USULAN TUGAS AKHIR
PENGENALAN POLA TULISAN TANGAN AKSARA SASAK MENGGUNAKAN EKSTRAKSI FITUR PCA DENGAN METODE ANN

Oleh :

Frizqa Ervina

F1D016029

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MATARAM

2020

Usulan Tugas Akhir

PENGENALAN POLA TULISAN TANGAN AKSARA SASAK MENGGUNAKAN EKSTRAKSI FITUR PCA DENGAN METODE ANN

Oleh:

FRIZQA ERVINA

F1D 016 029

Telah diperiksa dan disetujui oleh Tim Pembimbing:

1. Pembimbing Utama

Prof. I GP Suta Wijaya, S.T.,M.T.,D.Eng. Tanggal: 13 April 2020

NIP. 19731130 200003 1 001

2. Pembimbing Pendamping

Fitri Bimantoro,S.T.,M.Kom Tanggal: 13 April 2020

NIP. 19860622 201504 1 002

Mengetahui

Ketua Program Studi Teknik Informatika

Fakultas Teknik

Universitas Mataram

Prof. I GP Suta Wijaya, S.T.,M.T.,D.Eng.

NIP. 19731130 200003 1 001

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Aksara Sasak adalah salah satu dari ragam tulisan aksara di Indonesia yang digunakan sebagai tulisan tradisional khususnya di Lombok untuk menuliskan bahasa Sasak. Pengenalan pola aksara Sasak merupakan salah satu penelitan yang pernah dilakukan sebelumnya yang dimana, hasil akurasi yang didapatkan bervariasi sesuai dengan metode penelitian yang digunakan. Penelitian ini bertujuan untuk mengetahui akurasi yang dihasilkan dari pengenalan pola tulisan tangan aksara Sasak menggunakan gabungan metode ekstraksi fitur *Prncipal Component Analysis* (PCA) dengan metode klasifikasi *Artificial Neural Network* (ANN) yaitu *backpropagataion*, dengan memanfaatkan kelebihan *Prncipal Component Analysis* (PCA) dalam mereduksi data dan klasfikasi *backpropagataion* yang memiliki perhitungan yang lebih baik dalam mencapai nilai target yang diinginkan sehingga penelitian ini dapat membantu peneliti-peneliti selanjutnya untuk mengembangkan sebuah aplikasi tentang pengenalan aksara Sasak yang nantinya bisa digunakan masyarakat sebagai media belajar aksara Sasak. Data yang digunakan untuk penelitian ini merupakan data dari sumber dengan tulisan tangan di kertas HVS (*Hout Vrij Schrift*) ukuran A4 menggunakan spidol dengan kategori SD, SMP, SMA dan Universitas yang pernah belajar aksara Sasak dan yang belum pernah mempelajari aksara Sasak.

Kata kunci : pengenalan pola, tulisan tangan, aksara, *principal component analysis* , *backpropagation*.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Aksara Sasak merupakan salah satu dari ragam tulisan aksara di Indonesia yang digunakan sebagai tulisan tradisional khususnya di Lombok untuk menuliskan Bahasa Sasak. Aksara Sasak terdiri dari 18 suku kata yang diadaptasi dari aksara Bali dan Jawa [1]. Masyarakat Lombok sudah mengenal aksara Sasak sejak jaman dahulu sebagai media komunikasi masyarakat Sasak melalui tulisan naskah tradisional menggunakan kertas. Biasanya aksara Sasak ditemukan di peninggalan kuno seperti naskah-naskah, ketika menceritakan sejarah Lombok atau perkembangan agama Islam [2]. Penggunaan aksara Sasak di Lombok saat ini digunakan untuk menulis nama-nama jalan, tetapi beberapa orang masih belum dapat membaca aksara Sasak dengan baik, ini diakibatkan karena pengaruh bahasa Indonesia sebagai bahasa nasional terutama dalam berbagai ranah resmi (formal) seperti pemerintahan dan pendidikan, yang seringkali menyebabkan frekuensi pemakaian bahasa daerah semakin berkurang dan membuat masyarakat tidak memahami bahasa daerah mereka sendiri serta kurangnya pendidikan aksara Sasak di sekolah-sekolah di Lombok [3].

Penelitian ini penting dilakukan untuk membantu orang untuk mengenali dan mengetahui aksara Sasak dengan baik. Sebelumnya, ada beberapa penelitian tentang pengenalan pola tulisan tangan contohnya yaitu penelitian pengenalan pola huruf aksara Sasak menggunakan metode *integral projection* dan *neural network* yang menghasilkan akurasi tertinggi menggunakan 2 *hidden layer* dengan nilai akurasi sebesar 41,38%. Hasil yang didapat masih kurang dari 50%, hal tersebut dipengaruhi oleh jumlah *hidden layer* dan *node* dalam jaringan [4]. Jumlah *node* dan *hidden layer* yang semakin banyak dalam jaringan tidak menjamin bahwa tingkat akurasi akan mengalami peningkatan karena pada *backpropagation* perlu dilakukan *trial and error* untuk kombinasi jumlah *hidden layer* dan *node* agar mendapat arsitektur terbaik.

Penelitian selanjutnya yaitu pengenalan pola tulisan tangan suku kata aksara Sasak menggunakan metode *moment invariant* dan *support vector mechine* yang menghasilkan nilai akurasi yang cukup tinggi yaitu 89,76% pada skenario pengujian pertama dengan proses pelatihan menggunakan 1800 data latih dan 900 data uji, sedangkan skenario kedua menghasilkan nilai akurasi 92,52% dengan menambahkan K-*Fold Cross Validation* pada proses pengujiannya [1].

Pada penelitian pengenalan karakter bahasa Urdu menggunakan *Principal Component*

Analysis (PCA) menghasilkan akurasi yang baik dengan rata-rata akurasi di atas 90%. PCA digunakan sebagai reduksi dimensi dan ekstraksi fitur data. Reduksi dimensi mampu menambah kecepatan komputasi tanpa kehilangan informasi penting saat data diproses [5], hal ini bagus digunakan dalam menangani data yang cukup banyak dan untuk klasifikasi berdasarkan penelitian pengenalan tulisan tangan secara *real time* menggunakan *backpropagation* [6] memiliki hasil yang baik dalam pengenalan pola karena *backpropagation* mempunyai arsitektur pemrosesan paralel sehingga menghasilkan efisiensi yang optimal. Selain itu, ada beberapa jenis jaringan seperti *perceptron*, *feed forward*, *feedback network* yang menyajikan cara variabel untuk mengaitkan *input* dengan *output* agar mencapai nilai output yang diinginkan.

Berdasarkan penelitian sebelumnya, maka digabungkan ekstraksi fitur *principal component analysis* (PCA) untuk proses ekstraksi fitur aksara Sasak untuk membantu menambah kecepatan serta mereduksi dimensi data tanpa mengurangi karakteristik data yang diuji dan menggunakan metode ANN yaitu *backpropagation* sebagai algoritma untuk memperkecil nilai *error* dengan cara penyesuaian bobot agar dapat mencapai nilai *output* yang diinginkan.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang sudah dijelaskan sebelumnya, dapat diambil rumusan masalah yaitu bagaimana performa dan presentase keberhasilan dari ekstrasi fitur PCA yang digabungkan dengan metode klasifikasi ANN untuk pengenalan pola huruf aksara Sasak.

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. Penelitian ini hanya menghasilkan nilai akurasi dalam bentuk nilai persentase.
2. Pengambilan data berupa tulisan tangan seseorang menggunakan kertas HVS A4 dengan tabel yang terdiri dari 3 baris dan 6 kolom yang dimana masing-masing kotak memiliki panjang dan lebar 4 cm membentuk persegi sama sisi.
3. Data tulisan di-*scan* menggunakan *scanner* agar menghasilkan soft file dalam bentuk format .jpg
4. Data yang digunakan dalam penelitian merupakan data asli tulisan tangan seseorang yang

diambil secara langsung sebanyak 10.800 data dengan kategori tulisan orang yang belajar aksara Sasak dan orang yang tidak belajar aksara Sasak yang terdiri dari tulisan tangan SD, DMP, SMA, kuliah dan ditambah dengan data peneliti sebelumnya sebanyak 2700 data sehingga total data yang digunakan sebanyak 13.500 data.

1.4 Tujuan

Tujuan yang diharapkan dari penelitian ini adalah untuk mengetahui performa dan persentase keberhasilan. dari ekstraksi fitur PCA dan klasifikasi metode ANN dalam 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 dibidang *machine learning* dan cara mengolah citra.
2. Bagi pembaca
   a. Dapat mengetahui bagaimana cara untuk mengolah citra agar dapat diproses ke dalam *machine learning*.
   b. Dapat dijadikan bahan ajar tentang gabungan PCA dan ANN.
   c. Hasil klasifikasi dapat menjadi rujukan untuk mengembangkan sistem pembelajaran huruf aksara yang lebih kompleks.

1.6 Sistematika Penulisan

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

1. Bab I Pendahuluan

   Bab ini menjelaskan tentang latar belakang, rumusan masalah, tujuan, batasan masalah, dan manfaat penelitian yang akan dilakukan. Latar belakang penelitian ini menjelaskan masalah yang melatar belakangi pembuatan sistem pengenalan pola huruf aksara Sasak berdasarkan penelitian-penelitian sebelumnya.
2. Bab II Tinjauan Pustaka dan Dasar Teori

   Bab ini membahas tentang penelitian-penelitian sebelumnya yang akan dijadikan

referensi dan pembelajaran untuk melakukan penelitian pengenalan pola aksara Sasak. Dasar teori yang terkait dengan penelitian yang akan dilakukan yaitu konsep dan algoritma cara perhitungan menggunakan PCA dan ANN.

3. Bab III Metodologi Penelitian

   Bab ini membahas tentang alat, bahan, metodologi, konsep, dan contoh perhitungan PCA dengan ANN yang akan digunakan untuk mencari nilai akurasi dari penelitian pengenalan pola huruf aksara Sasak.

4. Bab IV Hasil dan Perancangan

   Pada bab ini berisi penjelasan dan analisis dari hasil penelitian yang dilakukan. Analisis berupa analisis masalah, analisis kebutuhan sistem, analisis metode, serta perancangan sistem yang terdiri dan *flowchart*.

5. Bab V Kesimpulan dan Saran

   Bab ini berisi kesimpulan dan saran berdasarkan hasil penelitian yang telah dilakukan.

BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1 Tinjauan Pustaka

Penelitian mengenai pengenalan pola suatu citra sudah banyak dilakukan oleh para peneliti menggunakan metode ANN dan PCA dalam beberapa waktu terakhir. Penelitian-penelitian tersebut akan menjadi acuan untuk melaksanakan penelitian ini.

Penelitian tentang pengenalan tulisan tangan sebelumnya telah dilakukan beberapa kali. Penelitian yang dimaksud yaitu sistem pengenalan pola motif batik kediri [7]. Beberapa penelitian yang menggunakan metode ANN untuk pengenalan pola huruf hijaiyah khat kufi [8], pengenalan karakter *alphabet* dan arab [9], pengenalan sandi rumput pramuka [10], pengenalan tulisan aksara lampung[11] dan pengenalan notasi balok[12]. Sedangkan, penelitian yang menggunakan *principal component analysis* (PCA) yaitu penelitian pengenalan pola senyun [13], citra tanda tangan [14], pengenalan angka tulisan tangan [15] , dan pengenalan wajah [16].

Dari referensi yang diperoleh tersebut, semua penelitian berhasil melakukan pengenalan atau klasifikasi dengan baik. Akurasi dari tiap penelitian dapat sajikan dalam Tabel 1.1 Referensi penelitian sebelumnya.

Tabel 1.1 Referensi penelitian sebelumnya

NO	Penulis	Judul	Keterangan	Akurasi Pengujian
1	Anggi Wibowo (2017)	Sistem Pengenalan Pola Motif Batik Kediri	- Menggunakan ekstraksi fitur PCA dan klasifikasi *euclidean distance* - Preprocessing yang digunakan adalah RGB, dengan membagi citra ke dalam bentuk 3 warna berbeda yaitu citra merah, biru dan hijau. - Pengujian dilakukan menggunakan 10 jenis	80%

			batik berbeda - Semakin kecil nilai euclidean distance, maka data *testing* dan data *training* semakin mirip. - PCA digunakan untuk mereduksi dimensi citra untuk mempermudah komputasi	
2	Irvan Faturrahman, Arini dan Fitri Mintarsih (2018)	Pengenalan Pola Huruf Hijaiyah Khat Kufi Dengan Metode Deteksi Tepi Sobel Berbasis Jaringan Syaraf Tiruan Backpropagation	- Menggunakan parameter uji *learning rate* dan *epoch* yaitu *learning* rate 0.01, 0.05, 0.1, 0.5, dan *epoch* 1000, 3000, 5000, 10000. - Simulasi dilakukan pada 28 huruf hijaiyah. - Menggunakan 8 skenario pengujian - Data simulasi berukuran 300x300 *pixel* yang di-*resize* menjadi 30x30 *pixel* - *Penggunaan learning rate* yang semakin besar membuat algoritma tidak stabil dan akurasi rendah. - Semakin besar nilai	*learning rate* 0.01 dan *epoch* 10000 memiliki akurasi 100%

			epoch maka proses pelatihan semakin lama tetapi akurasi cukup tinggi	
3	Aro Taye Oladele, Musa Abdullahi Yola, Abdulkadir Ikeola Suhurat, Adeoye Latifat Bukola (2018)	*Recognition of Alphabet Characters and Arabic Numerals Using Backpropagation Neural Network*	- Menggunakan diagonal *feature extraction* dan klasifikasi *backpropagation* - 63 *dataset* untuk huruf arab - Menggunakan *epoch* sebanyak 5000 dan *learning rate* 0,01. - Menggunakan 2 *hidden layer* dan fungsi aktivasi sigmoid - 1650 karakter alphabet untuk *training* data dan 30 alphabet berbeda untuk tes sistem	91,66% untuk angka arab dan 92% untuk karakter alphabet
4	Syamsudin Zubair, Achmad Solichin (2017)	Pengenalan Karakter Sandi Rumput Pramuka Menggunakan Jaringan Saraf Tiruan dengan Metode *Backpropagation*	- 252 pola karakter yang akan diuji - 7 kelas pola yang dimana 5 merupakan tulisan tangan, dan 2 merupakan hasil olahan *tools* komputer. - Setiap 1 kelas pola karakter terdiri dari	76.28% untuk karakter tunggal , 78.37% untuk kata yang di sambung

