *code* dengan mendeteksi tepi dari *template* citra dan mengatur posisi kiri, atas, kanan dan bawah dari tepi tersebut. Penggunaan *code* berguna untuk meminimalkan waktu *preprocessing* dikarenakan jumlah *dataset* yang banyak.

3.2 Studi Literatur

Studi literatur dilakukan dengan mempelajari buku-buku, jurnal-jurnal penelitian sebelumnya serta sumber lain yang berkaitan dengan permasalahan yang diangkat pada penelitian ini. Adapun materi yang dipelajari dalam studi literatur berkaitan dengan ekstraksi fitur warna dengan metode statistik, ekstraksi fitur tekstur menggunakan LDA serta klasifikasi citra menggunakan klasifikasi *Backpropagation* serta materi lain yang beraitan dengan penelitian yang dilakukan.

3.3 Rancangan Penelitian

Adapun rancangan penelitian yang akan dilakukan digambarkan dengan diagram alir pada *Gambar* 3.1.



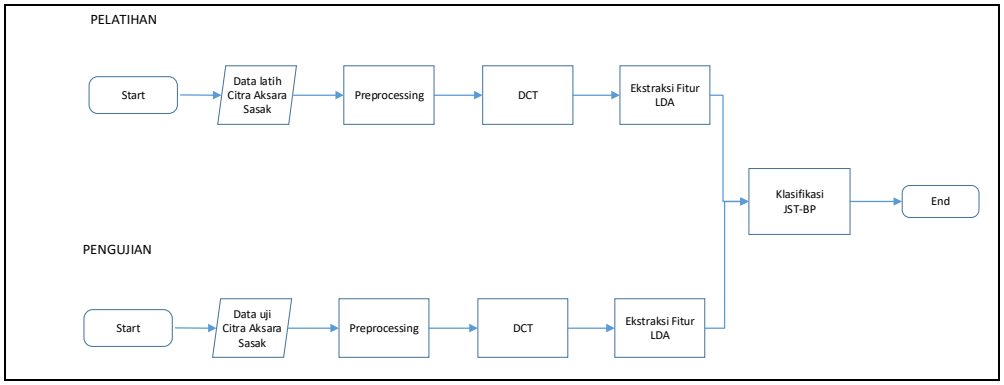
Gambar 3.1 Diagram alir proses pelatihan

Diagram alir di atas memperlihatkan bahwa proses perancangan sistem dimulai dari langkah awal yakni pengumpulan data dan langkah terakhir pembuatan laporan. Pengumpulan data disini artinya proses pengumpulan citra yakni Aksara Sasak dengan 18 karakter. Proses pengambilan citra aksara berdasarkan empat kategori, yaitu berdasarkan tulisan tangan anak-anak SD, SMP dan SMA serta tulisan tangan Mahasiswa. Pemilihan kategori responden berkaitan dengan ragam variasi model tulisan tangan. Karena pada *machine learning*, semakin beragam data yang di*train* maka proses pembelajaran pada mesin akan semakin baik dan dapat mengenali suatu tulisan dengan kemungkinan-kemungkinan yang ada. Pemilihan

empat kategori juga berkaitan dengan jenis responden yaitu yang pernah mempelajari dan belum pernah mempelajari aksara Sasak, karena berdasarkan pencarian data yang dilakukan oleh peneliti, kategori yang pernah mempelajari Aksara Sasak sebagian besar pada golongan SMA dan Kuliah. Kurikulum saat itu masih menyisipkan muatan lokal. Sementara untuk saat ini, kurikulum pembelajaran pada SD dan SMP tidak menyisipkan pelajaran muatan lokal. Langkah kedua yakni studi literatur untuk mempelajari cara membangun sistem sesuai dengan metode yang digunakan. Selanjutnya yakni tahap perancangan sistem sesuai dengan rancangan yang telah dibuat. Tahap pengujian sistem dilakukan untuk menguji skenario yang telah ditetapkan pada penelitian ini yaitu pengujian terhadap model terbaik pada JST-BP. Kinerja dari model tersebut apakah sudah berfungsi sesuai dengan tujuan. Sistem dikatakan sesuai jika sistem sudah mampu melakukan *training* pada *dataset* aksara yang ada, mampu melakukan proses klasifikasi pada suatu data baru, dan mengklasifikasikan data baru tersebut ke suatu kelas tertentu yang ada. Jika tidak, maka proses akan kembali ke tahap perancangan sistem, dan jika sesuai maka proses akan dilanjutkan ke tahap terakhir yakni pembuatan laporan.

3.4 Perancangan Algoritma

Subbab ini akan menjelaskan bagaimana sistem dirancang mulai dari tahap pelatihan sistem sampai dengan sistem depat mengklasifikasikan aksara. Blok diagram sistem terlihat pada *Gambar* 3.2.



Gambar 3.2 Blok diagram sistem

Proses pelatihan dan proses pengujian akan dijelaskan sebagai berikut:

a. Proses pelatihan

1. Citra yang di *input* ke dalam sistem merupakan citra yang di ambil secara langsung dari tulisan tangan seseorang yang di tulis dalam selebar kertas. Hasil tulisan tersebut di *scan* dalam bentuk format .jpg agar dapat di baca oleh komputer. *Gambar* yang sudah di dapatkan tersebut di *cropping* sesuai dengan banyaknya huruf aksara yaitu sebanyak 18 huruf, karena di dalam selembar kertas tersebut terdapat 18 huruf aksara yang berbeda yang akan di kelompokan ke dalam 18 folder berbeda sebagai data latih.
2. Tahap *preprocessing*, yakni tahap manipulasi *Gambar* sesuai keinginan. Karena proses *crop* dan *resize* telah dilakukan di luar sistem maka proses selanjutnya yang akan dilakukan di dalam sistem yakni proses konversi ruang warna menjadi *grayscale.*
3. Setelah masuk ke tahap *preprocessing*, data pada citra akan diolah menggunakan metode DCT untuk diambil fitur terbaik dari frekuensi rendahnya dengan koefisien dari rentang 64~256.
4. *Image* diekstraksi menggunakan metode LDA untuk mendapatkan ciri dari masing-masing citra aksara sasak dan kemudian hasil dari ekstraksi ciri tersebut akan menjadi data latih untuk system dan sebagai data inputan untuk proses klasifikasi pada *backpropagation*.
5. Proses klasifikasi yang digunakan pada penelitian ini adalah *Backpropagation Neural Network*.
6. Hasil klasifikasi tersebut di simpan oleh sistem sebagai proses pelatihan dan di sesuaikan dengan target apakah sesuai atau tidak hasil klasifikasi yang telah di jalankan oleh sistem.

b. Proses pengujian

1. *Input* citra aksara untuk klasifikasi (citra pengujian). Citra yang dimasukkan yaitu citra aksara yang telah di-*crop* dan di-*resize* di luar sistem.
2. Tahap *preprocessing* yang dilakukan di dalam sistem pada proses klasifikasi sama dengan pada proses pelatihan yakni konversi ruang warna.
3. Setelah masuk ke tahap *preprocessing*, data pada citra akan diolah menggunakan metode DCT untuk diambil fitur terbaik dari frekuensi rendahnya dengan koefisien dari rentang 64~256.

4. Ekstraksi fitur dilakukan untuk menentukan ciri dari masing-masing aksara sebanyak 18 karakter. LDA adalah metode ekstraksi fitur dan reduksi dimensi yang digunakan di dalam sistem.
5. Tahap klasifikasi dilakukan dengan metode *Backpropagation* untuk mengetahui karakter aksara sasak. Data hasil pelatihan dimuat untuk dibandingkan dengan data uji.
6. Keluaran akhir dari proses klasifikasi berupa jenis huruf dari karakter Aksara Sasak.

3.5 Data Acquisition

Data *acquisition* adalah proses pengambilan atau pengumpulan data yang akan digunakan pada proses *training* dan *testing*. Pada penelitian ini, data yang dibutuhkan adalah citra aksara sasak yang nantinya akan di proses. Pengambilan citra aksara sasak menggunakan *template* dari kolom tabel dengan *height* dan *weight* yang di *setting* 4cmx4cm dengan banyak kotak sebanyak 18 kotak atau tabel sesuai dengan jumlah karakter aksara sasak yaitu 18 buah. Kertas yang digunakan seragam yaitu HVS A4. *Template* pengambilan data tersebut menjadi tiga baris dan enam kolom yang nantinya akan ditulis oleh sumber menggunakan spidol berupa tulisan tangan.

Sumber pengambilan data tulisan tangan aksara sasak dibagi menjadi empat kategori yaitu SD, SMP, SMA dan Perguruan Tinggi. Pembagian kategori pada pengambilan data dilakukan untuk memperoleh keberagaman data dari sisi pendidikan dan pernah mempelajari aksara sasak sebelumnya. Keempat kategori dibagi lagi menjadi orang yang pernah mempelajari penulisan aksara sasak dan orang yang tidak pernah belajar penulisan aksara Sasak. Masing-masing kategori melibatkan sepuluh orang dan masing-masing orang menulis 18 karakter sebanyak 15 kali. Data yang terkumpul sebanyak 10800 citra. Data yang terkumpul lalu di scan dengan dengan resolusi tinggi.

Pada penelitian sebelumnya terdapat 2700 data yang terkumpul dengan sistematika pengambilan data yang sama yaitu dengan menggunakan tulisan tangan manual. Data ini akan digunakan sebagai data pembanding pada saat proses pelatihan dan klasifikasi citra. Tabel 3.1 menunjukkan perbedaan yang terdapat

pada kedua jenis *dataset*.

Tabel 3.1 Perbedaan yang terdapat pada kedua jenis *dataset*

Dataset 10800	Dataset 2700
• Pengambilan data berdasarkan empat kategori yaitu SD,SMP, SMA dan Perguruan Tinggi	• Pengambilan data hanya melibatkan satu kategori yaitu Perguruan Tinggi.
• Responden sebanyak 40 orang di mana masing-masing kategori terdiri atas 10 orang dan menulis karakter aksara sebanyak 15 kali. Sehingga jumlah *dataset* yaitu 10800.	• Responden sebanyak 15 orang di mana masing-masing orang menulis karakter aksara sebanyak 10 kali. Sehingga jumlah *dataset* yaitu 2700.
• Format citra yaitu JPG	• Format citra yaitu PNG
• Resolusi 600 dpi	• Resolusi 2400 dpi
• Dibedakan menjadi dua kategori yaitu pernah belajar Aksara Sasak dan belum pernah belajar Aksara Sasak.	• Tidak diberlakukan pembedaan kategori pernah dan belum pernah mempelajari Aksara Sasak.
• Size awal antara 176~200 pixel	• Size awal antara 190~230 pixel

3.6 Preprocessing

Preprocessing dilakukan untuk memperbaiki citra agar citra yang diolah memiliki hasil yang optimal, oleh karena itu pre-processing yang dilakukan di penelitan ini sebagai berikut :

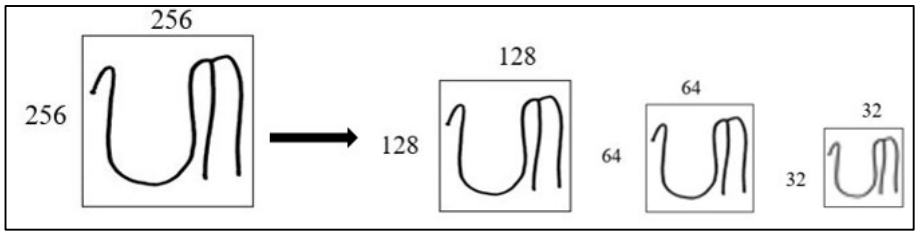
1. *Cropping*

   Cropping adalah proses pemotongan citra pada elemen tertentu pada area citra. Proses ini bertujuan untuk mengambil elemen yang diinginkan dari citra yaitu untuk memisahkan 18 huruf aksara yang ada pada 1 citra sebelumnya dan dibagi menjadi 18 citra berbeda.

2. *Resize*

   Resize merupakan proses pengubahan pixel citra. Proses ini dilakukan untuk mengubah citra pixel 256x256 menjadi 128x128, 64x64 dan 32x32 untuk mempermudah proses pada sistem dan melihat pengaruh *size* terhadap

penelitian. Semakin kecil pixel pada citra maka proses yang ada pada sistem akan lebih cepat. Proses *resize* pada citra aksara dapat dilihat pada Gambar 3.3



Gambar 3.3 Proses *resizing* citra aksara Sasak

3. *Grayscale*

   Grayscale adalah merubah citra warna menjadi citra berwana keabuan. *Grayscale* memungkinkan nilai minimal dan warna putih untuk nilai maksimal. Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna tersebut. Semakin besar jumlah bit warna yang disediakan di memori, maka semakin halus gradasi warna yang terbentuk. Sehingga pada penelitian ini dibutuhkan proses grayscale untuk meringkan kinerja sistem saat proses pelatihan dan pengujian.