			36 karakter sehingga 252 pola karakter yang akan diuji. - kegagalan dalam pengenalan karakternya yaitu disebabkan oleh buruknya kualitas citra dan konsistensi pola - sebaiknya menggunakan media yang memiliki tingkat pixel yang bagus	
5	Eliza Hara, Helmy Fitriawan, Yessi Mulyani (2016)	Penggunaan Deteksi Tepi (Canny) pada Sistem Pengenalan Tulisan Tangan Aksara Lampung Berbasis Jaringan Syaraf Tiruan	- 15 set data sampel dari 75 set yang diambil secara acak. - Pengambilan data menggunakan perangkat digital - Pelatihan dilakukan sebanyak 5 kali pelatihan dengan jumlah data masukan sebanyak 15 set (195 sub-set) dan neuron masukan 700 neuron - Hasil resize diubah ke dalam matriks vector 700x1 - aksara Lampung beberapa memiliki	78% dari 10 (sepuluh) kali pengujian dan 60% dari 100 kali pengujian

			karakter yang hampir sama, maka diperlukan proses pelatihan bertingkat - dari 100 penulisan kosakata aksara Lampung, 40 (empat puluh) kosakata diantaranya tidak dapat dikenali secara baik - deteksi tepi canny kurang maksimal mengenali karakter aksara karena hanya mampu mengenali tepinya saja - kegagalan dipengaruhi oleh data yang kurang bagus/ memilki kemiripan antara 1 dengan yang lainnya	
6	Rima Tri Wahyuningrum, Riza Mashita Wati, dan Aeri Rachmad (2011)	Pengenalan Pola Senyum Menggunakan *Backpropagation* Berbasis Ekstraksi Fitur *Principal Component Analysis* (PCA)	- Data yang digunakan sebanyak 250 data dari 10 orang - Citra *cropping* bagian mulut berukuran 39 x 25 pixel - PCA digunakan untuk mereduksi dimensi dari *image* yang diolah	82,67%

			- Menggunakan skenario *five fold cross validation* - Akurasi tertinggi saat menggunakan 10 *hidden layer* dan menggunakan nilai eigen dari 1 sampai 15	
7	Agung Wisnu Anggoro, Muhammad Zidny Naf'an, dan Elisa Usada (2019)	Identifikasi Citra Tanda Tangan Berdasarkan Grid Entropy dan PCA Menggunakan Multi Layer Perceptron	- Menggunakan data sebanyak 900 yang diperoleh dari 30 orang berbeda - menggunakan 2 *hidden layer* dengan variasi nilai *node* pada masing-masing *hidden layer* yaitu 10,20,30 dan 40 - 17 skenario berbeda - *Dataset* dibagi menjadi 10 *fold* dengan iterasi 10 kali - semakin banyak layer dan node yang gunakan belum tentu akan menghasilkan akurasi yang lebih tinggi - PCA untuk mereduksi dimensi dari percobaan grid	87,22%
8	Diyah Puspitaningrum,	Dampak Reduksi Sampel Menggunakan *Principal*	- 1060 sampel citra angka tulisan tangan	Pelatiha kesatu

	Dyan Kemala Sari dan Boko Susilo (2014)	*Component Analysis* (PCA) Pada Pelatihan Jaringan Syaraf Tiruan Terawasi (Studi Kasus : Pengenalan Angka Tulisan Tangan)	- 660 sebagai citra latih, 400 sebagai citra uji - Data diambil dari 106 orang yang berbeda - PCA memiliki peforma yang baik untuk kompresi data yang digabungkan dengan *backpropagation* - Lamanya pemrosesan dipengaruhi ukuran data dan jumlah segmen - Peforma PCA+ *backpropagation* lebih baik dari pada zoning+*backpropagation*	hasilnya 86% Pelatihan kedua hasilnya 86,25%.
9	Fiqih Ismawan (2015)	Hasil Ekstraksi Algoritma *Principal Component Analysis* (PCA) untuk Pengenalan Wajah dengan Bahasa Pemograman Java Eclips IDE	- Data yang digunakan sebanyak 1060 dari 106 orang (1 orang menulis 10 kali) - resolusi awal citra wajah berukuran 125 x 150 piksel. - pengujian sebanyak 5 citra wajah - (PCA) relatif mudah menangani sejumlah data yang cukup besar serta kemampuannya menangani data-data	86,5%

			dimensi yang kompleks	
10	John Pierre Haumahu (2019)	Implementasi Jaringan Syaraf Tiruan Untuk Pengenalan Pola Notasi Balok Menggunakan Metode *Backpropagation*	- 6 gambar dipakai sebagai gambar latih, dan 1 tidak dipakai sebagai gambar latih - data di *resize* menjadi 50x50 pixel - Menggunakan *epoch* 1000 untuk mengecilkan nilai *error* - Penentuan paramater dari *backpropagation* mempengaruhi akurasi - Ukuran citra 50 x 50 pixel	91,20%

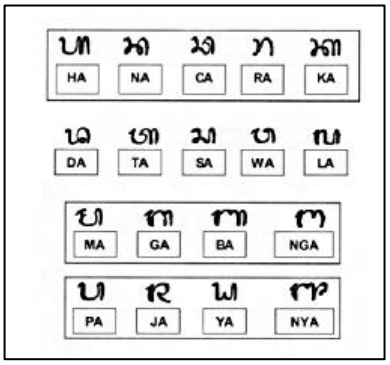
Berdasarkan penelitian-penelitian yang sudah dilakukan sebelumnya, diketahui bahwa hasil pengujian menggunakan ekstraksi *principal component analysis* memiliki akurasi tinggi begitu pula yang menggunakan metode ANN sebagai klasifikasinya. Oleh karena itu penelitian ini akan menggabungkan ekstraksi menggunakan PCA dengan metode ANN yang diharapkan mampu mengenali pola aksara Sasak dengan baik.

2.2 Dasar Teori

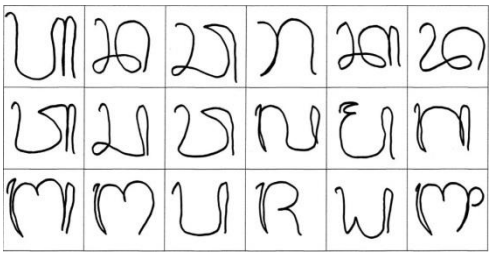
2.2.1 Aksara sasak

Banyaknya naskah yang ditemukan di Lombok dalam jumlah besar mengindikasikan bahwa tradisi tulis telah berkembang dengan baik sejak masyarakat Sasak mengenal tulisan Beberapa tulisan (aksara) yang digunakan dalam naskah-naskah di Lombok adalah aksara Jejawen, Arab, Bali, dan beberapa di antaranya (tapi jarang ditemukan) Bugis [2]. 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. Huruf Sasak terdiri dari 18 karakter dasar yang diadaptasi dari aksara Jawa dan Bali.



Gambar 2.1 Aksara Sasak digital



Gambar 2.2 Aksara Sasak tulisan tangan (*scan*)

2.2.2 Citra

Citra merupakan gambar dalam bidang dua dimensi yang berarti hanya memiliki dua sisi yaitu hanya panjang dan lebar saja. Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intesitas cahaya pada bidang dwimatra (dua dimensi). Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optic, seperti mata pada manusia, kamera, pemindai (*scanner*), dan lain-lain sehingga bayangan objek dalam bentuk citra dapat terekam [12].

2.2.3 Pengenalan Pola

Pengenalan pola adalah proses memberikan label berbagai golongan objek pada setiap piksel citra berdasarkan pemetaan jaringan keras dan perwilayahan berbagai jaringan lunak pada citra biometric [14] atau proses pengolahan data mentah yang akan diklasifikasikan sesuai ciri yang ada pada data.

2.2.4 *Principal Component Analysis*

Ekstraksi ciri merupakan suatu proses untuk mendapatkan ciri dari sebuah objek atau citra. Untuk mendapatkan ciri dari suatu citra membutuhkan suatu metode khusus. Salah satu metode yang digunakan dalam ekstraksi ciri sebuah citra adalah *Principal Component Analysis* (PCA).

PCA adalah sebuah metode untuk mengambil ciri-ciri penting dari sekumpulan data dengan melakukan dekomposisi terhadap data tersebut, sehingga menghasilkan koefisien-

koefisien yang tidak saling berkorelasi. PCA juga dikenal dengan transformasi *Kauhunen-Loeve* atau transformasi *Hotelling* atau teknik *Eigenface*. PCA merupakan alat teoritis yang penting dalam teori deteksi, pengenalan pola dan *image coding*. Tujuan dari PCA adalah mengambil variasi total pada karakter *alphanumeric* yang dilatihkan dan menjelaskan variasi tersebut dengan variabel yang jumlahnya lebih sedikit. Dengan kata lain PCA digunakan untuk merepresentasikan data dalam dimensi yang rendah sehingga waktu komputasi dapat dikurangi dan kompleksitas dari karakter *alphanumeric* yang tidak perlu, dapat dihilangkan. PCA menghasilkan *vector-vector eigen* atau *vector-vector* karakteristik yang kemudian akan digunakan untuk membentuk ruang *eigen*.

Dalam mereduksi dimensi data menjadi lebih rendah maka diperlukan penentuan *vector-vector eigen* yang dapat direduksi maupun *vector-vector eigen* yang tidak dapat direduksi dengan mengurutkan *vector-vector eigen* tersebut dari *vector eigen* yang mempunyai nilai *eigen* terbesar sampai yang terkecil. *Vector-vector eigen* yang dapat direduksi dalam *vector* yang mempunyai nilai *eigen* yang kecil. Hal ini dikarenakan nilai *eigen* yang kecil menandakan informasi yang dibawa oleh *vector eigen* tersebut tidak terlalu penting sehingga dapat direduksi tanpa mempengaruhi informasi penting karaktersitik *alphanumeric* tersebut[19]. Berikut adalah tahap – tahap penerepan *Principal Components Analys* [20]:

1. Menghitung nilai *Mean* dari setiap baris citra penelitian ($Xmean$). *Mean* citra dapat dihitung dengan persamaan (2-1)

$$Xmean_i = \frac{1}{N}\sum_{j=1}^{N} X_{(i,j)} \tag{2-1}$$

   Dimana :

   $Xmean_i$: Nilai *mean* pada kolom ke i matrik fitur citra

   N : Jumlah baris pada setiap kolom matrik fitur citra

   $X_{(i,j)}$: Nilai pada kolom ke i bari ke j matrik fitur citra

2. Merepresentasikan data dalam bentuk *mean corrected* data (A), yang menunjukan seberapa jauh perbedaan antara citra wajah dengan rata-rata citra. *Mean corrected* data dapat dihitung dengan persamaan (2-2)

$$A_{(i,j)} = X_{(i,j)} - Xmean_i \tag{2-2}$$

   Dimana :

   $A_{(i,j)}$: Nilai *mean corrected* data pada kolom ke i bari ke j matrik fitur citra

   $X_{(i,j)}$: Nilai pada kolom ke i bari ke j matrik fitur citra

   $Xmean_i$: Nilai *mean* pada kolom ke i matrik fitur citra

3. Mencari matrik kovarian (C) yang berfungsi ungtuk menyatakan hubungan penyebaran data dari dua variabel atau lebih. Matrik kovarian dapat dihitung dengan persamaan (2-3)

$$C = A\ x\ A^T \tag{2-3}$$

Dimana :

C : Matrik kovarian

A : Matrik *mean corrected* data

4. Mencari *eigen value* dan *eigen vector*, dimana *eigen value* merupakan nilai karakteristik suatu matrik sedangkan *eigen vector* merupakan *vector* karakteristik dari matrik yang selalu bersesuaian dengan *eigen value*. *Eigen value* dan *eigen vector* dapat dihitung dengan persamaan (2-4)

$$CV = \lambda V \tag{2-4}$$

Dimana :

C : Matrik kovarian

V : *Eigen vector*

λ : *Eigen value*

5. Mencari ciri PCA, *vector eigen* yang berkorelasi dengan *eigen value* terbesar. Dalam algoritma PCA tidak semua *eigen vector* digunakan, yang digunakan hanya *eigen vector* yang signifikan saja. Ciri PCA dapat dihitung dengan persamaan (2-5)

$$PC = A^T * V \tag{2-5}$$

Dimana :

PC : *Principal Components*

A : Matrik *mean corrected* data

V : *Eigen vector*

6. Langkah terakhir adalah melakukan transformasi data untuk menghasilkan data PCA. PCA dapat dihitung dengan persamaan berikut

$$PCA = A *\ PC \tag{2-6}$$

Dimana :

PCA : *Principal components analys*

PC : *Principal components*

A : Matrik *mean corrected* data

2.2.5 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan bisa dibayangkan seperti otak buatan di dalam cerita-cerita fiksi ilmiah. Otak buatan ini dapat berpikir seperti manusia, dan juga sepandai manusia dalam

menyimpulkan sesuatu dari potongan-potongan informasi yang diterimanya. Khayalan manusia tersebut mendorong para peneliti untuk mewujudkannya. Komputer diusahakan agar bisa berpikir sama seperti cara berpikir manusia. Caranya adalah dengan melakukan peniruan terhadap aktivitas-aktivitas yang terjadi di dalam sebuah jaringan syaraf biologis [21].

Salah satu contoh pengambilan ide dari jaringan syaraf biologis adalah adanya elemen-elemen pemrosesan pada jaringan syaraf tiruan yang sering terhubung dan beroperasi secara paralel. Ini meniru jaringan syaraf biologis yang tersusun dari yang tersusun dari sel-sel syaraf (*neuron*). Cara kerja dari elemen pemrosesan jaringan syaraf tiruan juga sama seperti cara *neuron* meng-*encode* informasi yang diterimanya.

Hal yang perlu mendapat perhatian adalah bahwa jaringan syaraf tiruan tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran atau kesimpulan yang ditarik oleh jaringan didasarkan pada pengalaman selama mengikuti proses pelatihan.

Pembagian arsitektur jaringan syaraf tiruan bisa dilihat dari kerangka kerja dan skema interkoneksi. Kerangka kerja jaringan syaraf tiruan bisa dilihat dari jumlah lapisan (*layer*) dan jumlah *node* pada setiap lapisan. Lapisan-lapisan penyusun jaringan syaraf tiruan dapat dibagi menjadi tiga, yaitu [21]:

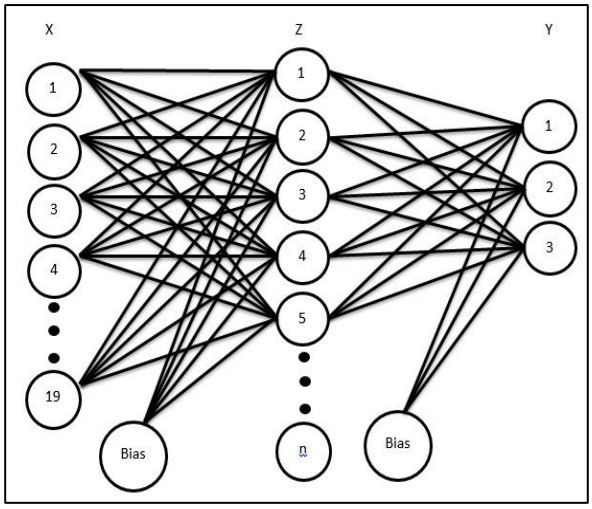
1. Lapisan masukan (*input layer*)

   Node-node di dalam lapisan *input* disebut unit-unit *input*. Unit-unit *input* menerima *input* dari dunia luar. *Input* yang dimasukkan merupakan penggambaran suatu masalah.
2. Lapisan tersembunyi (*hidden layer*)

   Node-node di dalam lapisan tersembunyi disebut unit-unit tersembunyi. *Output* dari lapisan ini tidak secara langsung dapat diamati.
2. Lapisan keluaran (*output layer*)

   Node-node pada lapisan *output* disebut unit-unit *output*. Keluaran atau *output* dari lapisan ini merupakan *output* jaringan syaraf tiruan terhadap suatu permasalahan.

Gambar 2.3 merupakan salah satu contoh arsitektur jaringan syaraf tiruan *multilayer* yang terdiri dari sebuah lapisan *input* (x), sebuah lapisan tersembunyi (z), dan sebuah lapisan *output* (y) [18].



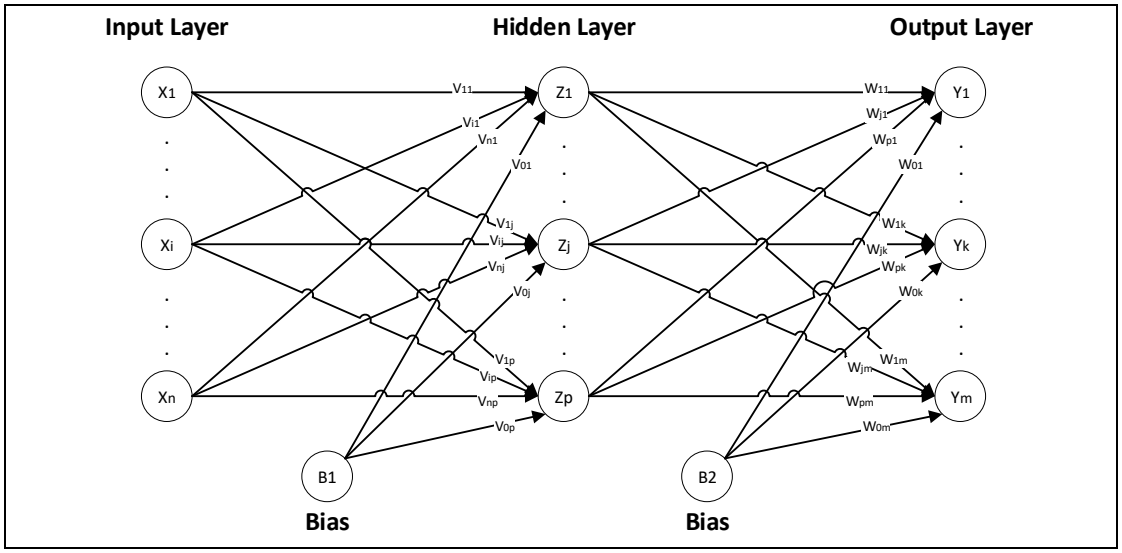
Gambar 2.3 Sebuah arsitektur jaringan syaraf tiruan

2.2.6 Jaringan Syaraf Tiruan (JST) *Backpropagation*

Jaringan syaraf tiruan 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 [22].

2.2.6.1 Arsitektur *Backpropagation*

Backpropagation memiliki beberapa unit (*neuron*) yang ada dalam satu atau lebih *layer* tersembunyi. Gambar 2.2 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 2.4 Arsitektur jaringan *backpropagation*

2.2.6.2 Fungsi Aktivasi

Dalam *backpropagation*, fungsi aktivasi yang dipakai harus memenuhi beberapa syarat yaitu: kontinu, terdiferensial dengan mudah, dan merupakan fungsi yang tidak turun secara monotonis. Salah satu fungsi yang memenuhi ketiga syarat tersebut sehingga sering dipakai adalah fungsi *sigmoid biner* yang memiliki *range* (0, 1) [22]. Persamaan fungsi aktivasi *sigmoid biner* yaitu sebagai berikut:

$$f(\mathrm{x}) = \frac{1}{1+e^{-x}} \quad (2\text{-}7)$$

Dan jika fungsi f(x) diturunkan menjadi persamaan (2-8)

$$f'(\mathrm{x}) = f(\mathrm{x})(1 - f(\mathrm{x})) \quad (2\text{-}8)$$

2.2.6.3 Algoritma *Backpropagation*

Algoritma pelatihan *backpropagation* terdiri dari proses *feedforward* dan *backpropagation*. Algoritma tersebut yaitu sebagai berikut [23]:

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) menggunakan

persamaan (2-9)

$$Z\_net_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

(2-9)

Gunakan fungsi aktivasi (2-7) untuk menghitung sinyal *output*-nya megunakan rumus (2-10):

$$Z_j = (\text{Z\_net}) \quad (2\text{-}10)$$

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) dengan persamaan (2-11)

$$Y\_net_k = w_{0k} + \sum_{j=1}^{p} z_j w_{jk} \quad (2\text{-}11)$$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya dengan persamaan (2-12)

$$Y_k = (y\_net_k) \quad (2\text{-}12)$$

Fase II: *Backpropagation*

Langkah 6: Hitung faktor δ unit keluaran berdasarkan kesalahan di setiap unit keluaran (y_k, k = 1, 2,…, m) dengan persamaan (2-13)

$$\delta_k = (t_k - y_k)\, f'(y\_net_k) \quad (2\text{-}13)$$

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 dengan persamaan (2-14)

$$\Delta w_{jk} = \alpha. .z_j \quad (2\text{-}14)$$

Kemudian dengan persamaan (2-15) hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k})

$$\Delta w_{0k} = \alpha.\delta_k \quad (2\text{-}15)$$

Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi (z_j, j = 1, 2,…, p) dengan persamaan (2-16)

$$\delta\_net_j = \sum_{k=1}^{m} \delta_k . w_{jk} \quad (2\text{-}16)$$

hitung faktor δ unit tersembunyi dengan persamaan (2-17):

$$\delta_j = \delta\_net_j\, f'(z\_net_j) \quad (2\text{-}17)$$

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}) dengan persamaan (2-18)

$$\Delta v_{ij} = \alpha.\delta_j.x_i \quad (2\text{-}18)$$

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai

v_{0j}) dengan persamaan (2-19)

$$\Delta v_{0j} = \alpha . \delta_j \quad (2\text{-}19)$$

Fase III: Perubahan Bobot

Langkah 8: Tiap-tiap unit *output* (, k = 1, 2,…, m) memperbaiki bobotnya (j = 0, 1, 2,..., p) dengan persamaan (2-20)

$$w_{jk}\ (baru) = w_{jk}\ (lama) + \Delta w_{jk} \quad (2\text{-}20)$$

Tiap-tiap unit tersembunyi (Z_j, j = 1, 2, 3,..., p) memperbaiki bobotnya (j = 0, 1, 2, 3,..., n) dengan persamaan (2-21)

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

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 [20].

BAB III
METODE PENELITIAN

3.1 Alat dan Bahan Penelitian

Pada penelitian pengenalan pola tulisan tangan aksara Sasak dibutuhkan alat dan bahan untuk memproses data dan menunjang kegiatan selama penelitian berlangsung berupa *software* dan *hardware* sebagai berikut :

3.1.1 Alat Penelitian

Berikut perangat keras dan perangkat lunak yang digunakan selama penelitian berlangsung :

1. Perangkat keras

   Perangkat keras yang digunakan pada penelitian ini adalah komputer dengan spesifikasi sebagai berikut:

   a. Prosesor Intel pentium(R) CPU B960 2.2GHz
   b. Memory 2 GB DDR3

2. Perangkat Lunak

   Perangkat lunak yang digunakan pada penelitian ini yaitu:

   a. Sistem operasi windows 7
   b. Bahasa Python
   c. JupyterLab
   d. Microsoft office word 2010
   e. Microsoft office excel 2010
   f. Microsoft power point 2010

3.1.2 Bahan Penelitian

Bahan penelitian yang digunakan yaitu kumpulan tulisan tangan orang yang akan di *scan* menggunakan *scanner* sebagai *dataset* yang terdiri dari kategori yang pernah belajar aksara dan dan yang belum belajar aksara dengan jenjang pendidikan SD, SMP, SMA dan Universitas. Data diperoleh dari 10 orang per-jenjang pendidikan, setiap orang menulis 18 karakter aksara sebanyak 15 kali di kertas yang berbeda dan ditambah dengan data peneliti sebelumnya sebanyak 2700 [4] data sehingga jumlah data yang didapat sebagai berikut,

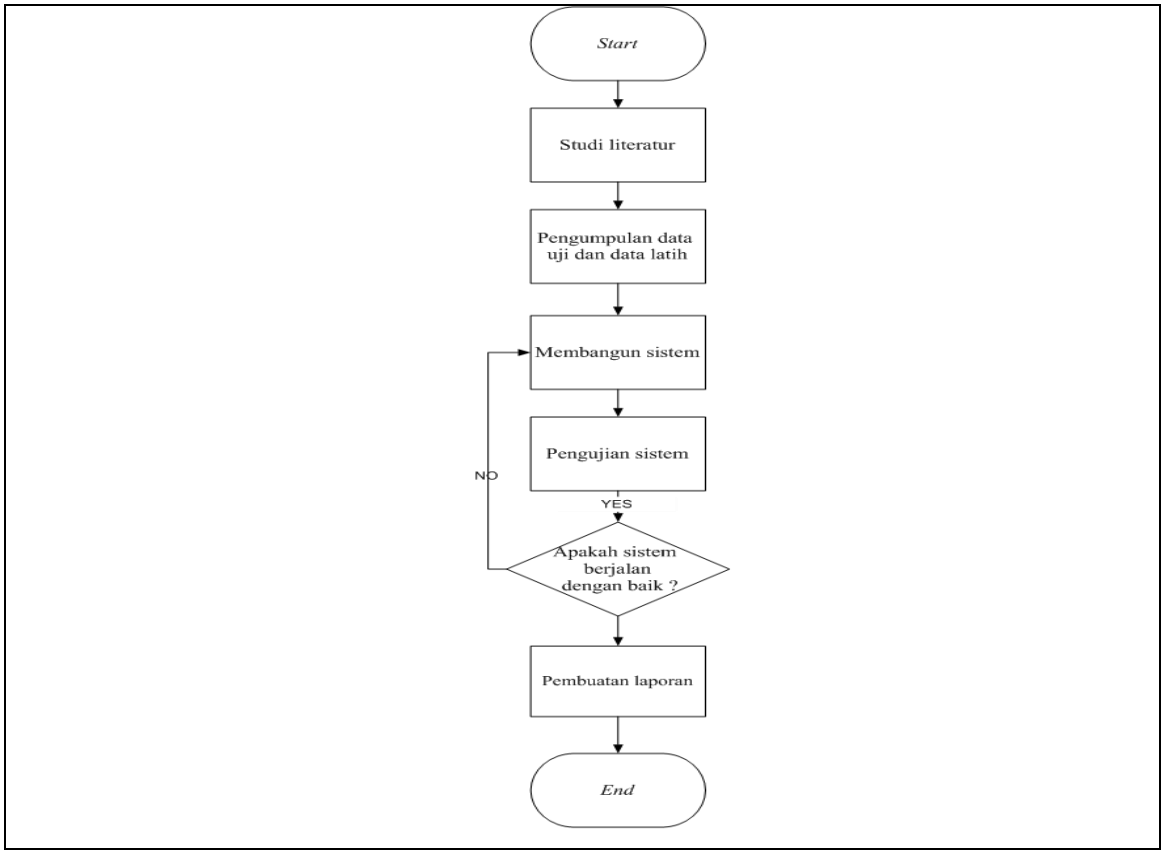
Data = 4 (SD, SMP, SMA, Univeritas) × 10 (orang) × 18 (karakter aksara) × 15 + 2700 (data peneliti sebelumnya) = 13.500 data.

3.2 Rencana Penelitian

Langkah awal dalam pembuatan penelitian pengenalan pola ini yaitu diawali dengan studi literatur untuk mendukung penilitian, menambah wawasan dan pengetahuan tentang permasalahan apa saja yang dapat di angkat dan sesuai dengan penelitian ini. Adapun materi yang dipelajari yaitu berkaitan dengan ekstraksi fitur PCA dan seleksi fitur menggunakan ANN dan akurasi yang dapat di hasilkan di masing-masing penerapannya. Studi literatur dilakukan dengan mencari jurnal, buku cetak, dan *e-book* melalui internet.

Pada tahap kedua yaitu mengumpulkan *dataset* dengan meminta 10 orang dari masing-masing jenjang pendidikan SD, SMP, SMA dan Universitas untuk menulis aksara Sasak menggunakan spidol yang sudah disediakan. Aksara Sasak yang di tulis terdiri dari 18 huruf yaitu ha, na, ca, ra, ka, da, ta, sa, wa, la, ma, ga, ba, nga, pa, ja, ya, nya sebanyak 15 kali. *Dataset* yang diambil dengan cara menulis pada selembar HVS ukuran A4 dengan tabel yang terdiri dari 3 baris dan 6 kolom dengan masing-masing kotak yang memiliki panjang dan lebar sebesar 4 cm yang membentuk persegi sama sisi, hal ini dilakukan agar huruf aksara yang ditulis konsisten memiliki ukuran yang sama sehingga diberi batasan panjang dan lebar penulisan per-hurufnya.