4. Reduksi dimensi

   Reduksi dimensi dilakukan pada penelitian ini untuk mengubah citra 2D menjadi citra 1D. Reduksi dimensi dilakukan untuk mempermudah proses perhitungan rata-rata baris pada citra. Berikut persamaan untuk reduksi dimensi pada Persamaan (3-1)

$$S = citra_{(ixj,1)} \tag{3-1}$$

$$\text{C1} = \begin{pmatrix} 25 & 28 & 32 \\ 33 & 36 & 32 \\ 24 & 27 & 31 \end{pmatrix} \text{C2} = \begin{pmatrix} 25 & 36 & 39 \\ 30 & 30 & 32 \\ 25 & 26 & 23 \end{pmatrix} \text{C3} = \begin{pmatrix} 26 & 39 & 27 \\ 31 & 31 & 27 \\ 22 & 21 & 34 \end{pmatrix}$$

Matriks C1,C2 dan C3 direduksi menjadi 1 dimensi menjadi

$$\text{C1} = \begin{bmatrix} 25 \\ 28 \\ 32 \\ 33 \\ 36 \\ 32 \\ 24 \\ 27 \\ 31 \end{bmatrix} \text{C2} = \begin{bmatrix} 25 \\ 36 \\ 39 \\ 30 \\ 30 \\ 32 \\ 25 \\ 26 \\ 23 \end{bmatrix} \text{C2} = \begin{bmatrix} 26 \\ 39 \\ 27 \\ 31 \\ 31 \\ 27 \\ 22 \\ 21 \\ 34 \end{bmatrix}$$

Reduksi dimensi dilakukan pada semua citra yang ada pada *dataset* untuk mengubahnya menjadi 1 dimensi. Kemudian semua citra yang ada digabungkan menjadi satu agar terbentuk matriks A menggunakan Persamaan (3-2)

$$A = [Ci, Ci + 1, \dots \dots \dots \dots \dots Cn]^{T} \quad (3\text{-}2)$$

Pada Tabel 3.2 yang berisi 9 citra berbeda yang sudah di reduksi menjadi 1 dimensi disusun menggunakan Persamaan (3-2)

Tabel 3.2 Matriks A

Citra	Fitur								
c1	25	28	32	33	36	32	24	27	31
c2	25	36	39	30	30	32	25	26	23
c3	26	39	27	31	31	27	22	21	34
c4	52	67	55	53	52	67	59	55	55
c5	53	69	57	55	51	66	58	66	53
c6	55	60	56	54	51	56	50	55	64
c7	86	92	81	82	86	98	98	83	87
c8	86	93	83	92	88	96	89	94	95
c9	96	92	82	91	97	97	97	85	96

5. Normalisasi

Normalisasi adalah proses mengubah nilai agar nilai bernilai antara 0 dan 1. Hal ini dilakukan pada setiap citra untuk mempermudah perhitungan. Berikut Persamaan (3-3) untuk proses normalisasi data.

$$N_{(i,j)} = \frac{fitur_{(i,j)} - min_{(j)}}{max_{(j)} - min_{(j)}} \quad (3\text{-}3)$$

Berikut contoh perhitungan menggunakan Persamaan (3-3)

$$N_{(1,1)} = \frac{25-24}{36-24} = 0.0833$$

$$N_{(2,1)} = \frac{28-2}{36-2} = 0.3333$$

Selanjutnya setiap baris dan kolom dilakukan perhitungan seperti di atas pada matriks A. Tabel 3.3 merupakan hasil normalisasi dari matriks A.

Tabel 3.3 Normalisasi matriks A

Fitur								
0.0833	0.3333	0.6667	0.75	1	0.6667	0	0.25	0.5833
0.125	0.8125	1	0.4375	0.4375	0.5625	0.125	0.1875	0
0.2778	1	0.3333	0.5556	0.5556	0.3333	0.0556	0	0.7222
0	1	0.2	0.0667	0	1	0.4667	0.2	0.2
0.1111	1	0.3333	0.2222	0	0.8333	0.3889	0.8333	0.111
0.3571	0.7143	0.4286	0.2857	0.0714	0.4286	0	0.3571	1
0.2941	0.6471	0	0.0588	0.2941	1	1	0.1176	0.3529
0.2308	0.7692	0	0.6923	0.3846	1	0.4615	0.8462	0.9231
0.9333	0.6667	0	0.6	1	1	1	0.2	0.9333

3.7 EKSTRAKSI FITUR

1. Mengubah matriks dua dimensi yang didapat dari pixel setiap *dataset* menjadi matriks satu dimensi atau mengubahnya ke dalam bentuk vektor baris atau kolom.

$$C1 = \begin{pmatrix} 0.0833 & 0.3333 & 0.6667 \\ 0.75 & 1 & 0.6667 \\ 0 & 0.25 & 0.5833 \end{pmatrix} \qquad C2 = \begin{pmatrix} 0.125 & 0.8125 & 1 \\ 0.4375 & 0.4375 & 0.5625 \\ 0.125 & 0.1875 & 0 \end{pmatrix}$$

$$C3 = \begin{pmatrix} 0.2778 & 1 & 0.3333 \\ 0.5556 & 0.5556 & 0.3333 \\ 0.5556 & 0 & 0.7222 \end{pmatrix}$$

Menjadi,

$$C1 = \begin{bmatrix} 0.0833 \\ 0.3333 \\ 0.6667 \\ 0.75 \\ 1 \\ 0.6667 \\ 0 \\ 0.25 \\ 0.5833 \end{bmatrix} \quad C2 = \begin{bmatrix} 0.125 \\ 0.8125 \\ 1 \\ 0.4375 \\ 0.4375 \\ 0.5625 \\ 0.125 \\ 0.1875 \\ 0 \end{bmatrix} \quad C3 = \begin{bmatrix} 0.2778 \\ 1 \\ 0.3333 \\ 0.5556 \\ 0.5556 \\ 0.3333 \\ 0.5556 \\ 0 \\ 0.7222 \end{bmatrix}$$

a. Data *training* yang didapat kemudian dikelompokkan ke dalam matriks sejumlah kelas (x_i).

$$X_1=\begin{bmatrix} 0.0833 & 0.125 & 0.2778 \\ 0.3333 & 0.8125 & 1 \\ 0.6667 & 1 & 0.3333 \\ 0.7500 & 0.4375 & 0.5556 \\ 1 & 0.4375 & 0.5556 \\ 0.667 & 0.5625 & 0.3333 \\ 0 & 0.125 & 0.0556 \\ 0.25 & 0.1875 & 0 \\ 0.5833 & 0 & 0.7222 \end{bmatrix} \quad X_2=\begin{bmatrix} 0 & 0.1111 & 0.3571 \\ 1 & 1 & 0.7143 \\ 0.2 & 0.3333 & 0.4286 \\ 0.0667 & 0.2222 & 0.2857 \\ 0 & 0 & 0.0714 \\ 1 & 0.8333 & 0.4286 \\ 0.4667 & 0.3889 & 0 \\ 0.2 & 0.8333 & 0.3571 \\ 0.2 & 0.1110 & 1 \end{bmatrix}$$

$$X_3=\begin{bmatrix} 0.2941 & 0.2308 & 0.9333 \\ 0.6471 & 0.7692 & 0.6667 \\ 0 & 0 & 0 \\ 0.6471 & 0.6923 & 0.6000 \\ 0.2941 & 0.3846 & 1 \\ 1 & 1 & 1 \\ 1 & 0.4615 & 1 \\ 0.2941 & 0.8462 & 0.2000 \\ 0.3529 & 0.9231 & 0.9333 \end{bmatrix}$$

b. Menghitung nilai rata – rata (*mean*) dari tiap – tiap kelas (μ_i) dengan Persamaan (2-1) ,dengan contoh perhitungan sebagai berikut:

$$\mu_1 = \frac{\begin{bmatrix} 0.0833 \\ 0.3333 \\ 0.6667 \\ 0.75 \\ 1 \\ 0.6667 \\ 0 \\ 0.25 \\ 0.5833 \end{bmatrix} + \begin{bmatrix} 0.125 \\ 0.8125 \\ 1 \\ 0.4375 \\ 0.4375 \\ 0.5625 \\ 0.125 \\ 0.1875 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.2778 \\ 1 \\ 0.3333 \\ 0.5556 \\ 0.5556 \\ 0.3333 \\ 0.5556 \\ 0 \\ 0.7222 \end{bmatrix}}{3}$$

$$\mu_1 = \begin{bmatrix} 0.1620 \\ 0.7153 \\ 0.6667 \\ 0.5810 \\ 0.6644 \\ 0.5208 \\ 0.0602 \\ 0.1458 \\ 0.4352 \end{bmatrix} \quad \mu_2 = \begin{bmatrix} 0.1561 \\ 0.9048 \\ 0.3206 \\ 0.0238 \\ 51.333 \\ 0.7540 \\ 0.2852 \\ 0.4635 \\ 0.4370 \end{bmatrix} \quad \mu_3 = \begin{bmatrix} 0.4861 \\ 0.6943 \\ 0 \\ 0.4504 \\ 0.5596 \\ 1 \\ 0.8205 \\ 0.3879 \\ 0.7364 \end{bmatrix}$$

c. Menghitung nilai rata – rata (*mean*) total dari semua kelas (μ) *global* dengan Persamaan (3-4):

$$\mu = \frac{1}{x_i + \cdots + x_c} \sum_{x \in \omega_i} x \tag{3-4}$$

$$\mu = \frac{1}{3} + \begin{bmatrix} 0.1620 \\ 0.7153 \\ 0.6667 \\ 0.5810 \\ 0.6644 \\ 0.5208 \\ 0.0602 \\ 0.1458 \\ 0.4352 \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 0.1561 \\ 0.9048 \\ 0.3206 \\ 0.0238 \\ 51.333 \\ 0.7540 \\ 0.2852 \\ 0.4635 \\ 0.4370 \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 0.4861 \\ 0.6943 \\ 0 \\ 0.4504 \\ 0.5596 \\ 1 \\ 0.8205 \\ 0.3879 \\ 0.7364 \end{bmatrix}$$

$$\mu = \begin{bmatrix} 0.2681 \\ 0.7715 \\ 0.3291 \\ 0.4076 \\ 0.4159 \\ 0.7583 \\ 0.3886 \\ 0.3324 \\ 0.5362 \end{bmatrix}$$

d. Menghitung matriks sebaran antar kelas (S_b) dengan Persamaan (3-5) dan matriks sebaran dalam kelas (S_w) dengan Persamaan (3-6):

$$S_b = \sum_{i=1}^{c} N_i \ (\mu_i - \mu)(\mu_i - \mu)^T \tag{3-5}$$

Berikut perhitungan μ_1, μ_2 dan μ_3 dikurangi dengan (μ) *global* $tb = (\mu_i - \mu)$, lalu hasilnya akan dikalikan dengan hasil *tranpose* tb , beirkut adalah contoh perhitungannya:

$$tb = \begin{bmatrix} 0.1620 \\ 0.7153 \\ 0.6667 \\ 0.5810 \\ 0.6644 \\ 0.5208 \\ 0.0602 \\ 0.1458 \\ 0.4352 \end{bmatrix} - \begin{bmatrix} 0.2681 \\ 0.7715 \\ 0.3291 \\ 0.4076 \\ 0.4159 \\ 0.7583 \\ 0.3886 \\ 0.3324 \\ 0.5362 \end{bmatrix} \quad \begin{bmatrix} 0.1561 \\ 0.9048 \\ 0.3206 \\ 0.0238 \\ 51.333 \\ 0.7540 \\ 0.2852 \\ 0.4635 \\ 0.4370 \end{bmatrix} - \begin{bmatrix} 0.2681 \\ 0.7715 \\ 0.3291 \\ 0.4076 \\ 0.4159 \\ 0.7583 \\ 0.3886 \\ 0.3324 \\ 0.5362 \end{bmatrix} \quad \begin{bmatrix} 0.4861 \\ 0.6943 \\ 0 \\ 0.4504 \\ 0.5596 \\ 1 \\ 0.8205 \\ 0.3879 \\ 0.7364 \end{bmatrix} - \begin{bmatrix} 0.2681 \\ 0.7715 \\ 0.3291 \\ 0.4076 \\ 0.4159 \\ 0.7583 \\ 0.3886 \\ 0.3324 \\ 0.5362 \end{bmatrix}$$

$$tb = \begin{bmatrix} -0.1060 & -0.1120 & 0.2180 \\ -0.0562 & 0.1333 & -0.0771 \\ 0.3376 & -0.0085 & -0.3291 \\ 0.1734 & -0.2161 & 0.0427 \\ 0.2485 & -0.3921 & 0.1437 \\ -0.2374 & -0.0043 & 0.2417 \\ -0.3284 & -0.1034 & 0.4319 \\ -0.1866 & 0.1311 & 0.0555 \\ -0.1010 & -0.0992 & 0.2002 \end{bmatrix}$$

Matriks tb merupakan hasil dari perhitungan masing-masing μ_1, μ_2 dan μ_3 dikurangi dengan (μ) *global.* Menghasilkan matriks ordo 9x3. Hasil dari matriks tb lalu di *transpose* dan menghasilkan matriks 3x9, sehingga berdasarkan Persamaan (3-5) perhitungan S_b menghasilkan nilai matriks 9x9 .

$S_b =$

0.0713	-0.0258	-0.1066	0.0151	0.0489	0.0784	0.1406	0.0172	0.0655
-0.0258	0.0269	0.0053	-0.0418	-0.0773	-0.0059	-0.0286	0.0237	-0.0230
-0.1066	0.0053	0.2223	0.0463	0.0399	-0.1597	-0.2521	-0.0824	-0.0992
0.0151	-0.0418	0.0463	0.0786	0.1340	-0.0299	-0.0161	-0.0583	0.0125
0.0489	-0.0773	0.0399	0.1340	0.2361	-0.0226	0.0210	-0.0898	0.0426
0.0784	-0.0059	-0.1597	-0.0299	-0.0226	0.1148	0.1828	0.0572	0.0728
0.1406	-0.0286	-0.2521	-0.0161	0.0210	0.1828	0.3051	0.0717	0.1299
0.0172	0.0237	-0.0824	-0.0583	-0.0898	0.0572	0.0717	0.0551	0.0170
0.0655	-0.0230	-0.0992	0.0125	0.0426	0.0728	0.1299	0.0170	0.0601

$$S_w = \sum_{i=1}^{c} \sum_{j=1}^{N_i} \left((x_j - \mu_i)(x_j - \mu_i)^T \right) \quad (3\text{-}6)$$

tw_1 adalah hasil dari pengurangan data pada kelas x_1 dengan μ_1 pada Persamaan (3-6). x_1 adalah kelas pertama yang memiliki tiga data yaitu c_1, c_2 dan c_3 dan μ_1

adalah matriks *mean* dari kelas x_1. Hasil dari tw_1 adalah matriks 9x3. Perhitungan yang sama juga dilakukan untuk kelas x_2 yang juga memiliki tiga data dan kelas x_3 yang memiliki tiga data, sehingga menghasilkan nilai matriks tw_2 dan tw_3.