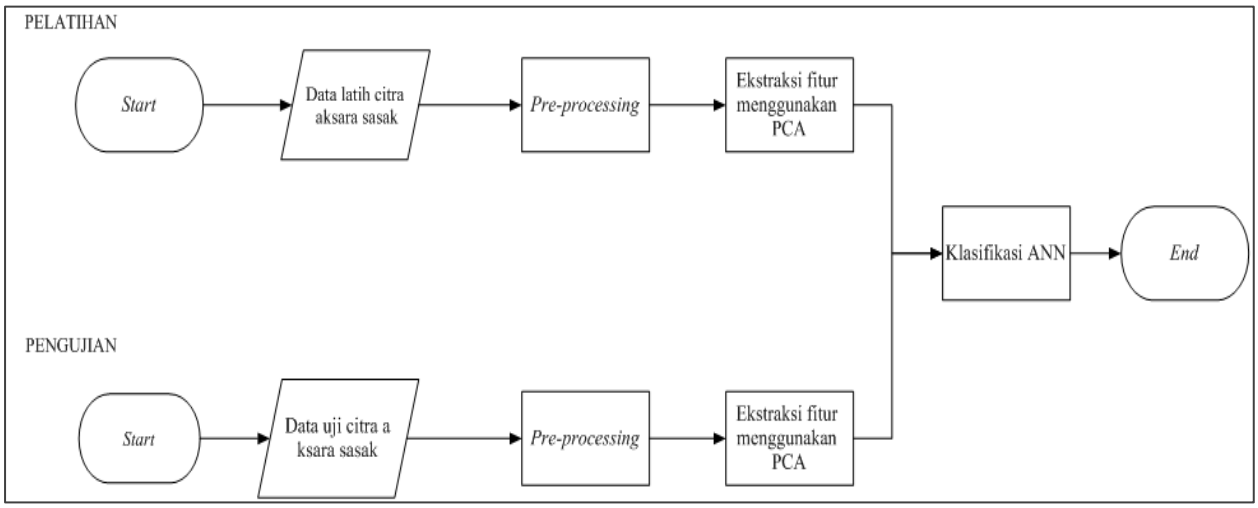
Selanjutnya yaitu membangun sistem pengenalan pola aksara Sasak sesuai dengan rencana yang telah dibuat. Tahap *training* data dilakukan agar sistem dapat mengenali fitur yang ada pada setiap karakter aksara dan selanjutnya yaitu tahap pengujian sistem, yanag dimana sistem diuji apakah sistem tersebut dapat berjalan dengan baik sesuai dengan tujuan pembuatan sistem. Apabila sistem belum sesuai maka sistem akan diperbaiki dan diperbaharui dengan kembali ke tahap pengembangan sistem. Tahap akhir yaitu membuat laporan. Diagram alir proses pembuatan sistem dapat digambarkan seperi pada Gambar 3.1.



Gambar 3.1 Diagram alur penelitian

3.3 Rancangan Algoritma

Sistem yang akan dikembangkan untuk pengenalan pola aksara Sasak yaitu terdiri dari 2 proses inti yaitu proses pelatihan dan proses pengujian (klasifikasi) yang dapat dilihat pada Gambar 3.2.



Gambar 3.2 Blok diagram sistem

Berikut penjelasan dari aliran proses yang ada pada sistem pada Gambar 3.2 :

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. Selanjutnya akan dilakukan tahap *pre-processing* yang merupakan proses untuk memperbaiki citra untuk menghilangkan *noise* pada gambar. *Pre-processing* yang dilakukan yaitu *resize* untuk mengubah ukuran pixel citra menjadi 64x64 dan proses *greysacle* dilakukan untuk mengubah menjadi citra biner, agar menghasilkan nilai 1 untuk hitam dan 0 untuk putih.
3. Gambar akan di ekstraksi menggunakan PCA untuk mendapatkan ciri dari masing-masing citra aksara Sasak dan kemudian hasil tersebut menjadi data latih untuk sistem.
4. Proses klasifikasi yang digunakan pada penelitian ini adalah ANN.
5. Hasil klasifikasi tersebut di simpan oleh sistem sebagai proses pelatihan.

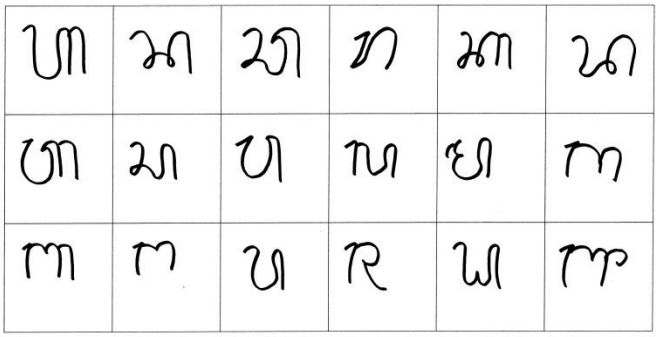
b. Proses pengujian
1. Citra di *input* ke dalam sistem sebagai proses pengujian terhadap sistem. Citra yang di-*input* merupakan *dataset* yang di masukan secara random.
2. citra yang di masukan di proses kembali seperti pada proses pelatihan yaitu di *resize* terlebih dahulu lalu di *greyscale*.
3. Kemudian citra akan di ekstraksi menggunakan PCA.
4. Hasil ekstraksi fitur PCA akan di-*load* dan di klasifikasikan menggukan ANN yang menghasilkan nilai akurasi dalam bentuk persentase dan jenis huruf.

3.3.1 Data *acquisition* (pengambilan data)

Data *acquisition* adalah proses untuk menggumpulkan atau mengambil data yang dibutuhkan untuk proses penelitian. Data yang diambil nantinya akan digunakan untuk proses *training* dan *testing* pada sistem. Data yang diambil pada penelitian ini berupa data tulisan tangan orang yang ditulis menggunakan spidol yang sudah disediakan pada selembar kertas HVS ukuran a4. Pada lembaran HVS yang diberikan sudah disediakan *template* tabel yang terdiri dari 3 baris dan 6 kolom yang berarti ada 18 kotak yang tersedia untuk menulis aksara sesuai dengan jumlah huruf aksara Sasak sebanyak 18 huruf. Pada tabel tersebut, 1 kotak memiliki ukuran panjang 4 cm dan dengan lebar 4 cm yang membentuk persegi sama sisi. Hal

ini untuk mempermudah penulis agar setiap hutuf yang dituliskan memiliki ukuran yang relative sama dengan diberi batasan panjang dan lebar.

Tulisan tangan yang digunakan untuk data pada penelitian diambil dari kategori tulisan SD, SMP, SMA dan kuliah yang pernah belajar aksara dan yang tidak mempelajari aksara. Masing-masing dari kategori tersebut dicari sumber sebanyak 10 orang untuk menulis aksara Sasak sebanyak 15 kali di lembar HVS yang berbeda. Data tesebut akan di-*scan* menggunakan *scanner* dengan *high resolution* seperti pada Gambar 3.3. Data hasil *scanner* dengan format .jpg tersebut akan di *crop* sesuai dengan banyaknya huruf aksara sebanyak 18 huruf. Gambar tersebut akan di *resize* resolusinya agar seluruh data yang didapat seragam.



Gambar 3.3 Contoh hasil *scan* aksara Sasak

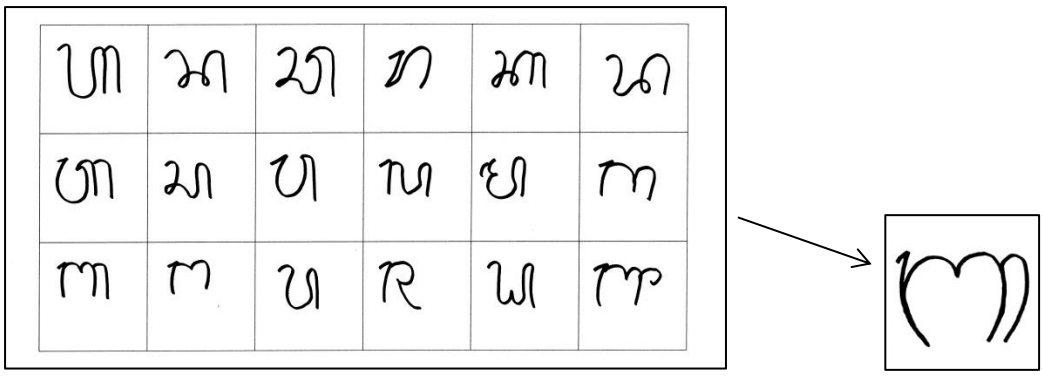
Sedangkan pada penelitian sebelumnya terdapat 2700 data yang terkumpul dengan sistematika penambilan data dengan menulis tangan menggunakan spidol. Data ini akan digunakan sebagai data tambahan dalam proses penelitian ini. Jadi, pengambilan data pada penelitian ini memiliki dua sumber yaitu data yang diambil sendiri dan data pada penelitian sebelumnya sehingga data yang digunakan sebanyak 13.500 jika ditambah dengan 10.800 data yang diperoleh dari SD, SMP, SMA dan Universitas.

3.3.2 *Pre-processing*

Pre-processing 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*

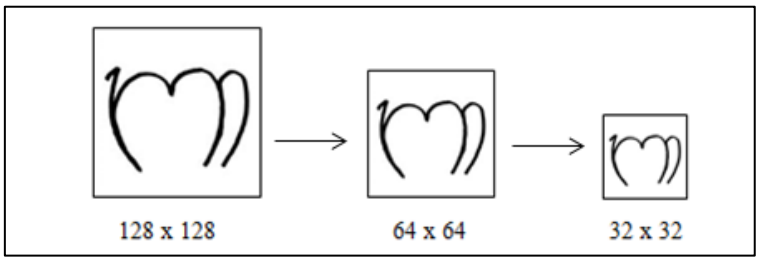
   Cropping adalah proses pemotongan citra pada elemen tertentu pada area citra. Proses ini bertujuan untuk mengambil elemen yang diinginkan dari citra yaitu untuk memisahkan 18 huruf aksara yang ada pada 1 citra sebelumnya dan dibagi menjadi 18 citra berbeda.



Gambar 3.4 Contoh *cropping*

2. *Resize*

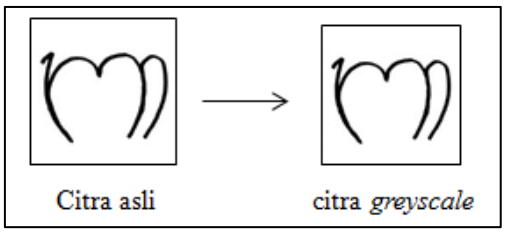
   Resize merupakan proses pengubahan pixel citra. Proses ini dilakukan untuk mengubah citra pixel 256x256 menjadi 64x64 untuk mempermudah proses pada sistem. Semakin kecil pixel pada citra maka proses yang ada pada sistem akan lebih cepat.



Gambar 3.5 Contoh *resize*

3. *Greyscale*

   Greyscale adalah merubah citra warna menjadi citra berwana keabuan. *Greyscale* 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 *greyscale* untuk meringkan kinerja sistem saat proses pelatihan dan pengujian.



Gambar 3.6 Contoh *greyscale*

4. Metode DCT (Discreate Cosine Transform)

   Penambahan metode DCT[24] karena pada proses perhitungan determinan di ekstraksi fitur PCA menghasilkan matriks singular atau matriks yang tidak memiliki nilai

determinan. Hal ini disebabkan oleh *background* citra yang memiliki dominan berwarna putih dari pada hitam (tulisan aksara). Jika nilai *mean* dihitung dari citra tersebut, maka nilai mean yang didapatkan bernilai satu dan menyebabkan determinan menghasilkan nilai 0. Dengan adanya metode DCT ini membantu mengatasi matriks singular dan mengumpulkan informasi fitur penting pada citra aksara. Berdasarkan penelitian sebelumnya yang menggunakan DCT disebutkan bahwa DCT memiliki 2 tujuan utama dalam penggunannya yaitu merangkum fitur tanpa menghapus informasi pada gambar dan membangun proses pelatihan yang lebih sederhana[24]. Jumlah koefisien yang digunakan dalam tahapan pengujian yaitu dari 64-256. Koefisien ini didapatkan berdasarkan penelitian sebelumnya.

5. Reduksi dimensi

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

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

Gambar 3.3 merupakan contoh reduksi dimensi pada matriks 3x3 dengan persamaan 3-1:

$$\text{C1} = \begin{pmatrix} 111 & 123 & 142 \\ 153 & 98 & 144 \\ 150 & 168 & 138 \end{pmatrix} \quad \text{C2} = \begin{pmatrix} 102 & 104 & 152 \\ 99 & 96 & 105 \\ 114 & 120 & 117 \end{pmatrix}$$

Matriks A direduksi menjadi 1 dimensi menjadi

$$\text{C1} = \begin{bmatrix} 111 \\ 123 \\ 142 \\ 153 \\ 98 \\ 144 \\ 150 \\ 168 \\ 138 \end{bmatrix} \quad \text{C2} = \begin{bmatrix} 102 \\ 104 \\ 152 \\ 99 \\ 96 \\ 105 \\ 114 \\ 120 \\ 117 \end{bmatrix}$$

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)

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

Pada tabel 3.1 yang berisi 8 citra berbeda yang sudah di reduksi menjadi 1 dimensi disusun menggunakan persamaan (3-2).

Tabel 3.1 Matriks A

Citra	Fitur								
c1	111	123	142	153	98	144	150	168	138
c2	102	104	152	99	96	105	114	120	117
c3	180	180	183	186	165	153	144	153	171
c4	117	126	141	179	123	144	177	147	168
c5	164	142	146	149	152	146	143	109	146
c6	123	137	171	141	137	163	134	168	134
c7	158	138	149	149	146	155	152	149	149
c8	121	152	179	152	152	152	152	138	146

5. Normalisasi

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

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

Dimana :

$N_{(i,j)}$: Nilai pada kolom ke i baris ke j matrik fitur citra

$fitur_{(i,j)}$: Nilai fitur pada kolom ke i baris ke j matrik fitur citra

$min_{(j)}$: Nilai minimum pada kolom j matrik fitur citra

$max_{(j)}$: Nilai maksimum pada kolom j matrik fitur citra

Berikut contoh perhitungan menggunakan persamaan (3-3)

$$N_{(1,1)} = \frac{111 - 98}{168 - 98} = 0{,}185714286$$

$$N_{(2,1)} = \frac{123 - 98}{168 - 98} = 0{,}357142857$$

Selanjutnya setiap baris dan kolom dilakukan perhitungan seperti di atas pada matriks A. Tabel 3.2 merupakan hasil normalisasi dari matriks A.

Tabel 3.2 Normalisasi matriks A

Fitur								
0.1857143	0.3571429	0.628571	0.785714	0	0.657143	0.742857	1	0.571429
0.1071429	0.1428571	1	0.053571	0	0.160714	0.321429	0.428571	0.375
0.8571429	0.8571429	0.928571	1	0.5	0.214286	0	0.214286	0.642857
0	0.1451613	0.387097	1	0.096774	0.435484	0.967742	0.483871	0.822581
1	0.6	0.672727	0.727273	0.781818	0.672727	0.618182	0	0.672727
0	0.2916667	1	0.375	0.291667	0.833333	0.229167	0.9375	0.229167
1	0	0.55	0.55	0.4	0.85	0.7	0.55	0.55
0	0.5344828	1	0.534483	0.534483	0.534483	0.534483	0.293103	0.431034

3.3.3 Ekstraksi Fitur

Proses ekstrasi fitur merupakan proses untuk meng-ekstrak citra agar memperoleh fitur ciri dari masing-masing citra. Proses ekstraksi yang digunakan pada penelitian ini menggunakan metode PCA. Berikut contoh perhitungan metode PCA :

1. Menghitung nilai mean dari setiap kolom

   Tahap awal yang perlu dilakukan yaitu menghitung rata-rata pada setiap kolom matrik A menggunakan persamaan (3-4). Berikut persamaan untuk menghitung nilai rata-rata :

$$A\ mean_i = \frac{1}{N}\sum_{j=1}^{N} A_{(i,j)} \quad (3\text{-}4)$$

   Berikut contoh perhitungan rata-rata dari tiap kolom yang ada pada matriks A menggunakan persamaan (3-4)

$$A\ mean_1 = \frac{0.1071429+\ 0.8571429+0+1+0+1+0}{8} = 0{,}39375$$

$$A\ mean_2 = \frac{0.3571429+0.1428571+0.8571429+0.1451613+0.6+0.2916667+0+0.5344828}{8} = 0{,}36606$$

   Perhitungan rata-rata dilakukan disemua kolom yang ada pada matriks A. Berikut hasil rata-rata tiap kolom matriks A.

Tabel 3.3 Rata-rata kolom

Rata-rata kolom								
0.39375	0.366057	0.770871	0.628255	0.325593	0.544771	0.514232	0.488416	0.536849

2. Menghitung *mean corrected* data

   Selanjutnya menghitung *mean corrected* data matriks A untuk mengetahui perbedaan antara citra dengan rata-rata citra yang sudah dihitung sebelumnya. Persamaan (3-5) untuk menghitung *mean corrected* data sebagai berikut :

$$X = A_{(i,j)} - Amean_{(i)} \quad (3\text{-}5)$$

   Contoh perhitungan menggunakan persamaan (3-5) untuk mencari nilai mean corrected data sebagai berikut :

$$\begin{aligned} X &= A_{(1,1)} - Amean_{(1)} \\ &= 0.1857143 - 0{,}39375 \\ &= -0.20804 \end{aligned}$$

$$\begin{aligned} X &= A_{(1,2)} - Amean_{(1)} \\ &= 0.8571429 - 0{,}39375 \\ &= -0.28661 \end{aligned}$$

   Perhitungan yang sama dilakukan di setiap nilai yang ada pada matrik A. Sehingga hasilnya seperti berikut :

Tabel 3.4 Matriks A *corrected data*

matriks A (*corrected data*)								
-0.20804	-0.00891	-0.1423	0.157459	-0.32559	0.112372	0.228625	0.511584	0.034579
-0.28661	-0.2232	0.229129	-0.57468	-0.32559	-0.38406	-0.1928	-0.05985	-0.16185
0.463393	0.491086	0.157701	0.371745	0.174407	-0.33049	-0.51423	-0.27413	0.106008
-0.39375	-0.2209	-0.38377	0.371745	-0.22882	-0.10929	0.45351	-0.00455	0.285731
0.60625	0.233943	-0.09814	0.099018	0.456225	0.127956	0.103949	-0.48842	0.135878
-0.39375	-0.07439	0.229129	-0.25326	-0.03393	0.288562	-0.28507	0.449084	-0.30768
0.60625	-0.36606	-0.22087	-0.07826	0.074407	0.305229	0.185768	0.061584	0.013151
-0.39375	0.168426	0.229129	-0.09377	0.20889	-0.01029	0.02025	-0.19531	-0.10581

3. Menghitung matriks kovarian

   Setelah nilai mean *corrected* di dapatkan, selanjutnya masuk ke tahap perhitungan kovarian. Nilai kovarian bertujuan untuk melihat hubungan antara beberapa *variable* yang terkait dan tingkat ketelitian berikut persamaannya :

$$C_{(i,j)} = A_{(i,j)}\, X\, A_{(i,j)}{}^{T} \tag{3-6}$$

   Berikut hasil perhitungan matriks kovarian dengan persamaan (3-6)

Tabel 3.5 Matriks kovarian

matriks kovarian							
0.522222	-0.078919	-0.41275	0.370488	-0.45422	0.207498	-0.01925	-0.13507
-0.07892	0.835184	-0.25139	-0.15635	-0.51585	0.305604	-0.28077	0.142497
-0.41275	-0.25139	1.109421	-0.41873	0.549272	-0.38742	-0.16167	-0.02673
0.370488	-0.156352	-0.41873	0.840946	-0.2461	-0.25362	-0.06484	-0.0718
-0.45422	-0.515853	0.549272	-0.2461	0.934039	-0.57301	0.359852	-0.05398
0.207498	0.305604	-0.38742	-0.25362	-0.57301	0.739237	-0.18606	0.147776
-0.01925	-0.28077	-0.16167	-0.06484	0.359852	-0.18606	0.69362	-0.34089
-0.13507	0.142497	-0.02673	-0.0718	-0.05398	0.147776	-0.34089	0.338195

4. Menghitung nilai eigen (*eigen values*) dan eigen vektor (*eigen vector*)

   Selanjutnya matriks kovarian yang sudah di dapatkan, dihitung kembali untuk mencari nilai *eigen* dan *vector eigen*-nya. Untuk menghitung nilai eigen dan *vecktor eigen* diperlukan persamaan sebagai berikut :

$$CV = \lambda V \tag{3-7}$$

   Dari persamaan di atas diketahui bahwa nilai *eigen* harus terlebih dahulu dihitung untuk mendapatkan *eigen* vektornya. Nilai eigen dapat dihitung menggunakan $\det(C - \lambda I) = 0$, yang dimana I merupakan matriks identitas yang dikalikan dengan eigen (λ) agar membentuk sebuah matriks yang bisa melakukan pengurangan dengan matriks kovarian.

   Di misalkan matriks kovarian dengan nilai $C = \begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix}$

Matriks kovarian dimasukkan ke dalam rumus $\det(C - \lambda I) = 0$ menjadi :

$$\begin{bmatrix} 4 & -5 \\ 2 & -3 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = 0$$

$$\begin{bmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{bmatrix} = 0$$

$$\det(C - \lambda I) = \lambda^2 - \lambda - 2$$

$$= (\lambda + 1)(\lambda - 2)$$

Sehingga di dapatkan nilai *eigen* nya yaitu λ= -1 dan λ=2. Langkah selanjutnya yaitu mensubstitusikan masing-masing nilai eigen ke dalam matriks.

Jika λ= -1 maka substitusikan ke dalam $(C - \lambda I)v = 0$

$$\begin{bmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 4+1 & -5 \\ 2 & -3+1 \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 3 & -5 \\ 2 & -2 \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = 0$$

Untuk menghitung *vector eigen* diperlukan matriks diperluas dan OBE maka didapatkan nilai v1= v2= 1.

Sedangkan untuk λ= 2 di substitusikan ke dalam $(C - \lambda I)v = 0$

$$\begin{bmatrix} 4-\lambda & -5 \\ 2 & -3-\lambda \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 4-2 & -5 \\ 2 & -3-2 \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = 0$$

$$\begin{bmatrix} 2 & -5 \\ 2 & -5 \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = 0$$

Di dapatkan hasil v1=5 dan v2=5.

Untuk hasil kovarian dari matriks A yang di cari nilai *eigen* dan vektornya, hasil yang di dapat sebagai berikut :

Tabel 3.6 *Eigen value* dan *eigen vector*

Eigen Value							
2.539897	1.504889	0.89347	0.518963	0.455156	0.080437	0.007021	0.01303
Eigen Vector							
-0.28767	-0.306	0.067176	0.518003	-0.37381	0.757917	0.031576	0.186265
-0.3478	0.395433	-0.03912	-0.45135	-0.16612	0.181212	0.210232	0.307049
0.501694	0.352946	0.348933	0.47302	-0.53234	-0.00668	-0.04693	0.218156
-0.18423	-0.55443	0.536725	-0.16681	0.058915	-0.13182	0.251552	0.424587
0.572025	-0.1042	-0.11723	-0.22656	0.344245	0.564933	0.604391	0.572476
-0.38095	0.301882	-0.34417	0.359497	0.047642	0.035361	0.449482	0.557002
0.18173	-0.38221	-0.64149	-0.04151	-0.144	-0.11682	-0.326	0.046109

-0.07267	0.26449	0.201046	-0.30675	0.635846	0.203116	-0.46446	-0.05527

5. Menghitung ciri PCA

   Untuk menghitung nilai PC (*principal component*) menggunakan persamaan berikut :

$$PC = A^T * V \tag{3-9}$$

   Dari persamaan di atas diketahui bahwa A merupakan matriks *corrected data* yang telah dihitung sebelumnya dan dikalikan dengan matriks V (*vector eigen*) yang telah dipilih dari 6 nilai eigen tertinggi. Hasil yang di dapatkan sebagai berikut :

Tabel 3.7 Nilai PC

1.100129	-0.18572	-0.45603	0.123189	-0.29223	0.016968
0.450661	0.34789	0.327791	0.249051	0.002767	0.185142
-0.08915	0.627004	-0.04908	0.004975	0.063197	0.008207
0.418303	-0.43199	0.4692	0.373313	-0.16572	0.000792
0.608798	0.128329	-0.11863	-0.08376	0.347017	0.013499
-0.02494	-0.28791	-0.46359	0.158691	0.198587	0.07693
-0.13991	-0.74167	0.057705	-0.2534	0.25536	0.113046
-0.68813	-0.16317	-0.23746	0.492555	-0.31543	0.072415
0.251902	-0.33564	0.259259	-0.01604	-0.0627	0.001358

6. Menghitung PC

   Menghitung nilai PCA adalah langkah terakhir dari proses esktraksi. Untuk menghitung nilai PCA maka digunakan persamaan sebagai berikut :

$$PCA = A * PC \tag{3-10}$$

   Hasil dari perhitungan PCA dapat dilihat pada tabel di bawah ini

Tabel 3.8 Nilai PCA

-0.73066	-0.46049	0.060019	0.268824	-0.17014	0.060965
-0.83797	0.64338	-0.04556	-0.31599	-0.01661	-0.10505
1.274252	0.531144	0.311762	0.24548	-0.2423	-0.00054
-0.46793	-0.83436	0.479547	-0.08657	0.026815	-0.0106
1.452886	-0.15681	-0.10474	-0.11757	0.156685	0.045442
-0.96756	0.4543	-0.30751	0.186565	0.021685	0.002844
0.461575	-0.57519	-0.57316	-0.02154	-0.06554	-0.0094
-0.18458	0.398028	0.179629	-0.15919	0.289409	0.016338

PCA memproses data sebanyak 10.800 citra yang memiliki 18 kelas berbeda sesuai dengan jumlah karakter aksara sasak. Dalam penomoran ID kelas pada dataset citra disesuaikan dengan urutan folder yang ada pada PC, yang dimana ID-nya dimulai dari angka 0. Di dalam proses ekstraksi fitur PCA hasil yang didapatkan berupa matriks proyeksi yang tersusun

berdasarkan eigen vektor yang memiliki nilai eigen terbesar sampai nilai terkecil. Jumlah eigen terbaik dipilih berdasarkan pengujian dengan cara mencoba eigen dengan kelipatan 64 dengan pengujian nilai eigen maksimum sebesar 256. Setelah mendapat nilai eigen terbaik maka nilai eigen vektor juga dapat dihasilkan. Eigen vektor tersebut disimpan dan akan dilanjutkan ke tahapan klasifikasi menggunakan *backpropagation*.