$$tw_1 = \begin{bmatrix} 0.0833 & 0.125 & 0.2778 \\ 0.3333 & 0.8125 & 1.000 \\ 0.6667 & 1.000 & 0.3333 \\ 0.7500 & 0.4375 & 0.5556 \\ 1.000 & 0.4375 & 0.5556 \\ 0.667 & 0.5625 & 0.3333 \\ 0.000 & 0.125 & 0.0556 \\ 0.250 & 0.1875 & 0.000 \\ 0.5833 & 0.000 & 0.7222 \end{bmatrix} - \begin{bmatrix} 0.1620 \\ 0.7153 \\ 0.6667 \\ 0.5810 \\ 0.6644 \\ 0.5208 \\ 0.0602 \\ 0.1458 \\ 0.4352 \end{bmatrix}$$

$$tw_1 = \begin{bmatrix} -0.0787 & -0.0370 & 0.1158 \\ -0.3820 & 0.0972 & 0.2847 \\ 0.0000 & 0.3333 & -0.3334 \\ 0.1690 & -0.1435 & -0.0254 \\ 0.3356 & -0.2269 & -0.1088 \\ 0.1459 & 0.0417 & -0.1875 \\ -0.0602 & 0.0648 & -0.0046 \\ 0.1042 & 0.0417 & -0.1458 \\ 0.1481 & -0.4352 & 0.2870 \end{bmatrix}$$

$$tw_2 = \begin{bmatrix} -0.1561 & -0.0450 & 0.2010 \\ 0.0952 & 0.0952 & -0.1905 \\ -0.1206 & 0.0127 & 0.1080 \\ -0.1248 & 0.0307 & 0.0942 \\ -0.0238 & -0.0238 & 0.0476 \\ 0.2460 & 0.0793 & -0.3254 \\ 0.1815 & 0.1037 & -0.2852 \\ -0.2635 & 0.3698 & -0.1064 \\ -0.2370 & -0.3260 & 0.5630 \end{bmatrix}$$

$$tw_3 = \begin{bmatrix} -0.1920 & -0.2553 & 0.4472 \\ -0.0472 & 0.0749 & -0.0276 \\ 0 & 0 & 0 \\ -0.3916 & 0.2419 & 0.1496 \\ -0.2655 & -0.1750 & 0.4404 \\ 0 & 0 & 0 \\ 0.1795 & -0.3590 & 0.1795 \\ -0.2703 & 0.4583 & -0.1879 \\ -0.3835 & 0.1867 & 0.1969 \end{bmatrix}$$

Berdasarkan Persamaan (3-6) perhitungan S_w merupakan pencarian nilai matriks dari *scatter within* menghasilkan nilai tw_1. Hasil tw_1 selanjutnya dikalikan dengan hasil *transpose*nya. Begitupula dengan tw_2 dan tw_3 dikalikan dengan masing-

masing *transpose*nya. Hasil dari ketiga perkalian dijumlahkan sesuai dengan rumus *sigma*, sehingga menghasilkan S_w matriks ordo 9x9.

$S_w =$

0.3898	-0.0204	-0.0110	0.1064	0.2763	-0.1421	0.0489	-0.1727	0.3165
-0.0204	0.2994	-0.0934	-0.0802	-0.2076	-0.0121	0.0692	0.0054	-0.1514
-0.0110	-0.0934	0.2486	-0.0137	-0.0316	0.0126	-0.0282	0.0875	-0.1555
0.1064	-0.0802	-0.0137	0.3094	0.2263	-0.0355	-0.1960	0.2381	0.3776
0.2763	-0.2076	-0.0316	0.2263	0.4744	0.0367	0.0395	-0.0574	0.3133
-0.1421	-0.0121	0.0126	-0.0355	0.0367	0.2309	0.1405	0.0434	-0.3177
0.0489	0.0692	-0.0282	-0.1960	0.0395	0.1405	0.3262	-0.2288	-0.3763
-0.1727	0.0054	0.0875	0.2381	-0.0574	0.0434	-0.2288	0.5698	-0.0103
0.3165	-0.1514	-0.1555	0.3776	0.3133	-0.3177	-0.3763	-0.0103	0.9938

e. Menghitung nilai *covariance* matriks © menggunakan nilai S_b dan S_w yang telah didapat dengan Persamaan (3-7):

$$C = (S_w)^{-1} * S_b \tag{3-7}$$

Setelah mendapatkan nilai S_w dan S_b , dilanjutkan dengan pencarian nilai dari *covariance* matriks dengan menginverskan matriks S_w dan dikalikan dengan matriks S_b, sehingga menghasilkan matriks *covariance* ordo 9x9.

$C_=$

-2.0927	-0.3833	5.2880	1.9184	2.4385	-3.7581	-5.5656	-2.4467	-1.9659
1.6961	-0.1158	-3.4777	-0.6711	-0.5274	2.5001	3.9718	1.2566	1.5767
1.1896	0.0041	-2.6009	-0.6474	-0.6597	1.8627	2.8936	1.0267	1.1092
-1.5102	0.9830	1.4294	-1.2259	-2.5198	-1.1003	-2.4244	0.3797	-1.3695
2.4173	-0.1459	-4.9928	-0.9960	-0.8165	3.5877	5.6848	1.8235	2.2479
-1.2429	-0.6329	3.9084	1.9792	2.8250	-2.7504	-3.8176	-2.1433	-1.1834
0.3808	0.4264	-1.6380	-1.0881	-1.6553	1.1401	1.4635	1.0526	0.3717
0.3017	-0.3267	-0.0387	0.5149	0.9460	0.0531	0.3197	-0.2978	0.2685
0.6728	-0.2667	-0.9613	0.1912	0.5407	0.7093	1.2965	0.1225	0.6168

f. Menghitung *eigen value* (λ) dan *eigen vecto*r (v) dengan Persamaan (3-8):

$$S_b v = \lambda S_w v \tag{3-8}$$

Didapatkan matriks *eigen vecto*r (v) dengan ordo 9x9 dan matriks *eigen value* (λ) dengan ordo 9x9. Pemilihan *eigen value* berdasarkan nilai terbesar dari diagonal matriks λ, dimana pada matriks λ nilai terbesar berada pada baris ke satu, dua dan tiga.

$v =$

```
 0.5695 + 0.0000i  -0.1120 + 0.0000i   0.1253 + 0.0000i   0.4955 + 0.0000i   0.4955 + 0.0000i
-0.3360 + 0.0000i   0.3095 + 0.0000i  -0.6113 + 0.0000i  -0.2601 - 0.1912i  -0.2601 + 0.1912i
-0.2608 + 0.0000i   0.1734 + 0.0000i   0.0321 + 0.0000i   0.2941 + 0.1812i   0.2941 - 0.1812i
 0.0400 + 0.0000i  -0.7269 + 0.0000i   0.5060 + 0.0000i  -0.1578 - 0.4350i  -0.1578 + 0.4350i
-0.4844 + 0.0000i   0.4314 + 0.0000i  -0.5461 + 0.0000i  -0.0978 + 0.2304i  -0.0978 - 0.2304i
 0.4576 + 0.0000i   0.1413 + 0.0000i  -0.0222 + 0.0000i   0.0499 + 0.2668i   0.0499 - 0.2668i
-0.2087 + 0.0000i  -0.1625 + 0.0000i   0.0927 + 0.0000i  -0.0907 - 0.2226i  -0.0907 + 0.2226i
 0.0304 + 0.0000i   0.2121 + 0.0000i  -0.1273 + 0.0000i   0.1213 + 0.2032i   0.1213 - 0.2032i
-0.0683 + 0.0000i   0.2360 + 0.0000i  -0.1731 + 0.0000i   0.0491 + 0.2535i   0.0491 - 0.2535i

-0.2567 - 0.0352i  -0.2567 + 0.0352i   0.0492 - 0.2599i   0.0492 + 0.2599i
 0.5090 + 0.0000i   0.5090 + 0.0000i  -0.1085 + 0.2990i  -0.1085 - 0.2990i
-0.2245 - 0.2443i  -0.2245 + 0.2443i   0.2146 - 0.1962i   0.2146 + 0.1962i
 0.0416 - 0.1954i   0.0416 + 0.1954i   0.1826 + 0.2370i   0.1826 - 0.2370i
 0.2955 + 0.1449i   0.2955 - 0.1449i  -0.2586 + 0.0771i  -0.2586 - 0.0771i
 0.2761 - 0.2079i   0.2761 + 0.2079i   0.2372 - 0.0501i   0.2372 + 0.0501i
-0.2699 + 0.1270i  -0.2699 - 0.1270i  -0.2573 + 0.0033i  -0.2573 - 0.0033i
 0.0817 - 0.1492i   0.0817 + 0.1492i   0.0061 + 0.0834i   0.0061 - 0.0834i
 0.1116 - 0.4070i   0.1116 + 0.4070i   0.6708 + 0.0000i   0.6708 + 0.0000i
```

$\lambda =$

```
-7.1026 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i  -0.7171 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 - 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i

 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
-0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 + 0.0000i   0.0000 + 0.0000i
 0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 0.0000i
```

Pada matriks d_1 nilai dari diagonal *eigen value* terbesar ada pada baris ke satu, kedua dan ketiga, maka *eigen vecto*r dari nilai tersebut disusun di w menjadi matriks ordo 9x3.w nantinya akan di *transpose* menjadi ordo 3x9 . Hasil dari w^T dikali dengan data dari x_1, x_2 dan x_3 akan menjadi matriks proyeksi ekstark fitur data training pada kelas x_1, x_2 dan x_3.

$$d_1 \begin{bmatrix} 7.1026 \\ 0.7171 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix} w = \begin{bmatrix} 0.5695 & -0.1120 & 0.1253 \\ -0.3360 & 0.3095 & -0.6113 \\ -0.2608 & 0.1734 & 0.0321 \\ 0.0400 & -0.7269 & 0.5060 \\ -0.4844 & 0.4314 & -0.5461 \\ 0.4576 & 0.1413 & -0.0222 \\ -0.2087 & -0.1625 & 0.0927 \\ 0.0304 & 0.2121 & -0.1273 \\ -0.0683 & 0.2360 & -0.1731 \end{bmatrix}$$

g. Memproyeksi citra asal dengan nilai *eigen value* berdasarkan *eigen vector* terpilih menggunakan Persamaan (3-9):

$$p = w^T x^i \tag{3-9}$$

$$\text{Proyeksi}_1 = \begin{bmatrix} -0.4200 & -0.4200 & -0.4200 \\ 0.3805 & 0.3805 & 0.3805 \\ -0.4862 & -0.4913 & -0.7154 \end{bmatrix}$$

$$\text{Proyeksi}_2 = \begin{bmatrix} -0.0328 & -0.0328 & -0.0328 \\ 0.4508 & 0.4508 & 0.4508 \\ -0.6102 & -0.5820 & -0.5007 \end{bmatrix}$$

$$\text{Proyeksi}_3 = \begin{bmatrix} 0.0384 & 0.0384 & 0.0384 \\ 0.3385 & 0.3385 & 0.3385 \\ -0.4952 & -0.5480 & -0.6497 \end{bmatrix}$$

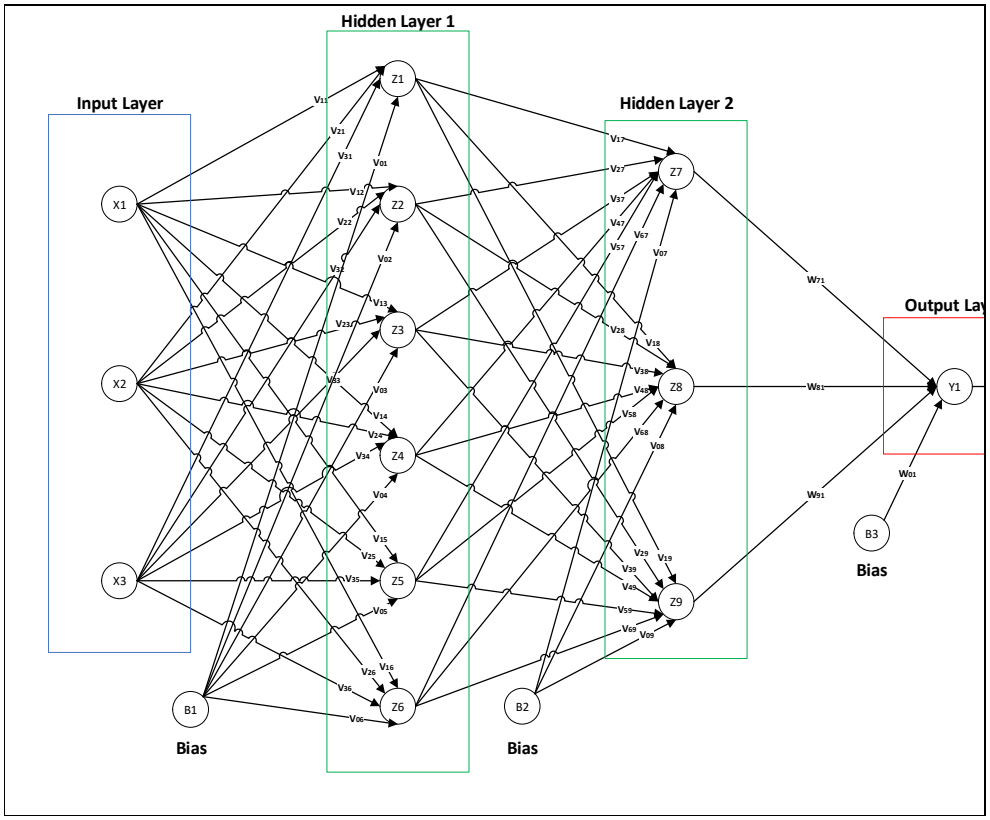
3.8 Klasifikasi

Proses klasifikasi yang digunakan pada peneliatian ini yaitu menggunakan metode *backpropagation*. Pengklasifikasian dengan menggunakan metode *backpropagation* pada penelitian ini akan menghasilkan *output* berupa klasifikasi dari 18 karakter aksara sasak. *Input* dari *backpropagation* merupakan *output* dari proses ekstraksi fitur menggunakan metode *LDA.* Berikut diberikan contoh data pada Tabel 3.4 yang terdiri dari empat data dengan tiga inputan serta terdapat target untuk masing-masing data, selanjutnya akan dilakukan proses perhitungan backpropagation sesuai dengan contoh data tersebut.

Tabel 3.4 Contoh Data Ekstraksi Fitur

No	X1	X2	X3	Target
1	1	1	0	0
2	0	1	1	1
3	1	0	1	1
4	0	0	1	0

Contoh data pelatihan data ini dapat dilihat arsitektur *backpropagation* pada *Gambar* 3.4



Gambar 3.4 Arsitektur *Backpropagation*

Arsitektur *backpropagation* yang digunakan untuk contoh dapat dilihat pada *Gambar* 3.4 . *Backpropagation* yang terdiri dari 1 *input layer*, 2 *hidden layer* dan 1 *layer output*. Di dalam *layer input* terdapat 3 *neuron*, sedangkan di dalam *hidden layer* pertama terdapat 6 *neuron* , *hidden layer* kedua terdaat 3 *neuron* dan 1 *neuron* di *layer output*. Pelatihan ini memiliki beberapa ketentuan untuk contoh metode seperti berikut :

1. Batas *error* = 0.0001
2. Batas *epoch* = 15000
3. *Learning rate* = 0.01
4. Aktivasi *sigmoid biner*

Proses klasifikasi menggunakan metode *backpropagation* memiliki 10 langkah termasuk tahap inisialisasi yang terbagi dalam 3 fase yaitu *feed forward, backpropagation* dan *update* bobot. Berikut tahapan proses yang dilakukan pada penelitian ini.

Langkah 0 : inisialisasi bobot awal yang ada pada Tabel 3.5

Tabel 3.5 Bias dan bobot awal dari *input layer* ke *hidden layer* pertama

DARI/KE	BIAS (B1)	x1	x2	x3
Z1	0,140	0,124	0,086	0,074
Z2	-0,204	-0,221	-0,740	0,210
Z3	0,160	-0,0084	-0,111	-0,162
Z4	-0,185	0,110	-0,105	0,070
Z5	0,158	-0,273	-0,070	-0,819
Z6	-0,116	-0,028	-0,068	-0,028

Selanjutnya untuk bias dan bobot awal dari *hidden layer* pertama ke *hidden layer kedua* dinyatakan dengan V01 sampai V36 dan bias ke hidden layer pertama dinyatakan dengan V01 sampai V06. Sedangkan untuk bias dan bobot dari hidden layer pertama ke hidden layer kedua dinyatakan V09 sampai V69 pada Tabel di bawah ini :

Tabel 3.6 Bias dan bobot awal dari *hidden layer* pertama ke *hidden layer* kedua

DARI/KE	Bias (B2)	Z1	Z2	Z3	Z4	Z5	Z6
Z7	-0,022	0,171	-0,141	0,042	0,045	-0,032	0,022
Z8	0,003	-0,018	0,141	-0,073	0,018	0,004	0,017
Z9	-0,083	-0,014	0,011	-0,028	0,099	0,058	0,082

Bias dan bobot awal dari *hidden layer* kedua ke *output layer* dinyatakan dengan W_{01}, $W_{71,}$ W_{81} dan $W_{91,}$ yaitu:

Tabel 3.7 Bias dan bobot awal dari *hidden layer* kedua ke *output layer*

DARI/KE	Bias (B3)	Z7	Z8	Z9
Y1	0,18	0,035	0,078	0,114

Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 sampai 9. Kondisi penghentian terpenuhi jika *error* < 0.0001 atau *epoch* > 15000.

Langkah 2: Untuk setiap pasang data pelatihan, lakukan langkah 3 sampai 8

Fase I: ***Feedforward***

Langkah 3: Tiap unit masukan (x_i, i = 1,2,…, n) menerima sinyal dan meneruskannya ke unit selanjutnya, yaitu lapisan tersembunyi. Yang digunakan pada contoh ini adalah data pertama dengan $X_1 = 1$, $X_2 = 1$ dan $X_3=0$.

Langkah 4 : Hitung semua keluaran pada lapisan tersembunyi (Z_j, j = 1, 2,…, p) menggunakan Persamaan dibawah ini

$$Z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad (3\text{-}10)$$

berikut perhitungannya :

$Z\_net_1 = 0.14 + (0* 0.124)+ (0 * 0.086) + (1 * 0.074)$

$= 0.214$

$Z\_net_2 = -0.204+ (0* -0.221) + (0* -0.74) + (1 * 0.21)$

$= 0.006$

$Z\_net_3 = 0.16 + (0* -0.0084) + (0* -0.111) + (1 * -0.162)$

$= -0.002$

$Z\_net_4 = -0.185 + (0*0.11) + (0* -0.105) + (1 * 0.07)$

$= -0.115$

$Z\_net_5 = 0.158 + (0* -0.273) + (0* -0.07) + (1 * -0.819)$

$= -0.661$

$Z\_net_6 = -0.116 + (0* -0.028) + (0* -0.068) + (1 * -0.028)$

$= -0.144$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya dengan Persamaan (3-11) di bawah ini :

$$Z_j = f'(Z\_) \quad (3\text{-}11)$$

$Z_1 = \frac{1}{1+e^{-0.214}} = 0.5532967569$

$Z_2 = \frac{1}{1+e^{-0.006}} = 0.5014999955$

$Z_3 = \frac{1}{1+e^{-(-0.002)}} = 0.4995000002$

$Z_4 = \frac{1}{1+e^{-(-0.115)}} = 0.4712816430$

$Z_5 = \frac{1}{1+e^{-(-0.661)}} = 0.3405150112$

$Z_6 = \frac{1}{1+e^{-(-0.144)}} = 0.4640620793$

Dan kirimkan sinyal tersebut ke semua unit lapisan pada *layer hidden* kedua (*neuron-neuron*).

Z_net_7 = -0.022 + (0.5532967569 * 0.171) + (0.5014999955 * -0.141) + (0.4995000002 * 0.042) + (0.4712816430 * 0.045) + (0.3405150112 * -0.032) + (0.4640620793 * 0.22)

= 0,0434018054

Z_net_8 = -0.003 + (0.5532967569 * -0.018) + (0.5014999955* 0.141) + (0.4995000002* -0.073)+ (0.4712816430* 0.018) + (0.3405150112* 0.004) + (0.4640620793 * 0.017)

= 0,0450228427

Z_net_9 `= -0.083 + (0.5532967569 * -0.014) + (0.5014999955* 0.011) + (0.4995000002* -0.028)+ (0.4712816430* 0.099) + (0.3405150112* 0.058) + (0.4640620793 * 0.082)

= 0,0052441892

Gunakan fungsi aktivasi *sigmoid biner* untuk menghitung sinyal keluaran *hidden layer* kedua.

$Z_7 = \frac{1}{1+e^{-0.043401}} - 1 \quad = 0{,}5108487484$

$Z_8 = \frac{1}{1+e^{-0.045022}} - 1 \quad = 0{,}5112538097$

$Z_9 = \frac{1}{1+e^{-0.005244}} - 1 \quad = 0{,}5013110443$

Langkah ini dilakukan sebanyak jumlah lapisan tersembunyi.

Langkah 5 : Hitung semua keluaran jaringan di lapisan *output* (Y_k, k = 1, 2,…, m) menggunakan Persamaan di bawah ini,

$$Y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \qquad (3\text{-}12)$$

Y_net_1 = 0.18.+ (0*0.035) + (0* 0.078) + (0* 0.114)

= 0,2278188278

Gunakan fungsi aktivasi sigmoid biner untuk menghitung sinyal *output*-nya:

$Y_1 = \frac{1}{1+e^{-0,22781}} = 1{,}7962685085$

Fase II: *Backpropagation*

Langkah 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran (y_k, k = 1, 2,…, m) menggunakan Persamaan (3-13) sebagai berikut :

$$\delta_1 = (t_1 - y_1)\, f'(y\_net_1) \quad (3\text{-}13)$$

= (1 – 1,7962685085) f' (0,2278188278)

= -1,4303120462

δ merupakan unit kesalahan yang akan dipakai dalam perubahan bobot *layer* di bawahnya (langkah 7). Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki w_{jk}) dengan laju percepatan (*learning rate*) dengan Persamaan (3-14)

$$\Delta w_{jk} = \alpha.\mathrm{W}_j \quad (3\text{-}14)$$

berkut perhitungannya :

ΔW_{71} = 0.01 * -1,4303120462 * 0,5108487484 = -0,0073067312

ΔW_{81} = 0.01 * -1,4303120462 * 0,5112538097 = -0,0073125248

ΔW_{91} = 0.01 * -1,4303120462 * 0,5013110443 = -0,0071703123

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}) menggunakan Persamaan di bawah ini :

$$\Delta \mathrm{v}_{0k} = \alpha.\delta_k \quad (3\text{-}15)$$

berikut perhitungannya :

ΔW_{01} = 0.01 * -1,4303120462 = -0,0143031205

Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi (z_j, j = 1, 2,…, p) menggunakan Persamaan di bawah ini

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k.w_{jk} \quad (3\text{-}16)$$

berikut perhitungannya :

δ_net_7 = -1,4303120462 * 0,0434018054 = -0,0643967143

δ_net_8 = -1,4303120462 * 0,0450228427 = -0,0643967143

δ_net_9 = -1,4303120462 * 0,0052441892 = -0,0075008269

Faktor δ unit tersembunyi pada hidden layer pertama dihit ung menggunakan

Persamaan di bawah ini

$$\delta_j = \delta\_net_j f'(\text{Z}\_net_j) \quad (3\text{-}17)$$

berikut perhitungannya :

δ_7 = -0,0620781251 * f' (0,0434018054)

= -0,0051613529

δ_8 = -0,0643967143 * f' (0,0450228427)

= -0,0055452376

δ_9 = -0,0075008269 * f' (0,0052441892)

= -0,0000782604

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}) dari input layer ke hidden layer pertama menggunakan Persamaan di bawah ini

$$\Delta v_{ij} = \alpha.\delta_j.x_i \quad (3\text{-}18)$$

berikut perhitungannya :

ΔV_{17} = 0.01 * -0.0051613529 * 0.2140000000 = -0.0000285576

ΔV_{27} = 0.01 * -0.0051613529 * 0.0060000000 = -0.0000258842

ΔV_{37} = 0.01 * -0.0051613529 * -0.0020000000 = -0.0000257810

ΔV_{47} = 0.01 * -0.0051613529 * -0.1150000000 = -0.0000243245

ΔV_{57} = 0.01 * -0.0051613529 * -0.6610000000 = -0.0000175752

ΔV_{67} = 0.01 * -0.0051613529 * -0.1440000000 = -0.0000239519

ΔV_{18} = 0.01 * -0.0055452376 * 0.21400000000 = -0.0000306816

ΔV_{28} = 0.01 * -0.0055452376 * 0.0060000000 = -0.0000278094

ΔV_{38} = 0.01 * -0.0055452376 * -0.0020000000 = -0.0000276985

ΔV_{48} = 0.01 * -0.0055452376 * -0.1150000000 = -0.0000261337

ΔV_{58} = 0.01 * -0.0055452376 * -0.6610000000 = -0.0000188824

ΔV_{68} = 0.01 *-0.0055452376 * -0.1440000000 = -0.0027726188

ΔV_{19} = 0.01 * -0.0000782604 * 0.2140000000 = -0.0000004330

ΔV_{29} = 0.01 * -0.0000782604 * 0.0060000000 = -0.0000003925

ΔV_{39} = 0.01 * -0.0000782604 * -0.0020000000 = -0.0000003909

ΔV_{49} = 0.01 * -0.0000782604 * -0.1150000000 = -0.0000003688

ΔV_{59} = 0.01 * -0.0000782604 * -0.6610000000 = -0.0000002665

ΔV_{69} = 0.01 * -0.0000782604 * -0.1440000000 = -0.0000003632

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}) menggunakan Persamaan di bawah ini,

$$\Delta v_{0j} = \alpha . \delta_j \quad (3\text{-}19)$$

berikut perhitungannya :

ΔV_{07} = 0.01 * - 0,0051613529

= -0,0000535413

$\Delta V0_8$ = 0.01 * - 0,0055452376

= -0,0000554524

$\Delta V0_9$ = 0.01 * - 0,0000782604

= -0,0000007826

Kemudian, hitung faktor δ unit tersembunyi *hidden layer* pertama berdasarkan kesalahan di setiap unit tersembunyi (Z_j, j = 1, 2, …, p) menggunakan Persamaan (2-19)

δ_net_1 = (-0,0000535413* 0,1710000000) + (-0,0000554524*-0,0180000000) + (-0,0000007826*-0,0140000000)

= -0,0000078168

δ_net_2 = (-0,0000516135*-0,1410000000) + (-0,0000554524*0,1410000000) + (-0,0000007826*0,0110000000)

= -0,0000005499

δ_net_3 = (-0.0000516135*0,0420000000) + (-0.0000554524*-0,0730000000) + (-0.0000007826*-0,0280000000)

= 0,0000019022

δ_net_4 = (-0.0000516135* 0,0450000000) + (-0.0000554524*0,0180000000) + (-0.0000007826*0,0990000000)

= -0,0000033982

δ_net_5 = (-0.0000516135*-0,0320000000) + (-0.0000554524*0,0040000000) + (-0.0000007826*0,0580000000)

= 0,0000013844

δ_net_6 = (-0.0000516135*0,0220000000) + (-0.0000554524*0,0170000000) + (-0.0000007826*0,0820000000)

= -0,0000021424

Faktor δ unit tersembunyi *hidden layer* pertama dihitung menggunakan Persamaan (3-20):

δ_1 = $\delta\_net_1 f'$(Z_net_1)
= -0.0000078168* f' (0.2140000000)
= 0.0000114060

δ_2 = -0.0000005499* f' (0.0060000000)
= 0.0000010899

δ_3 = 0.0000019022* f' (-0.0020000000)
= -0.0000038158

δ_4 = -0.0000033982* f' (-0.1150000000)
= 0.0000080894

δ_5 = 0.0000013844* f' (-0.6610000000)
= -0.0000078742

δ_6 = -0.0000021424* f' (-0.1440000000)
= 0.0000053316

Fase III: Perubahan Bobot

Langkah 8: Tiap-tiap unit *output* (, k = 1, 2,…, m) memperbaiki bobotnya (j = 0, 1, 2,..., p) menggunakan Persamaan di bawah ini

$$w_{jk}\ (baru) = w_{jk}\ (lama) + \Delta w_{jk} \quad (3\text{-}21)$$

berikut perhitungannya,

W_{01} ($baru$) = 0,1800000000 + (-0,0143031205) = 0,1656968795

W_{71} ($baru$) = 0,0350000000 + (-0,0073067312) = 0,0276932688

W_{81} ($baru$) = 0,0780000000 + (-0,0073125248) = 0,0706874752

W_{91} ($baru$) = 0,1140000000 + (-0,0071703123) = 0,1068296877

Tiap-tiap unit tersembunyi (Z_j, j = 1, 2, 3,..., p) memperbaiki bobotnya (j = 0, 1, 2, 3,..., n) menggunakan Persamaan di bawah ini

$$v_{ij}\ (baru) = v_{ij}\ (lama) + \Delta v_i \quad (3\text{-}22)$$

ΔV_{01}(baru) = 0,1400000000 + 0,0000001141 = 0.004101

ΔV_{02}(baru) = -0,2040000000 + 0,0000000109 = -0,2039999891

ΔV_{03} (baru) = 0,1600000000 + (-0,0000000382) = 0,1599999618

ΔV_{04}(baru) = -0,1850000000 + 0,0000000809 = -0,1849999191

ΔV_{05}(baru) = 0,1580000000 + (-0,0000000787) = 0,1579999213

ΔV_{06}(baru) = -0,1160000000 + 0,0000000533 = -0,1159999467

Perhitungan dilakukan sampai mendapatkan niai V_{36} baru.