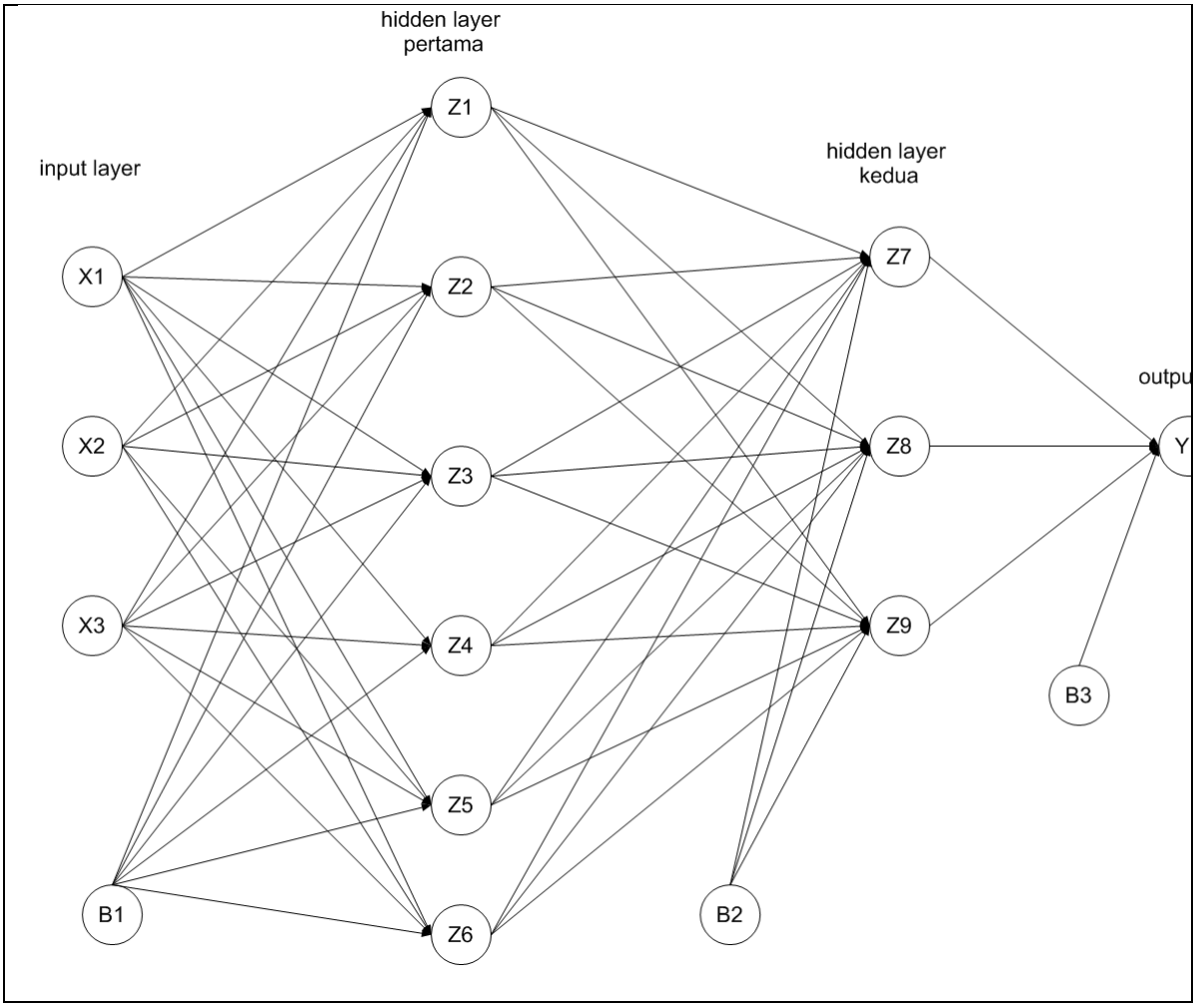
3.3.4 Klasifikasi

Proses klasifikasi yang digunakan pada peneliatian ini yaitu menggunakan metode *backpropagation*. *Backpropagation* adalah jenis pembelajaran terkontrol dimana menggunakan pola penyesuaian bobot untuk mencapai nilai kesalahan yang minimum antara keluaran hasil prediksi dengan keluaran yang sebenarnya. Metode ini merupakan salah satu metode yang sangat baik dalam menangani masalah pengenalan pola-pola kompleks [7]. Berikut diberikan contoh data pada Tabel 3.9 yang terdiri dari 4 data yang memiliki 3 *input* dan nilai target. Proses *backpropagation* dilakukan untuk klassifikasi data tersebut sesuai dengan target yang ditentukan atau tidak. Berikut contoh data yang akan digunakan untuk perhitungan *backpropagation*.

Tabel 3.9 Contoh data ekstraksi fitur

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

Contoh data pelatihan data ini dapat dilihat arsitektur *backpropagation* pada Gambar 3.3



Gambar 3.7 Arsitektur *Backpropagation*

Arsitektur *backpropagation* yang digunakan untuk contoh dapat dilihat pada Gambar 3.7 *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 terdapat 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* = 10.000
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 3.10

Tabel 3.10 Bias dan bobot awal dari *input* layer ke *hidden layer* pertama

Dari- / Ke-	Bias (B1)	X1	X2	X3
Z1	0.0041	0.0400	0.1200	0.0100
Z2	0.0101	0.0080	0.1120	0.0099
Z3	0.1002	0.0230	0.0171	0.0078
Z4	0.0010	0.0011	0.0249	0.0224
Z5	0.0098	0.0078	0.0255	0.0042
Z6	0.0790	0.0143	0.0002	0.0196

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 3.11 Bias dan bobot awal dari *hidden layer* pertama ke *output layer* kedua

Dari- / Ke-	Bias (B2)	Z1	Z2	Z3	Z4	Z5	Z6
Z7	0.0153	0.0172	0.1020	0.0259	0.0130	0.0191	0.0307
Z8	0.0034	0.0233	0.1030	0.0205	0.1001	0.0026	0.0296
Z9	0.1044	0.0215	0.0384	0.0020	0.0104	0.0013	0.0154

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

Tabel 3.12 Bias dan bobot awal dari *hidden layer* kedua ke *output layer*

Dari- / Ke-	Bias (B3)	Z7	Z8	Z9
Y1	0.0850	0.0123	0.0089	0.0711

Langkah 1: Jika kondisi penghentian belum terpenuhi, lakukan langkah 2 sampai 9. Kondisi penghentian terpenuhi jika *error* < 0.0001 atau *epoch* > 1000.

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 $X_1 = 0$, $X_2 = 1$ dan $X_3=0$.

Langkah 4 : Hitung semua keluaran pada lapisan tersembunyi (Z_j, j = 1, 2,…, p) menggunakan persamaan (2-9). Berikut contoh perhitungannya :

$Z\_net_1 = 0.0041 + (0* 0.0400) + (1* 0.1200) + (0 * 0.0100)$

$= 0.1241$

$Z\_net_2 = 0.0101 + (0* 0.0080) + (1* 0.1120) + (0 * 0.0099)$

$= 0.1221$

$Z\_net_3 = 0.1002 + (0* 0.0230) + (1 * 0.0171) + (0 * 0.0078)$

$= 0.1173$

$Z\_net_4 = 0.0010 + (0* 0.0011) + (1 * 0.0249) + (0 * 0.0224)$

$= 0.0259$

$Z\_net_5 = 0.0098 + (0 * 0.0078) + (1 * 0.0255) + (0 * 0.0042)$

$= 0.0353$

$Z\_net_6 = 0.0079 + (0 * 0.0143) + (1 * 0.0002) + (0 * 0.0196)$

$= 0.0792$

Gunakan fungsi aktivasi untuk menghitung sinyal *output*-nya dengan persamaan (2-10)

$Z_1 = \frac{1}{1+e^{-0.1241}} = 0.530$

$Z_2 = \frac{1}{1+e^{-0.1221}} = 0.5305$

$Z_3 = \frac{1}{1+e^{-0.1173}} = 0.5293$

$Z_4 = \frac{1}{1+e^{-0.0259}} = 0.5065$

$Z_5 = \frac{1}{1+e^{-0.0353}} = 0.5088$

$Z_6 = \frac{1}{1+e^{-0.0792}} = 0.5198$

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

$Z\_net_7 = 0.0153 + (0.530* 0.0172) + (0.5305* 0.1020) + (0.5293* 0.0259) + (0.5065* 0.0130)$

$+ (0.5088* 0.0191) + (0.5198* 0.0307)$

$= 0.1245$

$Z\_net_8 = 0.0034 + (0.530* 0.0233) + (0.5305* 0.1030) + (0.5293*0.0205) + (0.5065* 0.1001)$

$+ (0.5088* 0.0026) + (0.5198* 0.0296)$

$= 0.1486$

$Z\_net_9 = 0.1044 + (0.530* 0.0215) + (0.5305* 0.0384) + (0.5293* 0.0020) + (0.5065* 0.0104)$

$+ (0.5088* 0.0013) + (0.5198* 0.0154)$

$= 0.1511$

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

$Z_7 = \frac{1}{1+e^{-0.1245}} = 0.5311$

$Z_8 = \frac{1}{1+e^{-0.1486}} = 0.5371$

$Z_9 = \frac{1}{1+e^{-0.1511}} = 0.5377$

Langkah ini dilakukan sebanyak jumlah lapisan tersembunyi.

Langkah 5 : Hitung semua keluaran jaringan di lapisan *output* (Y_k, k = 1, 2,…, m) menggunakan persamaan (2-11)

Y_net_1 = 0.0850+ (0.5311*0.1245) + (0.5371* 0.1486) + (0.5377* 0.1511)

= 0.3122

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

$Y_1 = \frac{1}{1+e^{-0..3122}} = 1.7318$

Fase II: *Backpropagation*

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

$\delta_1 = (t_1 - y_1)\ f'(y\_net_1)$

$= (1 - 1.7318)\ f'\ (0.3122)$

= -1.2672

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 (2-14), berikut perhitungannya :

ΔW_{71} = 0.01 * -1.2672* 0.5311= -0.0067

ΔW_{81} = 0.01 * -1.2672* 0.5371= -0.0068

ΔW_{91} = 0.01 * -1.2672* 0.5377= -0.0068

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai w_{0k}) menggunakan persamaan (2-15) di bawah ini, berikut perhitungannya :

ΔW_{01} = 0.01 * -1.2672= -0.0126

Langkah 7: Hitung faktor δ unit tersembunyi berdasarkan kesalahan di setiap unit tersembunyi (z_j, j = 1, 2,…, p) menggunakan persamaan (2-16), berikut perhitungannya :

δ_net_7 = -1.2672*0.1245 = -0.1577

δ_net_8 = -1.2672*0.1486 = -0.1884

δ_net_9= -1.2672*0.1511 = -0.1915

Faktor δ unit tersembunyi pada *hidden layer* pertama dihitung menggunakan persamaan (2-17), berikut perhitungannya :

δ_7 = -0.1577* f' (0.1245)

= -0.0347

δ_8 = -0.1884* f' (0.1486)

= -0.0484

δ_9 = -0.1915* f' (0.1511)

= -0.0499

Kemudian hitung koreksi bobot (yang nantinya akan digunakan untuk memperbaiki nilai v_{ij}) dari *input layer* ke *hidden layer* pertama menggunakan persamaan (2-18), berikut perhitungannya :

ΔV_{17} = 0.01 * -0.0347* 0.531 = -0.000184

ΔV_{27} = 0.01 * -0.0347* 0.5305 = -0.000184

ΔV_{37} = 0.01 * -0.0347* 0.5293 = -0.000184

ΔV_{47} = 0.01 * -0.0347* 0.5065 = -0.000176

ΔV_{57} = 0.01 * -0.0347* 0.5088 = -0.000177

ΔV_{67} = 0.01 * -0.0347* 0.5198 = -0.000181

ΔV_{18} = 0.01 * -0.0484* 0.531 = -0.00026

ΔV_{28} = 0.01 * -0.0484* 0.5305 = -0.00026

ΔV_{38} = 0.01 * -0.0484* 0.5293 = -0.00026

ΔV_{48} = 0.01 * -0.0484* 0.5065 = -0.00025

ΔV_{58} = 0.01 * -0.0484* 0.5088 = -0.00025

ΔV_{68} = 0.01 * -0.0484* 0.5198 = -0.02423

ΔV_{19} = 0.01 * -0.0499* 0.531 = -0.00027

ΔV_{29} = 0.01 * -0.0499* 0.5305 = -0.00027

ΔV_{39} = 0.01 * -0.0499* 0.5293 = -0.00026

ΔV_{49} = 0.01 * -0.0499* 0.5065 = -0.00025

ΔV_{59} = 0.01 * -0.0499 * 0.5088 = -0.00025

ΔV_{69} = 0.01 * -0.0499*0.5088 = -0.00026

Kemudian hitung juga koreksi bias (yang nantinya akan digunakan untuk memperbaiki nilai v_{0j}) menggunakan persamaan (2-19), berikut perhitungannya :

ΔV_{07} = 0.01 * 0.0347

= -0.0003478

$\Delta V0_8$ = 0.01 * 0.0484

= -0.0004846

$\Delta V0_9 = 0.01 * 0.099$

$= -0.0004999$

Kemudian, hitung faktor δ unit tersembunyi *hidden layer* pertama berdasarkan kesalahan di setiap unit tersembunyi (Z_j, $j = 1, 2, \ldots, p$) menggunakan persamaan (2-16)

δ_net_1 = (-0.0003478* 0.0172) + (-0.0004846* 0.0233) + (-0.0004999* 0.0215)

= -0.000028

δ_net_2 = (-0.0003478* 0.1020) + (-0.0004846* 0.1030) + (-0.0004999* 0.0384)

= -0.000105

δ_net_3 = (-0.0003478* 0.0259) + (-0.0004846* 0.0205) + (-0.0004999* 0.0020)

= -0.000020

δ_net_4 = (-0.0003478* 0.0130) + (-0.0004846* 0.1001) + (-0.0004999* 0.0104)

= -0.000058

δ_net_5 = (-0.0003478* 0.0191) + (-0.0004846* 0.0026) + (-0.0004999* 0.0013)

= -0.000009

δ_net_6 = (-0.0003478* 0.0307) + (-0.0004846* 0.0296) + (-0.0004999* 0.0154)

= -0.000033

Faktor δ unit tersembunyi *hidden layer* pertama dihitung menggunakan persamaan (2-17)

$\delta_1 = \delta\_net_1 f'(Z\_net_1)$

= -0.000028* f' (0.1241)

= 0.000047

δ_2 = -0.000105* f' (0.1221)

= 0.000174

δ_3 = -0.000020* f' (0.1173)

= 0.00003

δ_4 = -0.000058* f' (0.0259)

= 0.00011

δ_5 = -0.000009* f' (0.0353)

= 0.00002

δ_6 = -0.000033* f' (0.0792)

= 0.00006

Fase III: Perubahan Bobot

Langkah 8: Tiap-tiap unit *output* (, $k = 1, 2, \ldots, m$) memperbaiki bobotnya ($j = 0, 1, 2, \ldots, p$) menggunakan persamaan (2-20), berikut perhitungannya

W_{01} $(baru)$ = 0.0850 + (-0.0126) = 0.07232

W_{71} $(baru)$ = 0.0123 + 0.0067 = 0.00559696

W_{81} $(baru)$ = 0.0089 + 0.0068 = 0.0020934

W_{91} $(baru)$ = 0.0711 + 0.0068 = 0.0642855

Tiap-tiap unit tersembunyi (Z_j, j = 1, 2, 3,..., p) memperbaiki bobotnya (j = 0, 1, 2, 3,..., n) menggunakan persamaan (2-21), berikut perhitungannya

ΔV_{01}(baru) = 0.0041 + 0.000001 = 0.004101

ΔV_{02}(baru) = 0.0101 + 0.000002 = 0.010102

ΔV_{03} (baru)= 0.1002 + 0.000000 = 0.100200

ΔV_{04}(baru) = 0.001 + 0.000001= 0.001001

ΔV_{05}(baru) = 0.0098 + 0.000000 = 0.009800

ΔV_{06}(baru) = 0.079 + 0.000001= 0.079001

Perhitungan dilakukan sampai mendapatkan niai V_{36} baru.

Langkah 9: Kondisi pelatihan berhenti jika *error* <= 0.0001 atau jumlah *epoch* mencapai 10.000.

Software Jupyterlab 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 3.13 dan Tabel 3.14.