Langkah 9: Kondisi pelatihan berhenti jika *error* <= 0.0001 atau jumlah *epoch* mencapai 15000.

Python digunakan untuk pembuatan jaringan *backpropagation* yang sesuai dengan inisialisasi awal dan parameter dari pelatihan manual yang telah dilakukan. Sehingga diperoleh bias dan bobot akhir yang dapat dilihat pada Tabel 3.8 dan Tabel 3.9.

Tabel 3.8 Bias dan bobot akhir dari *input layer* ke *hidden layer* pertama

Dari- / Ke-	Bias (B1)	X1	X2	X3
Z1	0.21	2.14	2.14	4.33
Z2	-0.56	-0.67	-0.76	-1.5
Z3	0.22	0.53	0.4	0.92
Z4	-1.99	-2.03	-2.04	-4.12
Z5	-1.02	-1.09	-1.11	-2.31
Z6	-0.96	-1.09	-1.1	-2.24

Tabel 3.9 Bias dan bobot akhir dari *hidden layer* pertama ke *hidden layer kedua*

Dari/Ke	Bias (B2)	Z1	Z2	Z3	Z4	Z5	Z6
Z7	-1.24	-4.72	1.03	-0.93	3.69	1.83	1.57
Z8	1.11	4	-0.77	0.84	-3.28	-1.46	-1.63
Z9	1.09	3.98	-0.85	0.95	-3.37	-1.53	-1.42

Tabel 3.10 Bias dan bobot akhir dari *hidden layer* kedua ke *output layer*

Dari- / Ke-	Bias (B2)	Z7	Z8	Z9
Y	2.19	13.00	-8.83	-8.94

Dari hasl yang sudah di dapatkan sebelumnya, nilai akan dibulatkan ke integer terdekat (*treshold*) dan menghasilkan Tabel 3.11.

Tabel 3.11 *Output* data latih

Data ke-	***Output***	**Hasil *threshold***	**Target**
1	0,0000365000	0	0
2	0,9999667000	1	1
3	0,9999133000	1	1
4	0,0000313000	0	0

Untuk mendapatkan tingkat akurasi dari hasil pelatihan, jumlah hasil *threshold* yang sesuai target dibagi dengan jumlah data. Tingkat akurasi = (4/4) * 100 % = 100 %.

3.9 Teknik Pengujian Sistem

Tahapan pengujian sistem merupakan tahapan untuk mencoba sistem apakah berjalan dengan baik dan benar serta untuk mengetahui kekurangan sistem jika terjadi kesalahan dalam proses pengujian. Dalam pengujian sistem dilakukan perhitungan akurasi dengan rumus sebagai berikut :

$$Akurasi = \frac{juml \quad data\ sesuai\ target}{Total\ keseluruha \quad data} \tag{3-23}$$

$$Presisi = \frac{jumla \quad data\ yang\ sesuai\ target\ di\ satu\ kelas}{jumlah\ seluruh\ data\ yang\ sesuai\ target} \tag{3-24}$$

$$Recall = \frac{jumla \quad data\ yang\ sesuai\ targ \quad di\ satu\ kelas}{jumlah\ data\ di\ satu\ kelas} \tag{3-25}$$

Pada Tabel 3.12 Data *dummy* sebagai contoh untuk perhitungan pengujian sistem.

Actual value / Predict value	HA	NA	CA	RA	KA
HA	10	0	0	2	0
NA	0	12	0	0	0
CA	1	1	7	2	1
RA	0	0	0	9	3
KA	1	2	3	6	0

Berdasarkan Persamaan (3-23) berikut adalah perhitungan akurasi berdasarkan data *dummy* pada Tabel 3.12.

$$Akurasi = \frac{10+12+7+9}{60} \text{ x } 100\%$$

$$= 63\%$$

Untuk menentukan presisi dari data *dummy* pada Tabel 3.12 dapat dihitung menggunakan Persamaan (3-24), dimana jumlah data yang sesuai target di satu kelas dibagi dengan jumlah seluruh data yang sesuai target.

$$Presisi_{HA} = \frac{10}{12} \qquad Presisi_{NA} = \frac{12}{12}$$
$$= 0.83 \qquad = 1$$
$$= 83\ \% \qquad = 100\%$$

Berdasarkan Persamaan (3-24), perhitungan *recall* menggunakan nilai target yang sesuai tiap kelas di bagi dengan jumlah data di kelas tersebut.

$$Recall_{HA} = \frac{10}{12} \qquad Recall_{NA} = \frac{12}{14}$$
$$= 0.83 \qquad = 0.8$$
$$= 83\ \% \qquad = 80\ \%$$

3.10 Skenario pengujian sistem

Pada sistem pengenalan pola aksara digunakan metode *backpropagation* untuk tahap pengujiannya, untuk mengetahui pengaruh *size* citra jika ukuran *pixel* dibuat berbeda yaitu dengan ukuran 128x128, 64x64 dan 32x32. Scenario uji

tersebut digunakan untuk melihat apakah proses pngujian akan lebih lama karena akuran *pixel* citra berbeda dan memiliki pengaruh terhadap akurasi uji atau tidak. Selain itu, terdapat juga beberapa parameter yang nantinya digunakan dalam proses ekstraksi fitur dan klasifikasi, yaitu :

- Jumlah *hidden layer* : 1, 2, 3 *layer* (ditentukan pada proses *trial* dan *error*).
- *Learning rate* sebagai parameter uji : 0.1~0.5
- Batas *epoch* sebagai parameter uji : 1000
- Batas *error* sebagai parameter uji : 0.001
- Pemilihan *eigen value* untuk hasil ekstraksi sebagai parameter uji sesuai dengan *trial* dan *error* pada percobaan.
- *Neuron output*: 18 *neuron.*
- Fungsi aktivasi *sigmoid biner.*
- Dalam penentuan presisi data latih dan data uji digunakan pendekatan *K-Fold Cross Validation*. Proses pengujian menggunakan *K-Fold Cross Validation* dengan menggunakan total data sampel yaitu 15x40x18= 10800 data sampel dimana K yang digunakan sebanyak 10 *K-Fold*. Jadi pada tiap *fold t*erdiri atas 180 data karakter aksara. Tabel 3.13 menunjukkan tahapan pengujian dengan menggunakan *K-Fold Cross Validation*.

Tabel 3.13 Tahap pengujian *k-fold*

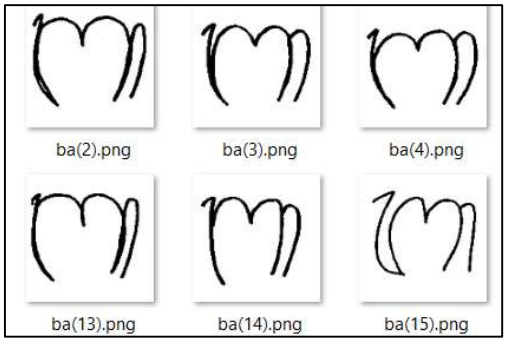
Tahap 1	Tahap 2	Tahap 3	Tahap 4	Tahap 5
Fold1 *test*	Fold2 *test*	Fold3 *test*	Fold4 *test*	Fold5 *test*
Fold2 *train*	Fold3 *train*	Fold4 *train*	Fold5 *train*	Fold6 *train*
Fold3 *train*	Fold4 *train*	Fold5 *train*	Fold6 *train*	Fold7*train*
Fold4 *train*	Fold5 *train*	Fold6 *train*	Fold7 *train*	Fold8 *train*
Fold5 *train*	Fold6 *train*	Fold7 *train*	Fold8 *train*	Fold9 *train*
Fold6 *train*	Fold7 *train*	Fold8 *train*	Fold9 *train*	Fold10 *train*
Fold7 *train*	Fold8 *train*	Fold9 *train*	Fold10*train*	Fold1 *train*
Fold8 *train*	Fold9 *train*	Fold10*train*	Fold1 *train*	Fold2 *train*
Fold9 *train*	Fold10*train*	Fold1 *train*	Fold2 *train*	Fold3 *train*
Fold10 *train*	Fold1 *train*	Fold2 *train*	Fold3 *train*	Fold4 *train*
Tahap 6	Tahap 7	Tahap 8	Tahap 9	Tahap 10
Fold6 *test*	Fold7 *test*	Fold8 *test*	Fold9 *test*	Fold10 *test*
Fold7 *train*	Fold8 *train*	Fold9 *train*	Fold10 *train*	Fold1 *train*
Fold8 *train*	Fold9 *train*	Fold10 *train*	Fold1 *train*	Fold2 *train*
Fold9 *train*	Fold10 *train*	Fold1 *train*	Fold2 *train*	Fold3 *train*
Fold10 *train*	Fold1 *train*	Fold2 *train*	Fold3 *train*	Fold4 *train*
Fold1 *train*	Fold2 *train*	Fold3 *train*	Fold4 *train*	Fold5 *train*
Fold2 *train*	Fold3 *train*	Fold4 *train*	Fold5 *train*	Fold6 *train*
Fold3 *train*	Fold4 *train*	Fold5 *train*	Fold6 *train*	Fold7 *train*
Fold4 *train*	Fold5 *train*	Fold6 *train*	Fold7 *train*	Fold8 *train*
Fold5 *train*	Fold6 *train*	Fold5 *train*	Fold8 *train*	Fold9 *train*

Beberapa variasi pengujian akan dilakukan untuk mengetahui performa dari teknik klasifikasi, presisi dan *recall*

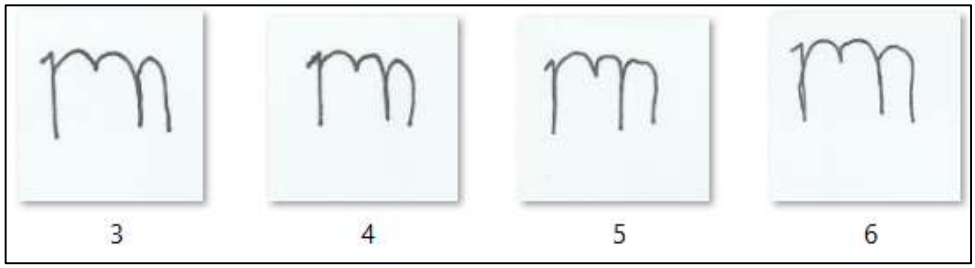
.BAB IV
HASIL DAN PEMBAHASAN

4.1. Pengumpulan *Dataset*

Pada penelitian ini, pembuatan model *machine learning* menggunakan dua jenis *dataset*. *Dataset* pertama merupakan *dataset* citra aksara sasak yang diambil oleh peneliti dari empat kategori sampel yaitu SD, SMP, SMA dan Perguruan Tinggi. Pembagian kategori pada pengambilan data dilakukan untuk memperoleh keberagaman data sampel. Keempat kategori dibagi lagi menjadi orang yang pernah mempelajari penulisan aksara sasak dan orang yang tidak pernah belajar penulisan aksara Sasak. Masing-masing kategori pendidikan melibatkan sepuluh orang dan masing-masing orang menulis 18 karakter sebanyak 15 kali. *Dataset* yang terkumpul sebanyak 10800 data. Gambar 4.1 merupakan contoh *dataset* citra aksara sasak yang diambil oleh peneliti. Kemudian kedua yaitu bersumber dari penelitian sebelumnya yang melibatkan kategori pendidikan Perguruan Tinggi sebanyak 15 orang dan masing-masing orang menulis 18 karakter sebanyak 10 kali. Sehingga *dataset* berjumlah 2700 data. Gambar 4.2 adalah contoh citra aksara sasak dari penelitian sebelumnya. Kedua jenis *dataset* diambil dengan cara manual yaitu ditulis oleh sumber menggunakan spidol. Perbedaan mendetail pada kedua *dataset* telah dijelaskan pada Tabel 3.1.



Gambar 4.1 Contoh citra Aksara Sasak yang diambil oleh peneliti.



Gambar 4.2 Contoh citra Aksara Sasak dari penelitian sebelumnya[3].

4.2. Mekanisme penelitian

Ada beberapa tahapan yang dilakukan pada penelitian ini yaitu yang pertama adalah pencarian model terbaik untuk tahap ekstraksi dan klasifikasi citra *dataset* yang digunakan untuk pencarian model ini adalah *dataset* 10800. Selanjutnya dilakukan pengujian terhadap tiga jenis *dataset* berdasarkan model terbaik yang telah didapatkan. Ketiga jenis *dataset* tersebut yaitu *dataset* 10800, *dataset* 2700 dan *dataset* gabungan dari data 10800 dan 2700. Tahapan ini yang dilakukan adalah membandingkan hasil dari ketiga jenis *dataset* yaitu *dataset* yang diambil oleh peneliti dan *dataset* yang diperoleh dari penelitian sebelumnya serta *dataset* gabungan dari kedua jenis *dataset* yang digunakan. Pengujian ini dilakukan untuk melihat pengaruh perlakuan data yang berbeda pada tulisan aksara sasak dengan karakter yang sama. Perbedaan dari kedua *dataset* yaitu berdasarkan kertas yang digunakan, *template* pengambilan data serta perbedaan jenis *scanner* dan resolusi *scanner* yang digunakan. Pada penelitian ini, *scanner* yang digunakan yaitu *Canon MP* 287 dengan resolusi 600 dpi, sementara *dataset* pada penelitian sebelumnya menggunakan *scanner Canon LiDE* 120 dengan resolusi 2400 dpi.

Pengujian terhadap model *machine learning* yang dibangun menggunakan *dataset* yang sudah disebutkan sebelumnya. Pengujian ini dilakukan dengan berbagai parameter dengan urutan pengujian yang ditunjukkan pada Tabel 4.1.

Tabel 4.1 Parameter pengujian pada penelitian ini

No	Parameter Pengujian
1.	Nilai *eigen value* pada proses ekstraksi ciri di LDA (100%,75% dan 50%)
2.	Jumlah *hidden layer neural network* (1HL, 2HL dan 3HL)
3.	Ukuran *neuron hidden layer* (32 *neuron*, 64 *neuron* dan 96 *neuron*)

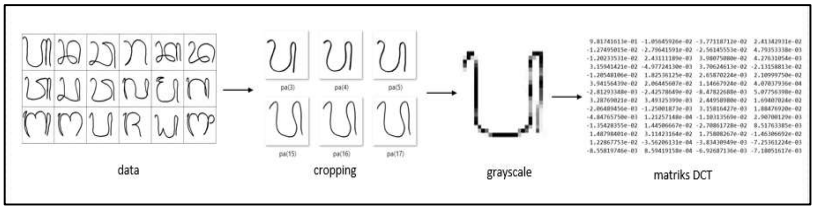
No	Parameter Pengujian
4.	Ukuran citra masukan (32x32, 64x64 dan 128x128)
5.	*Learning rates* (0.1~0.5)

Model terbaik selanjutnya diujikan menggunakan *dataset* pada penelitian sebelumnya yaitu sebanyak 2700 data serta *dataset* gabungan sebanyak 13500 data.

4.3. *Preprocessing*

Pada penelitian ini proses preprocessing yang dilakukan yaitu *cropping, resize*, *grayscale*, reduksi dimensi dan melibatkan proses DCT. Proses DCT harus dilakukan karena jika tidak *eigen analysis* pada proses ekstraksi fitur tidak dapat dilakukan, hal ini dikarenakan matriks yang dihasilkan yaitu matriks singular. Ukuran citra yang digunakan yaitu 128x128, 64x64 dan 32x32 pixel. Penggunaan metode DCT untuk menghindari matriks *scatter* S_b dan S_w pada proses ekstraksi fitur LDA bersifat singular.

Tahap *preproccesing* pada citra aksara sasak yaitu dilakukan proses *cropping*, *grayscale, resizing*, normalisasi dan reduksi dimensi seperti pada Gambar 4.3. Tahapan *cropping* dilakukan dengan menggunakan *coding* yang disesuaikan dengan *template* saat pengambilan data. Penggunaan *coding* dikarenakan jumlah *dataset* yang banyak dan untuk meminimalisir waktu saat proses *preprocessing*. Proses *cropping* dilakukan dengan deteksi tepi kotak *template* Aksara Sasak. Tepi tersebut di-*setting* menjadi warna putih dengan mengatur ukuran kiri, kanan atas dan bawah kotak *template* dengan ukuran *pixel* tertentu agar tepi tersebut hilang. Proses pengaturan harus dilakukan dengan baik agar citra yang diinginkan tidak terpotong. Selanjutnya yaitu *load dataset* menggunakan *method* "imread" dari *library* "opencv". Selanjutnya, citra masuk ke proses *grayscle* dan *resize* untuk mendapatkan ukuran citra yang diinginkan. Adapun variasi ukuran citra yang digunakan dalam pengujian yaitu 32×32 , 64×64 , 128×128 pixel *grayscale*.

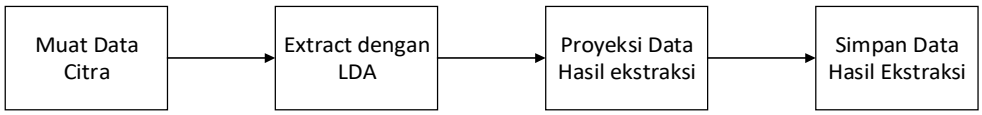


Gambar 4.3 Proses *preprocessing dataset* Aksara Sasak

Tambahan DCT sebagai *preprocessing* untuk menghindari hasil matriks singular dari matriks *scatter* S_b dan $S_w$pada proses ekstraksi fitur LDA. Matriks singular adalah matriks yang tidak mempunyai determinan yang mengakibatkan *eigen analysis* pada proses LDA tidak dapat dilakukan. Matriks singular terjadi karena *mean* atau rata-rata dari citra yang diproses menghasilkan nilai satu. Akibat dari citra pada aksara sasak memiliki dominan *backgroud* putih dibandingkan dengan pola aksara yang tertulis. Berdasarkan penelitian [35] DCT dipilih karena memiliki kemampuan yang baik untuk mengumpulkan informasi fitur penting pada frekuensi rendahnya. DCT memiliki kelebihan yaitu fitur pada citra dapat dibangun menggunakan persentase kecil dari koefisien DCT pada komponen frekuensi rendahnya. Pemilihan koefisien pada penelitian ini yaitu dengan rentang 64~256 koefisien, karena dengan jumlah tersebut citra hasil rekonstruksi masih mengandung informasi dominannya. Tahapan normalisasi citra dan reduksi dimensi dari dua dimensi menjadi satu dimensi diproses dalam *metod* DCT. Pemilihan koefisien DCT dan *eigen analysis* LDA menggunakan metode klasifikasi KNN. Pemilihan metode KNN untuk mempermudah mengetahui akurasi dari koefisien DCT dan *eigen value* sebelum masuk ke metode JST-BP, karena unruk mendapatkan arsitektur terbaik dari JST-BP harus mengetahui DCT dan *eigen value* terbaik dari masing-masing ukuran citra *input*. Setelah mendapatkan nilai koefisien DCT dan *eigen value* yang sesuai, pengujian selanjutnya dilanjutkan dengan mencari arsitektur JST-BP terbaik untuk pengujian- pengujian selanjutnya.

4.4. Ekstraksi Fitur

Citra hasil dari tahapan *preprocessing* selanjutnya akan masuk ke tahap ekstraksi fitur LDA. Pada Gambar 4.4 merupakan proses ekstaksi fitur LDA. Jumlah citra yang digunakan yaitu 10800 dengan 18 kelas. Hasil dari ekstraksi ciri LDA yaitu matriks proyeksi yang disusun dari *eigen vektor* yang bersesuaian dengan *eigan value* terbesarnya. Pemilihan *eigen value* sangat berpengaruh terhadap akurasi. Pada pengujian ini akan dipilih *eigen vektor* yang memiliki *eigen value* terbesar secara berjenjang yaitu 100%,75% dan 50% dari jumlah *eigen value*.



Gambar 4.4 Proses ekstraksi fitur LDA

Pengujian ini dilakukan untuk mengetahui jumlah *eigen vetor* terbaik berdasarkan *eigen value* terbesarnya untuk ekstraski fitur LDA. Pada pengujian ini digunakan 10800 citra dengan metode KNN. Pemilihan metode KNN untuk mempermudah mengetahui akurasi *eigen value* sebelum masuk ke metode JST-BP, karena unruk mendapatkan arsitektur terbaik dari JST-BP harus mengetahui DCT dan *eigen value* terbaik dari masing-masing ukuran citra *input*. Setelah mendapatkan nilai koefisien DCT dan *eigen value* yang sesuai, pengujian selanjutnya dilanjutkan dengan mencari arsitektur JST-BP terbaik untuk pengujian-pengujian selanjutnya.

4.5. *Training*

Pada Tabel 4.1, skenario pengujian yang akan dilakukan pada penelitian ini terdiri atas dua tahap yaitu pengujian parameter terbaik untuk menemukan arsitektur terbaik dan pengujian model tersebut terhadap ketiga jenis *dataset*. *Dataset* yang telah di ekstrak oleh LDA menghasilkan matriks proyeksi dan fitur citra yang selanjutnya masuk ke tahap klasifikasi KNN untuk mencari koefisien DCT dan *eigen analysis* LDA, lalu masuk ke proses pembentukan model JST-BP. Melihat data yang cukup besar maka rasio data *training* terhadap data *testing* yang digunakan pada pengujian paramater adalah 70:30 dengan kombinasi citra *training* dan citra *testing* untuk setiap kelasnya berjumlah sama (*balance*) *yang* di-*split* dengan "sklearn.model_selection". Sementara untuk pengujian model menggunakan *cross validation* dengan perbandingan 90:10. Hal ini dikarenakan untuk meminimalisir terjadinya *overlapping* atau tumpang tindih data pada saat pengujian. Hasil *testing* dan *training* pada *cross validation* nantinya akan dirata-ratakan sehingga didapatkan rata-rata dari 10 fold.

Proses *training* pada model JST-BP yang dibangun menggunakan *epoch* sebanyak 1000 dengan batch sebesar 100. Tiap *epoch* (iterasi), citra pada kelompok *training* diambil per 100 citra untuk *training*nya. Pada tahap *training*, dilakukan

perhitungan *cost* dari *weight* yang digunakan pada model. Nilai dari *cost* inilah yang menjadi bahan evaluasi model untuk melakukan optimisasi dengan cara memperbaharui nilai *weight* pada model. Sehingga pada tiap *epoch*-nya, model mengalami perbaikan dari sisi keberhasilannya mengklasifikasikan citra. Mekanisme optimisasi yang digunakan pada model ini adalah *Adam Optimizer. Adam* adalah algoritme pengoptimalan yang merupakan pembaharuan dari *stochastic gradient descent* (SGD) klasik untuk memperbarui *weight network* secara iteratif berdasarkan data *training* dan dapat meningkatkan kinerja proses SGD [39] .

4.6. *Testing*

Proses *testing* merupakan proses untuk menguji model yang didapatkan pada proses *training*. *Testing* dilakukan di tiap-tiap *epoch* setelah semua citra *training* selesai di *training*. *Testing* model menggunakan citra yang masuk ke dalam kelompok data *testing* sesuai dengan teknik pengujian metode *kfold cross validation.* Teknik *kflod* ini dilakukan dengan cara membagi seluruh citra pada *dataset* ke dalam 10 folder dengan perbandingan 90:10. Visualisasinya dapat dilihat pada Gambar 4.5 Selanjutnya untuk menentukan model yang terbaik akan diambil berdasarkan hasil rata-rata tiap *fold* pada k-fold *cross validation*.

Fold *testing* Fold *training*

f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10

Gambar 4.5 Mekanisme Kflod *cross validation*

4.7. Pengaruh DCT Terhadap Akurasi

Untuk pengujian ini, citra yang diujikan sebanyak 10800 citra dengan menggunakan citra ukuran 128x128 dengan koefisien awal 256. Citra hasil *preprocessing* selanjutnya masuk ke tahap ekstraksi fitur LDA. Setelah fitur dari citra didapatkan maka akan masuk ke tahap klasifikasi. Rasio citra perbandingan *training* dan *testing* adalah 70:30 dengan cara *split* data. Setelah itu citra masuk ke tahap klasifikasi dengan menggunakan metode klasifikasi KNN. KNN yang hal ini dilakukan untuk mempermudah mengetahui akurasi dari koefisien DCT dan *eigen value* sebelum masuk ke metode JST-BP. Berdasarkan penelitian [38] metode KNN memiliki akurasi tertinggi pada pengujian koefisien K bernilai 1, oleh karena itu koefisien K yang digunakan pada penelitian ini adalah 1. Hasil pengujian ditunjukkan pada Tabel 4.2 yang merupakan perbandingan jumlah koefisien DCT yang dipilih terhadap akurasi yang diberikan untuk 3 jenis variasi ukuran citra.

Tabel 4.2 Perbandingan akurasi terhadap jumlah koefisien DCT untuk 3 variasi ukuran citra

Size / DCT koef	Akurasi Testing(%)		
	128	64	32
64	90.92	90.24	85.92
81	90.49	90.58	86.14
100	90.86	89.84	85.77
121	90.37	90.06	83.98
144	84.72	89.07	84.07
169	89.04	88.42	83.58
196	89.38	88.76	82.87
256	86.57	86.32	84.07

Berdasarkan Tabel 4.1, citra dengan ukuran 128×128 memiliki akurasi tertinggi dengan koefisien DCT 64, yaitu 90.92%. Untuk citra dengan ukuran 64×64 dan 32×32 koefisien DCT 81 yang menghasilkan akurasi tertinggi yaitu pada dengan akurasi 90.58% dan 86.14%, %, hal ini sangat sejalan dengan hasil penelitian [35], bahwa dengan DCT citra dapat direpresentasikan dengan sebagian kecil koefisiennya yang mengandung informasi dominannya. Berdasarkan hasil akurasi tertinggi dari koefisien DCT terhadap masing-masing citra, maka koefisien dengan akurasi tertinggi dipilih sebagai ukuran koefisien untuk skenario pengujian selanjutnya.

4.8. Pengujian Jumlah Eigen Value

Pengujian jumlah *eigen value* pada penelitian ini bertujuan untuk mengetahui jumlah *eigen* terbaik untuk proyeksi data *input* menjadi ciri LDA. Caranya adalah dengan memilih *eigen vector* yang diperoleh dari *eigen analysis* yang bersesuaian dengan *eigen* value terbesar, kemudian memproyeksikannya ke dalam ruang *eigen* untuk setiap data masukan. Ukuran citra awal yang digunakan yaitu 128x128 dengan koefisien DCT 64, sesuai dengan hasil akurasi koefisien DCT tertinggi. Selanjutnya variasi beberapa pemilihan jumlah eigen pada tiga ukuran citra yang berbeda disimulasikan dengan teknik klasifikasi KNN sama seperti pada proses pada metode DCT dan diperoleh hasil uji seperti pada Tabel Tabel 4.3.

Tabel 4.3 Pengujian nilai *eigen* terhadap *size* citra

Size / Eigen value	Akurasi Testing(%)		
	128	64	32
17	90.92	90.58	86.14
12	89.87	90.09	84.81
9	88.51	87.80	81.88

Pada Tabel 4.3 nilai jumlah *eigen value* terbaik untuk klasifikasi Aksara Sasak adalah 17 untuk ketiga ukuran citra yang diujikan yaitu 128×128, 64×64 dan 32×32 dengan akurasi sebesar 90.92%, 90.58% dan 86.14%. Hal ini menunjukkan bahwa *eigen value* terbaik merupakan $n - 1$ dari kelas dan hasil ini sesuai dengan penelitian [40] dimana *eigen value* terbaik yaitu $n - 1$ dari kelasnya. Pada penelitian ini terdapat 18 kelas sesuai dengan jumlah karakter Aksara Sasak. Berdasarkan akurasi tertinggi yang didapatkan pada *eigen value* 17, maka skenario pengujian selanjutnya menggunakan *eigen value* dengan nilai 17.