Tabel 3.13 Bias dan bobot akhir dari *input layer* ke *hidden layer* pertama

Dari- / Ke-	Bias (B1)	X1	X2	X3
Z1	0.565	0.906	0.868	1.701
Z2	0.651	0.762	0.783	-858
Z3	-1.227	-1.337	-1.339	-2.727
Z4	-1.213	-1.338	-1.351	-2.721
Z5	-1.362	-1.456	-1.495	-2.965
Z6	2.195	2.255	2.25	4.554

Tabel 3.14 Bias dan bobot akhir dari *hidden layer* pertama ke *output layer*

Dari- / Ke-	Bias (B2)	Z1	Z2	Z3	Z4	Z5	Z6
Z7	-1.016	-1.558	1.075	1.981	1.828	2.226	-4.273
Z8	-0.0961	-1.501	1.265	1.945	1.965	2.106	-4.645
Z9	1.177	1.569	-1.153	-2.081	-2.165	-2.379	4.660

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

dan $W_{91,}$ yaitu:

Tabel 3.15 Bias dan bobot awal dari *hidden layer* kedua ke *output layer*

Dari- / Ke-	Bias (B3)	Z7	Z8	Z9
Y1	-3.314	9.289	10.034	-12.503

Dari hasil yang sudah di dapatkan sebelumnya, nilai akan dibulatkan ke *integer* terdekat (*treshold*) dan menghasilkan tabel 3.16.

Tabel 3.16 *Output* data latih

Data ke-	***Output***	**Hasil *threshold***	**Target**
1	0.9999999000	1	1
2	0.0000139900	0	0
3	0.9999000000	1	1
4	0.0000193840	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 %.

3.4 Teknik Pengujian Sistem

Tahapan pengujian sistem digunakan untuk mengetahui dan mencoba sistem apakah berjalan baik serta untuk mengetahui kekurangan sistem pada saat terjadi kesalahan. Berikut perhitungan akurasi, presisi dan recall dalam pengujian sistem :

$$Akurasi = \frac{jumlah\ data\ sesuai\ target}{Total\ keseluruhan\ data} \quad (2\text{-}22)$$

$$Presisi = \frac{jumlah\ data\ yang\ sesuai\ target\ di\ satu\ kelas}{jumlah\ seluruh\ data\ yang\ sesuai\ target} \quad (2\text{-}23)$$

$$Recall = \frac{jumlah\ data\ yang\ sesuai\ target\ di\ satu\ kelas}{jumlah\ data\ di\ satu\ kelas} \quad (2\text{-}24)$$

Pada tabel 3.17 merupakan data *dummy* sebagai contoh untuk perhitungan pengujian sistem.

Tabel 3.17 Data *dummy*

Class	Hasil prediksi (*actual 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

Untuk menghitung akurasinya, sebagai berikut :

$$Akurasi = \frac{10+12+7+9+0}{60} \text{ x } 100\%$$
$$= 63\%$$

Untuk menentukan presisi dapat dihitun menggunakan nilai yang sesuai target pada suatu kelas di bagi jumlah semua kelas, berikut perhitungannya :

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

Sedangkan untuk perhitungan *recall* menggunakan nilai target yang sesuai tiap kelas di bagi dengan jumlah kelasnya.

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

3.4.1 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. Selain itu, berikut parameter-parameter uji yang nantinya akan digunakan dalam sistem.

1. *Neuron output*: 18 *neuron*
2. Jumlah *hidden layer* : 1, 2 atau 3 (ditentukan pada proses *trial* dan *error*).
3. Fungsi aktivasi yang digunakan adalah sigmoid biner
4. *Learning rate* sebagai parameter uji : 0.1-0.5
5. Batas *epoch* sebagai parameter uji : 1000
6. Batas *error* sebagai parameter uji : 0.01
7. Pengujian terhadap citra hasil ekstraksi fitur PCA yang diklasifikasikan dengan *backpropagation* dan uji coba terhadap citra tanpa ekstrasi fitur PCA yang langsung diklasifikasikan menggunakan *backpropagation*.
8. Pemilihan *eigen value* dan *eigen vecktor* pada PCA untuk hasil ekstraksi sebagai parameter uji sesuai dengan *trial* dan *error*.

9. Pembagian data dilakukan untuk menentukan data latih dan data uji menggunakan *K-Fold Cross Validation*. Proses akan dilakukan sebanyak k=10, yang dimana sampel yang digunakan sebanyak 13.500 data jadi setiap *fold* terdiri dari 1350 data karakter aksara. Tabel 3.18 berikut menunjukkan tahapan pengujian dengan menggunakan *K-Fold Cross Validation*.

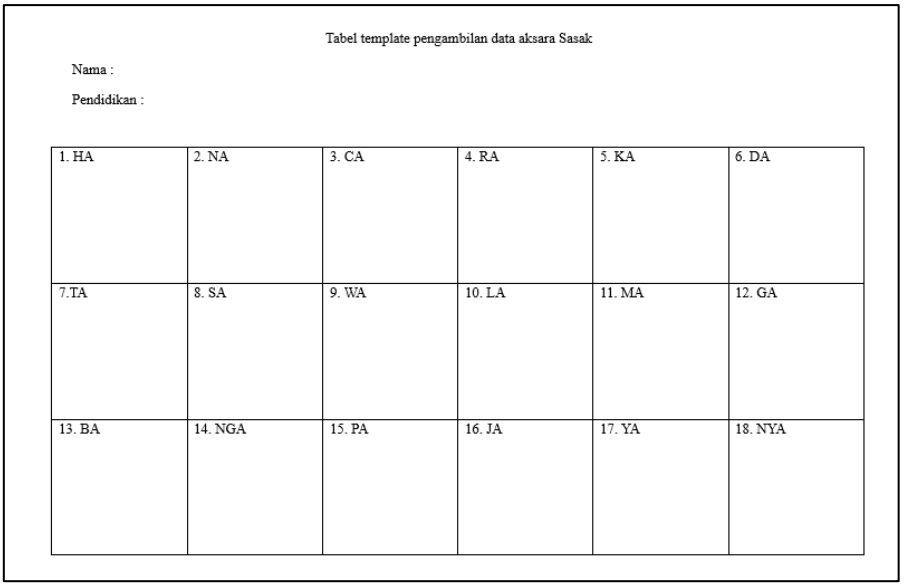
Tabel 3.18 Tahap pengujian *k-fold*

f1 *testing*	f2 training	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *testing*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *testing*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *testing*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *testing*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *testing*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *testing*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *testing*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *testing*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *testing*

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengumpulan Dataset

Pada penelitian ini, dataset yang digunakan yaitu dari dua sumber berbeda. Dataset pertama diperolah dari hasil pengambilan data secara pribadi menggunakan HVS yang sudah berisi *template* dalam bentuk tabel yang sudah ditentukan seperti pada Gambar 4.1.



Tabel template pengambilan data aksara Sasak

Nama :

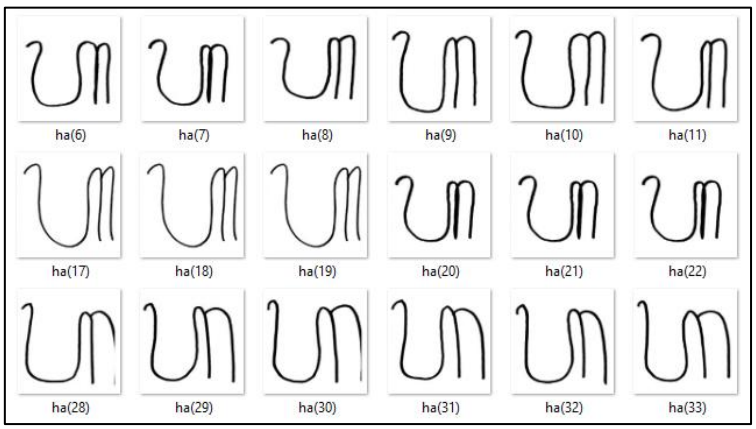
Pendidikan :

1. HA	2. NA	3. CA	4. RA	5. KA	6. DA
7.TA	8. SA	9. WA	10. LA	11. MA	12. GA
13. BA	14. NGA	15. PA	16. JA	17. YA	18. NYA

Gambar 4.1 Template pengambilan data

Data pertama bersumber dari kategori pendidikan yang berbeda-beda yaitu dari jenjang SD, SMP, SMA dan Universitas. Masing-masing per-jenjang pendidikan dicari sebanyak 10 orang untuk menulis 18 karakter aksara Sasak sebanyak 15 kali di HVS yang berbeda. Maka didapat dataset sebanyak 10.800 data yang diperoleh dari hasil perhitungan dataset yaitu

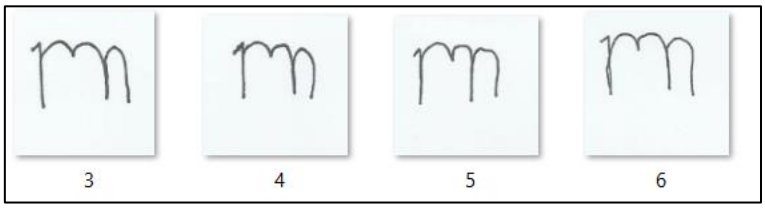
$$Dataset\ pertama$$
$$= 4\ (SD, SMP, SMA, Universitas)\ \times\ 10\ (orang)\ \times 18(karakter\ aksara)\ \times 15$$
$$= 10.800\ data$$



Gambar 4.2 Dataset pertama

Sedangkan dataset kedua diperoleh dari peneliti sebelumnya [4] sebanyak 2700. Data tersebut diperoleh dari 15 orang dari perguruan tinggi (Universitas), yang dimana masing-masing orang menulis 18 karakter aksara sebanyak 10 kali. Maka diperoleh perhitungan

$$Dataset\ kedua = 15\ (orang\ dari\ Universitas)\ \times 18(karakter\ aksara)\ \times 10 = 2700\ data$$



Gambar 4.3 Dataset kedua

4.2 Mekanisme Pengujian

Penelitian ini melakukan pengujian terhadap tiga dataset yang memiliki jumlah berbeda-beda, yaitu pada *dataset* pertama memiliki jumlah dataset sebanyak 10.800, yang kedua sebanyak 2700 dataset yang diperoleh dari penelitian sebelumnya dan 13.500 *dataset* yang didapat dari gabungan 10.800 dan 2700. Tahapan pertama yang dilakukan yaitu dengan mencari model terbaik pada ekstraksi fitur dan klasifikasi. Kemudian model terbaik yang didapatkan akan diuji ke masing masing dataset agar dapat membandingakan dan melihat pengaruh dari *dataset* tersebut yang memiliki perbedaan dalam pengambilan datanya. Perbedaannya yaitu pada pengambilan data menggunakan kertas, penulisan dengan template yang berbeda, jenis *scanner* dan resolusi *scanner* yang digunakan. Pada penelitian ini untuk 10.800 data diambil menggunakan HVS menggunakan template yang sudah disediakan lalu di *scan* menggunakan *Canon MP* 287 dengan resolusi 600 dpi, sedangkan untuk data 2700 pada penelitian sebelumnya[4] menggunakan *scanner Canon LiDE* 120 dengan resolusi 2400 dpi.

Oleh karena itu dilakukan pengujian terhadap model JST yang dibangun menggunakan dataset sebanyak 10.800 menggunakan parameter seperti berikut :

1. Ukuran citra masukan yaitu 32x32, 64x64 dan 128x128.
2. Jumlah koefisien DCT yang digunakan yaitu 64, 81, 100, 121, 144, 169, 196 dan 256.
3. Jumlah *eigen value* pada proses ekstraksi ciri PCA yang digunakan yaitu 64, 128, 192 dan 256.
4. Klasifikasi *backpropagataion*
   - Pada proses *training*, data yang digunakan sebanyak 70 persen dari dataset sedangkan 30 persen untuk tahapan *testing* (pengujian).
   - Jumlah *hidden layer neural network* yang digunakan yaitu 1 *hidden layer*, 2 *hidden layer* dan 3 *hidden layer*.

- Ukuran *neuron hidden layer* yaitu 32, 64 atau 128. Jumlah *neuron* di masing-masing *hidden layer* memiliki nilai yang sama.
- *Learning rate* 0.001 – 0.005.
- Menggunakan epoch 1000 dengan batch 100.

Selanjutnya model terbaik yang sudah didapat akan diujikan ke dataset pada penelitian sebelumnya yaitu 2700 dataset dan dataset gabungan yaitu 13.500 dan terakhir pengujian menggunakan *K-fold cross validation* untuk mengetahui peforma dari model yang sudah didapatkan pada proses *training* sebelumnya. Pengujian dilakukan menggunakan teknik *k-fold cross validation*. *K-fold cross validation* membagi seluruh *dataset* ke dalam 10 folder berbeda. Masing-masing folder memilki jumlah data yang sama. *Cross validation* mengatur folder mana yang akan dijadikan sebagai data *testing* dan *training* sesuai Tabel 4.1.

Tabel 4.1 *K-fold Cross Valdtaion*

f1 *testing*	f2 training	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *testing*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *testing*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *testing*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *testing*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *testing*	f7 *training*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *testing*	f8 *training*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *testing*	f9 *training*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *testing*	f10 *training*
f1 *training*	f2 *training*	f3 *training*	f4 *training*	f5 *training*	f6 *training*	f7 *training*	f8 *training*	f9 *training*	f10 *testing*

4.3 Pengujian Ekstraksi Fitur PCA

Pengujian ini dilakukan untuk mengetahui pengaruh penggunakan ekstraksi fitur PCA. Apakah PCA memiliki peforma yang bagus dalam mengekstrak ciri pada citra yang digabungkan dengan klasifikasi *backpropagation*. Sebelum masuk ke PCA terjadi penambahan metode DCT. Hal ini disebabkan hasil fitur di PCA menghasilkan matriks singular atau matriks yang tidak memiliki nilai determinan. Citra aksara yang digunakan memiliki dominan *background* berwarna putih dari pada hitam (tulisan aksara) sehingga jika nilai *mean* dihitung maka hasilnya *mean* yang didapat bernilai satu dan membuat determinan bernilai 0. Dengan adanya penambahan metode DCT dapat mengatasi matriks singular dan mengumpulkan

informasi penting pada citra tersebut. Berikut hasil pengujian menggunakan 3 dataset berbeda.

4.3.1 Pengaruh DCT Terhadap Akurasi

Pada pengujian ini, dilakukan pengujian terhadap 10800 *dataset* dengan mencoba variasi jumlah koefisien DCT terhadap *size* citra. Jumlah koefisien yang digunakan yaitu rentang jumlah 64-256 dengan *size* 32 × 32, *size* 64 × 64, dan *size* 128 × 128. *Size* citra diubah pada tahapan *pre-processing*, lalu jumlah koefisien DCT ditentukan dan masuk ke tahap ekstraksi fitur menggunakan PCA. Hasil dari PCA akan diklasifikasikan menggunakan *backpropagation* dengan *learning rate* 0,001, *epoch* 1000 dan 1 *hidden layer* yang memiliki 32 *neuron*.