4.9. Uji Pengaruh *Hidden Layer*

Jumlah *hidden layer* dalam arsitektur *neural network* tidak memiliki ketetapan yang pasti. Semakin sedikit atau semakin banyaknya jumlah *hidden layer* yang digunakan dalam suatu model belum tentu menambah akurasi dari model yang dibangun. Pengujian menggunakan citra dengan ukuran 128x128 dengan koefisien DCT 64, *eigen value* 17 dan *learning rate* 0.001. Oleh karena itu, pada pengujian ini penelitian menggunakan 3 variasi dengan jumlah *neuron*nya tetap yaitu sebagai

parameter uji dengan hasil seperti pada pada Tabel 4.4.

Tabel 4.4 Pengujian jumlah *hidden layer*

Jumlah HL	Akurasi Testing (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
1	81.23	81.24	81.23	175.109
2	91.72	91.73	91.73	247.941
3	91.45	91.46	91.45	323.919

Berdasarkan hasil pengujian pada Tabel 4.4, didapatkan bahwa akurasi *testing* model dengan 2 *hidden layer* memiliki performa yang lebih baik dibandingkan dengan 1 atau3 *hidden layer*. Model dengan arsitektur 2 *hidden layer* mampu memberikan performa akurasi sebesar 91.72% dengan waktu komputasi 247.941 s sedangkan model dengan arsitektur 1 dan 3 *hidden layer* menunjukkan performa akurasi pada angka 81.23% dengan waktu komputasi 175.109 s dan 91.45% dengan waktu komputasi 323.919 s. Maka dari itu dipilih 2 *hidden layer* karena memiliki akurasi tertinggi dengan waktu komputasi relatif lebih cepat dibandingkan dengan tiga *hidden layer*. Selanjutnya akan diujikan arsitektur 2 *hidden layer* untuk mencari ukuran *neuron* terbaiknya.

4.10. Uji Pengaruh Neuron

Tujuan dari pengujian ini untuk mengetahui pengaruh jumlah *neuron* berbeda terhadap performa proses klasifikasi pada *hidden layer* satu, *hidden layer* dua dan *hidden layer* tiga. Pada pengujian ini bentuk dari jaringan Neural Network yang dibangun yakni terdiri dari jumlah neuron 128, 96 dan 64. Ukuran citra yang digunakan yaitu 128x128 dengan koefisien DCT 64, *eigen value* 17 dan *learning rate* 0.001. Ketiga *neuron* dikombinasikan dan diujikan sesuai dengan Tabel 4.5.

Tabel 4.5 Pengujian jumlah *neuron* berbeda pada *hidden layer*

Percobaan ke -n	Jumlah Neuron			Akurasi (%)	Presisi (%)	Recall (%)	Time (s)
	Hidden Layer 1	Hidden Layer 2	Hidden Layer 3				
tahap 1	128	128	96	90.23	90.22	90.23	377.690
tahap 2	128	96	96	89.90	89.92	89.91	357.723
tahap 3	128	128	64	90.18	90.17	90.18	354.095
tahap 4	128	64	64	89.69	89.67	89.67	237.610
tahap 5	128	96	64	89.11	89.10	89.10	311.608
tahap 6	128	96	32	88.78	88.79	88.79	246.026

Percobaan ke -n	Jumlah Neuron			Akurasi (%)	Presisi (%)	Recall (%)	Time (s)
	Hidden Layer 1	Hidden Layer 2	Hidden Layer 3				
tahap 7	96	96	64	89.77	89.77	89.78	236.715
tahap 8	96	64	64	89.04	89.05	89.05	221.264
tahap 9	96	64	32	86.53	86.54	86.53	194.34
tahap 10	64	64	64	91.37	91.38	91.38	246.968
tahap 11	96	96	96	91.42	91.43	91.43	250.042
tahap 12	128	128	128	**91.72**	**91.73**	**91.73**	**382.725**

Pada Tabel 4.5 hasil dari kombinasi tiap *neuron* dengan ukuran berbeda menghasilkan akurasi yang berbeda. Untuk *neuron* berukuran besar memiliki akurasi yang lebih baik dibandingkan dengan *neuron* dengan ukuran yang lebih kecil, namun waktu komputasi yang dibutuhkan *neuron* ukuran besar lebih lama dibandingkan dengan jumlah *neuron* yang lebih sedikit. Akurasi *testing* model dengan *neuron hidden layer* berjumlah 128 lebih besar daripada akurasi model dengan *neuron hidden layer* berjumlah 64 dan 96. Model dengan *neuron hidden layer* berjumlah 128 memilki akurasi sebesar 91.72% sedangkan model dengan *neuron hidden layer* berjumlah 64 dan 96 memilki akurasi 91.37% dan 91.42%. Jumlah *neuron* 128 dan 96 memiliki perbandingan akurasi 0.30% dan perbandingan waktu komputasi 50.883s. Namun melihat dari sisi akurasi, maka jumlah *neuron* 128 yang terpilih sebagai model terbaik dengan waktu komputasi 340.925s. Dengan demikian, pengujian untuk parameter selanjutnya, menggunakan arsitektur dengan *neuron hidden layer* berjumlah 128.

4.11. Uji Pengaruh *Learning Rate*

Learning rate merupakan salah satu parameter *training* untuk menghitung nilai koreksi bobot pada waktu proses training. Dengan nilai α ini berada pada *range* nol sampai satu. Pada penelitian ini nilai dari *learning rate* yang diujikan yaitu 0.001, 0.002, 0.003, 0.004 dan 0.005. Pada pengujian ini arsitetur yang digunakan adalah terbaik dari pengujian sebelumnya. Yaitu menggunakan 2 HL dengan jumlah *neuron* 128 dengan ukuran citra 128x128, koefisien DCT 64 dan *eigen value* 17. Pengujian ini bertujuan untuk mengetahui performa dari *learning rate* dengan nilai berbeda dan akurasi terbaik nantinya digunakan sebagai paramter pengujian pada skenario selanjutnya. Tabel 4.6 menunjukkan hasil dari performa pengujian *learning rate*.

Tabel 4.6 Pengujian *learning rate*

Nilai Learning Rate	Akurasi Testing (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
0.001	91.72	91.73	91.73	258.964
0.002	91.82	91.83	91.83	249.725
0.003	92.11	92.12	92.13	240.670
0.004	91.59	91.60	91.60	239.446
0.005	91.66	91.67	91.67	237.350

Berdasrkan Tabel 4.6, *learning rate* dengan nilai 0.003 memiliki akurasi tertinggi yaitu 92.11% dengan waktu komputasi 240.670s. *Learning rate* lainnya yaitu 0.001 menghasilkan akurasi 91.72% dengan waktu komputasi 258.964s. *Learning rate* 0.002 menghasilkan akurasi 91.82% dengan waktu komputasi 249.725s. *Learning rate* 0.004 menghasilkan akurasi 91.59% dengan waktu komputasi 239.446s. Dan yang terakhir yaitu *learning rate* 0.005 menghasilkan akurasi 91.66% dengan waktu komputasi 237.350s. Berdasrkan hasil pengujian diatas dapat disumpulkan bahwa semakin besar nilai *leraning rate* yang digunakan maka semakin cepat waktu yang dibutuhkan pada proses *training*.

4.12. Uji Pengaruh Ukuran Citra

Ukuran citra merupakan bagian yang diatur di dalam persiapan sebelum melakukan proses *training* atau bisa disebut sebagai bagian yang ditentukan pada tahap *pre-processing*. Semakin kecil ukuran citra maka detail dari citra itu sendiri semakin tidak terlihat sedangkan semakin besar ukuran citra semakin banyak informasi fitur yang didapatkan. Pada penelitian ini *size* citra akan uji pada 3 ukuran citra yaitu 32x32, 64x64 dan 128x128 dengan menggunakan arsitektur *neuron* 128 2 *hidden layer*, *learning rate 0.003*. Hasil dari pengujian dapat dilihat pada Tabel 4.7.

Tabel 4.7 Pengujian *ukuran citra*

Ukuran Citra	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
32X32	85.70	85.72	85.72	254.880
64x64	91.32	91.33	91.33	251.864
128X128	92.11	92.12	92.13	240.670

Berdasarkan hasil pengujian Tabel 4.7, ukuran citra 128x128 memiliki performa

tertinggi dengan akurasi 92.11% dengan waktu komputasi 240.670 s, sementara citra dengan ukuran 32x32 dan 64x64 memiliki akurasi sebesar 85.70% dengan waktu komputasi 254.880 s dan 91.32% dengan waktu komputasi 251.864 s . Sehingga arsitektur terbaik pada pengujian terhadap ukuran citra masukan adalah citra 128x128 . Untuk waktu komputasi ukuran citra 128x128 memiliki waktu komputasi tercepat dibandingkan dengan ukuran citra 64x64 dan 32x32 . Hal ini dikarenakan pengaruh koefisien DCT pada citra ukuran 128x128 lebih kecil dibandingkan ukuran citra lainnya. Koefisien DCT dengan akurasi tertingi pada ukuran citra 128x128 yaitu 64, sementara untuk ukuran citra 64x64 dan 32x32 akurasi koefisien DCT tertinggi yaitu ada pada koefisien 81. Sehingga pemilihan hal ini dapat mempengaruhi waktu komputasi pada saat proses *training*.

4.13. Pengujian Model

4.13.1 Pengujian Model *Dataset* 10800 Citra

Berdasarkan pengujian parameter terbaik pada Tabel 4.2, Tabel 4.3, Tabel 4.4, Tabel 4.5, Tabel 4.6 dan Tabel 4.7 didapatkan hasil pengujian model pada Tabel 4.8 dengan menggunakan *cross validation* 10 *fold* untuk 10800 data. Pembagian data dalam tiap *fold* 1800 data per *fold* baik untuk tiap data *training* dan data *testing*. Pengujian ini menggunakan model terbaik hasil pengujian sebelumnya, yaitu *size* 128x128 dengan koefisien DCT 64, *leraning rate* 0.003, 2 *hidden layer* dengan jumlah *neuron* 128. Model terbaik menghasilkan rata-rata akurasi yaitu 92.20% dengan rata-rata waktu komputasi adalah 250.691s.

Tabel 4.8 Performa pengujian *dataset* 10800 dengan cross validation

cros validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	92.37	92.38	92.38	249.676
2	100	91.60	91.61	91.61	250.148
3	100	92.43	92.44	92.44	244.177
4	100	92.25	92.26	92.26	256.924
5	100	91.35	91.36	91.36	256.266
6	100	92.71	92.72	92.72	260.197
7	100	92.38	92.39	92.39	243.798
8	100	92.10	92.11	92.11	251.292
9	100	92.63	92.64	92.64	242.354

10	100	92.01	92.02	92.02	252.115
Rata-rata	100	92.20	92.21	92.21	250.691

4.13.2 Pengujian Model *Dataset* 2700 Citra

Sama seperti pengujian dataset 10800, model terbaik juga diujikan pada *dataset* 2700 citra Aksara Sasak [3] dengan menggunakan *cross validation* 10 *fold*. Pembagian data dalam tiap *fold* 270 data per *fold* baik untuk tiap data *training* dan data *testing*. Hasil pengujian pada Tabel 4.9 menunjukkan bahwa model terbaik mengahsilkan rata-rata akurasi yaitu 71.73% dengan rata-rata waktu komputasi adalah 65.446s.

Tabel 4.9 Performa pengujian *dataset* 2700 dengan *cross validation*

cros validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	99.84	73.82	73.81	73.82	85.908
2	99.78	71.11	71.12	71.11	62.919
3	99.89	75.16	75.17	75.16	62.097
4	99.94	74.93	74.94	74.93	64.133
5	99.52	72.71	72.72	72.71	65.575
6	99.57	72.22	72.23	72.22	78.552
7	99.78	73.45	73.46	73.45	62.235
8	99.89	75.18	75.19	75.18	57.833
9	99.89	72.71	72.72	72.71	57.314
10	99.84	74.07	74.08	74.07	57.890
Rata-rata	99.79	71.73	71.74	71.74	65.446

4.13.3 Pengujian Model *Dataset* 13500 Citra

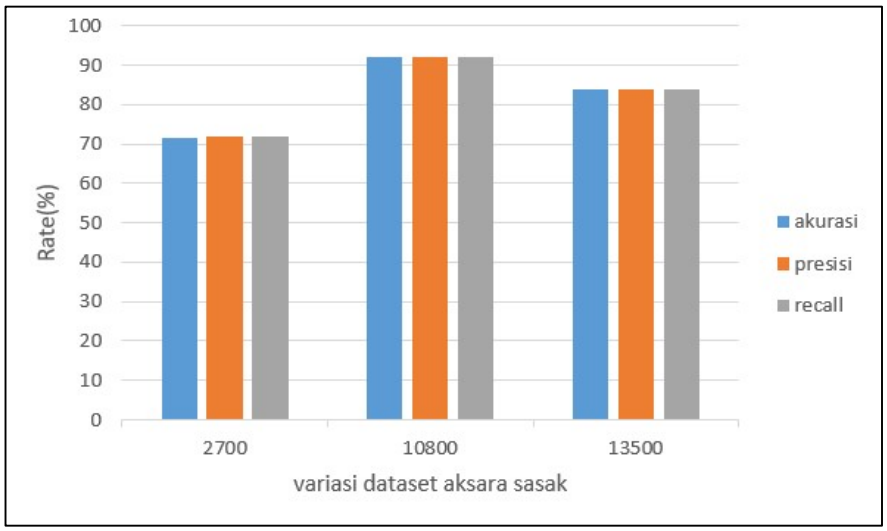
Pengujian terakhir dilakukan pada *dataset* gabungan dengan tujuan untuk mengetahui ketangguhan model terbaik terhadap variasi data. Pengujian ini menggunakan *cross validation* 10 *fold*. Pembagian data dalam tiap *fold* 1350 data per *fold* baik untuk tiap data *training* dan data *testing*. Hasil pengujian seperti pada Tabel 4.10 menunjukkan bahwa model terbaik yang digunakan untuk pengujian *dataset* 13500, menghasilkan rata-rata akurasi yaitu 83.92% dengan rata-rata waktu komputasi adalah 361.275s. Hal ini menunjukkan bahwa penambahan data aksara sasak pada penelitian [3] mengakibatkan turunnya akurasi pengujian citra aksara gabungan berdasarkan model terbaik yang telah diujikan sebelumnya. Perbedaan pengambilan data dan *preprocessing* citra pada penelitian [3], sangat berpengaruh

terhadap hasil pengujian pada penelitian ini.

Tabel 4.10 Performa pengujian *dataset* 13500 dengan *cross validation*

cros validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	99.88	85.75	85.76	85.75	348.126
2	99.57	83.80	83.81	83.80	360.224
3	98.22	83.38	83.38	83.37	369.242
4	99.43	83.43	83.44	83.43	365.426
5	99.75	84.14	84.15	84.14	337.357
6	99.88	84.49	84.50	84.50	363.858
7	99.91	84.69	84.70	84.69	367.107
8	99.31	83.23	83.24	83.24	367.031
9	99.95	83.72	83.73	83.73	364.443
10	99.88	82.46	82.47	82.46	370.213
Rata-rata	99.58	83.92	83.93	83.92	361.275

Berdasarkan pengujian yang telah dilakukan, dapat dibandingkan akurasi, presisi, dan *recall*-nya untuk 3 variasi *dataset* Aksara Sasak, seperti disajikan pada Gambar 4.6.



Gambar 4.6 Diagram perbandingan nilai akurasi, presisi, dan *recall* pengujian dengan keseluruhan variasi *dataset*

Gambar 4.6 menunjukkan bahwa performa terbaik ditunjukkan ketika melakukan pengujian menggunakan *dataset* Aksara Sasak sejumlah 10800 citra. Hal ini dikarenakan adanya perbedaan proses pengambilan data dan *preprocessing* data yang dilakukan pada *dataset* 10800 dan *dataset* 2700. Pengambilan data dan

preprocessing data sangat mempengaruhi akurasi dari suatu citra. Berdasarkan kedua jenis *dataset*, terlihat perbedaan ketebalan citra, citra pada *dataset* 10800 ditulis lebih tebal dibandingkan dengan citra pada *dataset* 2700. Proses *scanning* yang dilakukan juga berbeda . Hal ini dapat mempengaruhi fitur-fitur yang ada pada suatu citra. Banyaknya variasi penulisan aksara pada *dataset* 10800 berdasarkan empat kategori pendidikan menunjukkan hasil yang lebih baik. Sementara untuk data 2700 memiliki akurasi yang paling rendah di antara ketiga data, hal ini dikarenakan jumlah data yang lebih sedikit dan variasi tulisan aksara yang lebih sedikit pula sehingga proses pengenalan citra tidak maksimal. Dan untuk data gabungan antara 13500 memiliki akurasi dibawah data 10800 dan memiliki akurasi yang lebih tinggi daripada *dataset* 2700. Hal ini dikarenakan data perbedaan dalam mengolah *dataset* pada *dataset* 2700. Akurasi *dataset* 10800 lebih tinggi dibandingkan *dataset* 13500 dikarenakan pengaruh penggabungan *dataset* 2700 yang memiliki perlakuan data yang berbeda dengan *dataset* 10800. Jadi dapat disimpulkan bahwa proses perlakuan terhadap data seperti pengambilan data dan *preporocessing* data sangat mempengaruhi akurasi dari suatu citra. Banyaknya data dan variasi sumber penulisan beragam mengahsilkan akurasi yang lebih baik dibandingkan dengan jumlah data yang sedikit. Semkain banyak ragam tulisan atau jenis aksara yang ada maka JST-BP dapat belajar lebih banyak sehingga menghasilkan akurasi yang lebih tinggi.

BAB V
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang sudah dilakukan, terdapat beberapa hal yang bisa penulis simpulkan antara lain sebagai berikut.

1. Model terbaik yang dihasilkan pada penelitian ini menggunakan *input* citra *grayscale* dengan ukuran 128x128 pixel, dengan koefisien DCT 64, *eigen value* bernilai 17, *hidden layer* berjumlah 2 dengan *neuron* masing-masing sebanyak 128 dan *learning rate* 0.003.
2. Persentasi tertinggi pada pengujian ini yaitu 92.20% pada *dataset* 10800 berdasarkan arsitektur dan parameter uji terbaik dari pengujian sebelumnya. Hal ini membuktikan bahwa metode usulan memiliki potensi yang baik untuk dikembangkan sebagai media pembelajaran Aksara Sasak.
3. *Learning rate* terbaik pada pengujian ini yaitu 0.003, dimana semakin besar nilai *learning rate* maka dapat mempercepat proses komputasi.
4. *Eigen value* terbaik yaitu bernilai $n-1$ dari kelas yaitu 17 eigen, hal ini dikarenakan citra dibagi menjadi 18 kelas sesuai dengan jumlah karakter Aksara Sasak.
5. Akurasi tertinggi dari perbandingan ketiga jenis *dataset* ditunjukkan pada *dataset* 10800, hal ini disebabkan oleh adanya perbedaan proses pengambilan data dan preprocessing dengan *dataset* 2700. Hal ini menunjukkan pengambilan data dan persiapan data sebelum masuk ke tahap selanjutnya sangat mempengaruhi akurasi.

5.2. Saran

Ada beberapa saran yang dapat penulis berikan apabila penelitian ini akan dikembangkan kembali antara lain sebagai berikut.

1. Persiapan sebelum penelitian seperti pengambilan data harus diperhatikan untuk mendapatkan data penelitian yang bagus dan kategori sumber pengambilan data dapat ditambahkan seperti penutur langsung Aksara Sasak.
2. Pembuatan media pembelajaran aksara sasak dalam platform *andorid* atau *web*.

3. Dapat mencoba model arsitektur lain pada pengujian *neural network* seperti menambahkan *hidden layer* dan jumlah *neuron* yang lebih beragam.
4. Menguji parameter terbaik pada tiap *dataset* yaitu *dataset* 10800,2700 dan 13500.

DAFTAR PUSTAKA

[1] F. H. Tondo, "Kepunahan Bahasa-Bahasa Daerah: Faktor Penyebab Dan Implikasi Etnolinguistis," *J. Masy. Budaya*, vol. 11, no. 2, pp. 277–296, 2009.

[2] Governor, Bali Governor Regulation number 80 of 2018 About Protection and Use of Bali, Aksara, And Literature as well as the Implementation of the Bali language. 2018, pp. 1–9.

[3] Brian D. Ripley, "*Pattern Recognition and Neural Networks*", Cambridge: University Press, 1996.

[4] Riska Yulianti, "Pengenalan Pola Tulisan Tangan Suku Kata Aksara Sasak Menggunakan Metode *Moment Invariant* dan *Support Vector Machine*,". Mataram: Universitas Mataram, 2018.

[5] Eka Dina Juliani Utari, "Pengenalan Pola Tulisan Tangan Huruf Sasak Menggunakan Metode *Integral Projection* dan *Neural Network,*" *J-COSINE*, Vol. 3, No. 1, 2019.

[6] F. Fandiansyah, J. Y. Sari, and I. P. Ningrum, "Pengenalan Wajah Menggunakan Metode Linear Discriminant Analysis dan k Nearest Neighbor," *J. Ultim.*, vol. 9, no. 1, pp. 1–9, 2017.

[7] A. Rachmad, "Ekstraksi Fitur Menggunakan Metode Lda dan Pemilihan Eigen Value Pada Cacat Kertasduplek," *J. SimanteC*, Vol. 3, pp.142-149, 2013.

[8] R. Lim, Raymond dan K. Gunadi , "Face Recognition Menggunakan Metode Linear Discriminant Analysis (LDA)," *J. KOMMIT*, Vol.12, pp. 134-142, 2002.

[9] B. Pradinta, Ernawati dan E. P. Purwandari, "Identifikasi Citra Garis Telapak Tangan Menggunakan Metode Linear Discriminant Analysis Dengan Probabilitas Naïve Bayesian, " *J. Pseudocode*, Vol 4, No.2, pp.156-167, 2017

[10] B. Widoyono, T. Agung Budi Wirayuda dan B. Purnama, "Implementasi Linear Discriminant Analysisi Dan Jaringan Syaraf Tiruan Backprogation Untuk Membaca Angkat Pada Meteran Air Secara Otomatis," Telkom University, 2013.

[11] Farida Asriani, "*Handwritter Javanesse Character Recognation System*

Using Artificial Network Backpropagation," vol. 5, 2009.

[12] Nazla Nurmila, "Algoritma *Back Propagation Neural Network* Untuk Pengenalan Pola Karakter Huruf Jawa," vol. 1, no. 1, 2015.

[13] B. Isnawati, "Analisis Implementasi Jaringan Syaraf Tiruan *Back Propagation* Untuk Klasifikasi Huruf Dasar Aksara Jawa" Yogya: Universitas Gadjah Mada, 2015.

[14] D. Wulansari, "Identifikasi *Gender* Berdasarkan Citra Wajah Menggunakan Deteksi Tepi dan *Backpropagation*," J. *SNATi*, 2017.

[15] D. A. Pancorowati dan M. A. Bustomi, "Klasifikasi Pola Huruf Vokal dengan Menggunakan Jaringan Saraf Tiruan," *J.* TEKNIK POMITS, pp. 1-7, 2017.

[16] R. Abdilah, "Identifikasi Otentifikasi Citra Tanda Tangan Menggunakan *Wavelet* dan *Backpropagation*," *J. Seminar Nasional Aplikasi Teknologi sInformasi*, pp. 5-9, 2017.

[17] Sulistiyasni, "Klasifikasi Pola Sidik Jari Menggunakan Jaringan Syaraf Tiruan *Backpropagation*," *J. Ilmiah Teknik Informatika*, vol.10, pp. 215-224, 2016.

[18] F. Asriani, "Pengenalan Pola Aksara Jawa Tulisan Tangan dengan Jaringan Syaraf Tiruan Perambatan-Balik," *J. Ilmiah Dinamika Rekayasa*, vol. 5, no. 2, pp. 34-36, 2009.

[19] Fandiansyah, J. Y. Sari, and I. P. Ningrum, "Pengenalan Wajah Menggunakan Metode Linear Discriminant Analysis dan k Nearest Neighbor," *J. Ultim.*, vol. 9, no. 1, pp. 1–9, 2018.

[20] M. C. Wijaya and A. Prijono, Pengolahan Citra Digital Menggunakan MatLAB Image Processing Toolbox, Bandung: Informatika, 2007.

[21] I. M. Mataram, "Peramalan Beban Hari Libur Menggunakan *Artificial Neural Network*," *J. Tek. Elektro*, pp. 53-56, 2008.

[22] Bahrie., H. Sudirman & Lalu Ratmaja, "Bahan Ajar Muatan Lokal, Gumi Sasak, KSU 'Prima Guna' ". Lombok Timur, 2013.

[23] H. Yasri, "Cara Cepat Belajar Aksara Sasak". Mataram: Pustaka Widiya, 2010

[24] Teodoridis, S dan Koutrombas, K., "*Pattern Recognition 3th Edition*". London: *Academic Press*, 2006.

[25] Aloysius Tanto Wibowo, "Pengenalan Pola Tulisan Tangan Aksara Jawa

Dengan Algoritma *Backpropagation*". Yogyakarta, 2018.

[26] Shabrina, M., "Pengenalan Iris Mata Menggunakan Metode Analisis Komponen Utama (*Principal Component Analysis* – PCA) dan Jaringan Saraf Tiruan Perambatan Balik". Skripsi. Universitas Diponegoro, 2012.

[27] Maimon, O., Rokach, L. "*Data Mining and Knowledge Discovery Handbook Second Edition*". New York : Springer, 2010.

[28] Ramdhani, Y., "Komparasi Algoritma LDA Dan Naïve Bayes Dengan Optimasi Fitur Untuk Klasifikasi Citra Tunggal Pap Smear". Informatika, vol. 2, no. 2, pp. 434 – 441, 2005.

[29] G. K. Lim Resmana, Raymond, "*Face Recognition* Menggunakan Metode *Linear Discriminant Analysis* (LDA)," *Kommit*, p. 9, 2002.

[30] S. Cahyani, R. Wiryasaputra, and R. Gustriansyah, "Identifikasi Huruf Kapital Tulisan Tangan Menggunakan *Linear Discriminant Analysis* dan *Euclidean Distance*," vol. 01, pp. 57–67, 2018.

[31] A. Jumarwanto, R. Hartanto, and D. Prastiyanto, "Aplikasi Jaringan Saraf Tiruan *Backpropagation* Untuk Memprediksi Penyakit THT Di Rumah Sakit Mardi Rahayu Kudus," *J. Tek. Elektro*, vol. 1, no. 1, pp. 11–21, 2009.

[32] J. J. Siang, *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab*, 1st ed. Yogyakarta: ANDI, 2005.

[33] L. Fausett, *Fundamentals of neural networks: architectures, algorithms, and applications*. Melbourne: Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1994.

[34] W. Chen, J. Er-Meng and S. Wu, PCA and LDA in DCT domain, Pattern Recognition Letter, vo1.26, pp.2474-2482, 2005.

[35] I. G. P. S. Wijaya, K. Uchimura, and Z. Hu, "Face Recognition Based on Dominant Frequency Features and Multiresolution Metric," *Int. J. Immovative Comput. Inf. Control*, vol. 5, no. 1349–4198, pp. 641–651, 2009.

[36] R. V. Nahari, A. S. Editya, and R. Alfita, "Ekstraksi Fitur Daun Tembakau Berbasis Discrete Cosine Transform (DCT)," *J. Appl. Informatics Comput.*, vol. 4, no. 1, pp. 8–12, 2020.

[37] R. Krasmala, A. B. Purba, and U. T. Lenggana, "Kompresi Citra Dengan Menggabungkan Metode Discrete Cosine Transform (DCT) dan Algoritma

Huffman," *JOIN*, vol. 2, no. 1, pp. 1–9, 2017.

[38] A. N. Fadhlillah, "Analisis Dan Implementasi Klasifikasi K-Nearest Neighbor Telapak Kaki Manusia K-Nearest Neighbor (K-Nn) on System Identification of," *Telkom Univ.*, vol. 2, no. 2, pp. 2876–2883, 2015.

[39] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–15, 2015.

[40] S. Sholahuddin, A. Siregar, Rustam, E. Supriana, I. Hadi, "Penerapan Metode Liniear Discriminant Analysis Pada Pengenalan Wajah Berbasis Kamera," *Konf. Nas. Mat.*, vol. 15, pp. 1–9, 2010.