Hasil dari klasifikasi ini berupa akurasi yang akan menjadi acuan untuk melihat koefisien DCT mana yang memiliki peforma tinggi terhadap *size* citra. Hasil pengujian ditunjukan pada Tabel 4.2.

Tabel 4.2 Pengujian koefisien DCT terhadap *size* citra

Size / DCT	Akurasi *testing* (%)		
	32	64	128
64	86,66	86,14	86,97
81	86,54	86,91	86,79
100	85,86	85,67	84,62
121	85,49	86,11	84,66
144	84,29	85,64	85,09
169	84,16	83,27	83,39
196	83,70	83,91	82,31
256	81,97	82,46	82,00

Berdasarkan Tabel 4.2 dapat dilihat akurasi yang terbaik yang didapatkan pada ukuran citra 32 × 32 dan 128 × 128 dengan koefisien DCT 64 dengan akurasi 86.66 % dan 86,97 % sedangkan citra 64 × 64 hasil akurasi tertinggi menghasilkan 86,91% dengan koefisien 81. Sesuai dengan penelitian sebelumnya yang menggunakan DCT [24], hal ini membuktikan metode DCT dapat digunakan sebagai alat bantu ekstraksi fitur PCA untuk mengatasi matriks singular dari matriks *covariance*.

4.3.2 Pengaruh Jumlah Eigen Value

Pengujian ini bertujuan untuk mencari jumlah *eigen value* terbaik yang akan digunakan sebagai proyeksi data input pada ciri ekstraksi fitur PCA. Jumlah *eigen value* ditentukan berdasarkan nilai *eigen value* terbesar dari sejumlah eigen vektor yang dipilih melalui eigen analysis atas matriks covariance yang bersesuai dengan eigen value terbesar. Variasi *size* citra yang digunakan untuk pengujian ini yaitu *size* 32 × 32, *size* 64 × 64, *size* 128 × 128 dengan menggunakan jumlah *eigen value* kelipatan 64 dan variasi jumlah koefisien DCT dari 64 sampai 256. Kemudian diklasifikasikan menggunakan backpropagation dengan learning rate

0,001, *epoch* 1000, dengan 1 *hidden layer* yang memiliki 32 *neuron* sama seperti pengujian sebelumnya. Hasil pengaruh jumlah eigen value dapat dilihat pada Tabel 4.3.

Tabel 4.3 Pengujian jumlah *eigen value* terhadap 3 size citra berbeda dan koefisien DCT

DCT	size 32				size 64			
	eigen 64	eigen 128	eigen 192	eigen 256	eigen 64	eigen 128	eigen 192	eigen 256
64	86,66	NA	NA	NA	86,14	NA	NA	NA
81	86,54	NA	NA	NA	86,91	NA	NA	NA
100	85,86	NA	NA	NA	85,67	NA	NA	NA
121	85,49	NA	NA	NA	86,11	NA	NA	NA
144	84,29	87,77	NA	NA	85,64	87,31	NA	NA
169	84,16	85,98	NA	NA	83,27	85,64	NA	NA
196	83,70	85,27	86,82	NA	83,91	86,01	86,88	NA
256	81,97	85,33	85,40	85,95	82,46	84,90	85,95	86,48

DCT	size 128			
	eigen 64	eigen 128	eigen 192	eigen 256
64	86,97	NA	NA	NA
81	86,79	NA	NA	NA
100	84,62	NA	NA	NA
121	84,66	NA	NA	NA
144	85,09	87,80	NA	NA
169	83,39	85,40	NA	NA
196	82,31	86,41	87,16	NA
256	82,00	84,56	86,23	85,86

Jumlah *eigen value* yang mendapatkan akurasi terbaik yaitu 128 dengan jumlah koefisien DCT sebesar 144 yang menghasilkam 87,80%. Jumlah *eigen value* tersebut mampu mempresentasikan ciri dataset dengan baik sehingga menghasilkan nilai akurasi yang baik begitupula dengan jumlah koefisien DCT 144 yang mampu mengumpulkan fitur penting pada frekuensi rendah dalam citra.

4.3.3 Pengaruh Jumlah *Neuron* dan Jumlah *Hidden Layer*

Pengujian ini untuk mengetahui pengaruh jumlah *hidden layer* dan jumlah *neuron* terbaik untuk mendapatkan hasil akurasi tinggi pada tahapan *testing*. Diujikan 3 jumlah *hidden layer* yang berbeda dengan masing-masing *neuron* disetiap *hidden layer* sebanyak 32 *neuron*. Ukuran citra yang digunakan yaitu 128×128 dengan jumlah *eigen value* 128 dan jumlah koefisien DCT 144 sesuai hasil terbaik yang didapat dari pengujian sebelumnya. Sedangkan *learning rate* yang digunakan yaitu 0,001 dan *epoch* 1000 pada *backpropagation*. Hasilnya

ditujukan pada Tabel 4.4. Sesuai Tabel 4.4, akurasi tertinggi yaitu 90,58 % menggunakan 3 *hidden layer*.

Tabel 4.4 Hasil pengujian terhadap jumlah *hidden layer* dengan jumlah neuron 32

Jumlah HL (jumlah neuron)	Akurasi testing (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
1 (32)	87,80	88	88	1153.868s
2 (32,32)	90,46	91	90	1173.908s
3 (32,32,32)	90,58	91	91	1190.978s

Selanjutnya percobaan dengan jumlah *neuron* sebanyak 64 dengan 3 jumlah *hidden layer* berbeda. Hasil yang didapatkan sama seperti pengujian sebelumnya, yang dimana akurasi tertinggi yang didapat menggunakan 3 *hidden layer* sesuai dengan hasil pengujian pada Tabel 4.5.

Tabel 4.5 Hasil pengujian terhadap jumlah *hidden layer* dengan jumlah *neuron* 64

Jumlah HL (jumlah neuron)	Akurasi testing (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
1 (64)	86,88	87	87	1176.598s
2 (64,64)	92,16	92	92	1318.247s
3 (64,64,64)	92,31	92	92	1442.910s

Terakhir pengujian menggunakan jumlah *neuron* sebanyak 128, hasil akurasi tertinggi yang didapat yaitu 93,85% menggunakan 3 *hidden layer*. Hal ini membuktikan pengaruh banyaknya jumlah *hidden layer* dan *neuron* mempengaruhi akurasi *testing* pada tahapan klasifikasi. Semakin banyak jumlah *neuron* dan *hidden layer* maka akurasi yang didapatkan semakin tinggi. Hasilnya dapat dilihat pada Tabel 4.6. dan waktu komputasi juga semakin lama semakin naik jika jumlah *hidden layer* yang digunakan semakin banyak. Berdasarkan pengujian tersubut maka arsitektur terbaik yang didapat akan digunakan untuk tahapan pengujian selanjutnya.

Tabel 4.6 Hasil pengujian terhadap jumlah *hidden layer* dengan jumlah *neuron* 128

Jumlah HL (jumlah neuron)	Akurasi testing (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
1 (128)	86,11	86	86	1335.948s
2 (128,128)	93,48	94	93	1794.531s
3 (128,128,128)	93,85	94	94	2049.360s

4.3.4 Pengaruh *Learning Rate*

Pengujian *learning rate* dilakukan untuk mengetahui parameter *training* untuk mengkoreksi *weight* (bobot) pada proses pembelajaran sistem dalam mengenali target. Berdasarkan pengujian-pengujian sebelumya, arsitektur terbaik yang digunakan untuk

pengujian *learning rate* ini yaitu menggunakan 3 *hidden layer* dengan masing-masing *node* berjumlah 64. Citra yang diuji memiliki ukuran 128 × 128, koefisien DCT 144 dan dengan nilai *eigen* 128. Hasil yang didapat ditunjukkan pada Tabel 4.7.

Tabel 4.7 Hasil pengujian pengaruh *learning rate*

Learning Rate	Akurasi Testing (%)	Presisi (%)	Recall (%)	Waktu komputasi (s)
0.001	93,85	94	94	2049.360s
0.002	94,13	94	94	2001.972s
0.003	94,32	94	94	2023.755s
0.004	93,82	94	94	2009.415s
0.005	94,50	95	95	2061.048s

Berdasarkan Tabel 4.7 akurasi tertinggi yang didapat mengggunakan *learning rate* sebesar 0,005. Jika diihat dari hasil akurasinya, semakin tinggi *learning rate* maka akurasi yang didapat juga bertambah. Nilai tertinggi yang dihasilkan yaitu 94,50% dengan nilai *presisi* dan *recall* masing-masing 95%. Waktu komputasi untuk pengujian realtif sama, masih dalam rentang 2000s. *Learning rate* dengan akurasi terbaik yang didapatkan akan digunakan ke pengujian selanjutnya.

4.3.5 Pengujian Model

Pada penelitian ini pengujian model dilakukan terhadap 10.800 dataset, 2700 dataset dan 13.500 dataset yang dihasilkan dari penjumlah kedua dataset dengan sumber yang berbeda sebelumnya dengan tujuan untuk mengetahui dataset terbaik yang digunakan pada tahapan *testing*. Pengujian ini menggunakan citra dengan *size* 128 × 128, koefisien 144, jumlah *eigen value* 128, *learning rate* 0,005, dan 3 *hidden layer* dengan jumlah *neuron* dimasing-masing *hidden layer* sebanyak 128. Berikut pengujian model dengan 3 jumlah dataset berbeda.

4.3.5.1 Pengujian Terhadap 10.800 Dataset

Pada dataset 10.800 peforma pengujian sistem yang mendapatkan hasil akurasi terbaik menggunakan *size* citra 128 × 128, koefisien 144, jumlah eigen value 128, *learning rate* 0,005, dan 3 *hidden layer* dengan jumlah *neuron* dimasing-masing *hidden layer* sebanyak 128. Pengujian dilanjutkan menggunakan *cross validation* ditunjukkan pada Tabel 4.8. Penggunaan *cross validation* bertujuan untuk mengevaluasi kerja model.

Tabel 4.8 Pengujian dataset 10800 menggunakan *cross validation*

cross validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	94,50	95	95	2061,048
2	100	93,76	94	94	1743,435
3	100	93,02	94	94	1713,612

4	100	93,85	94	94	1719,778
5	100	92,68	93	93	1781,96
6	100	93,58	94	94	1749,731
7	100	94,22	95	95	1740,461
8	100	93,45	93	93	1775,42
9	100	93,42	93	93	1698,571
10	100	92,99	93	93	1724,743
Rata-rata	100	93,547	93,8	93,8	1770,875

Berdasarkan Tabel 4.8 rata-rata akurasi yang didapatkan yaitu 93,547%, presisi dan recall sebesar 93,8% dengan waktu komputasi rata-rata 1770,875s. Waktu terlama pengujian ditunjukan pada *cross validation* pertama yang mencapai 2061s.

4.3.5.2 Pengujian Terhadap 2700 Dataset

Pengujian dataset 2700 yang bersumber dari penelitian sebelumnya [4]. Dataset 2700 diuji menggunaan parameter terbaik yang sudah didapatkan sebelumnya sesuai percobaan pada 10.800 dataset yaitu menggunakan *size* citra 128×128, koefisien 144, jumlah *eigen value* 128, *learning rate* 0,005, dan 3 *hidden layer* dengan jumlah *neuron* dimasing-masing *hidden layer* sebanyak 128. Dilakukan pengujian menggunakan *cross validation*, untuk mengevaluasi model terhadap 2700 dataset. Hasil Tabel 4.9 menunjukan rata-rata pengujian menghasilkan akurasi sebesar 79,04% dengan rata-rata awaktu komputasi yaitu 351,289s. Berdasarkan tabel tersebut diketahui bahwa dataset 10.800 menghasilkan akurasi yang lebih baik dibandingkan dataset 2700, hal ini disebabkan perbedaan cara pengambilan data dan jumlah data pada proses *training* di *backpropagation*.

Tabel 4.9 Pengujian dataset 2700 menggunakan *cross validation*

cross validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	80,49	81	80	343,972
2	100	77,90	78	78	349,423
3	100	79,25	79	79	338,853
4	100	80,12	80	80	340,172
5	100	78,64	79	79	347,985
6	100	79,62	80	80	352,768
7	100	78,88	79	79	341,836
8	100	75,43	76	76	335,319
9	100	80,74	81	81	380,463
10	100	79,38	80	79	382,1
Rata-rata	100	79,04	79,3	79,1	351,289

4.3.5.3 Pengujian Terhadap 13.500 Dataset

Pengujian dataset 13.500 merupakan dataset gabungan dari 10.800 dataset dan 2700 dataset. Pengujian ini bertujuan untuk melihat pengaruh dataset 2700 jika digabungkan dengan

dataset 10800 akan menghasilkan akurasi yang lebih tinggi atau sebaliknya karena data untuk tahapan *training* bertambah banyak. Pengujian menggunakan *cross validation* juga dilakukan untuk 13.500 dataset dan hasil pengujian dapat dilihat pada Tabel 4.10.

Tabel 4.10 Pengujian dataset 13.500 menggunakan *cross validation*

cross validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	91,62	92	92	2.873,240
2	100	91,60	92	92	2.542,019
3	100	91,40	91	91	2.328,856
4	100	91,87	92	92	2.331,168
5	99,98	91,01	91	91	2413,223
6	99,29	90,88	91	91	2437,184
7	99,86	90,07	90	90	2387,921
8	100	91,77	92	92	2462,586
9	99,98	91,23	91	91	2.906,579
10	99,98	91,20	91	91	2.569,806
Rata-rata	99,909	91,265	91,3	91,3	2.525,26

Berdasarkan Tabel 4.10 rata-rata akurasi yang didapatkan yaitu 91,265% dengan waktu komputasi 2.525,26s menggunakan dataset 13.500. Dari pengujian tersebut membutikan bahwa dataset 2700 mempengaruhi akurasi yang didapat jika digabungkan dengan 10.800 dataset. Penambahkan dataset 2700 membuat akurasi menurun, namun akurasi tidak menurun secara signifikan.

4.4 Pengujian Model Tanpa Ekstraksi Fitur PCA

Pengujian ini bertujuan untuk menguji akurasi *testing* pada *backpropagation* tanpa ekstraksi fitur PCA menggunakan citra dengan *size* 128 × 128, koefisien DCT 144, *learning rate* 0,005, dan 3 *hidden layer* dengan jumlah *neuron* dimasing-masing *hidden layer* sebanyak 128. Berikut pengujian model dengan 3 jumlah dataset berbeda yaitu 10.800 dataset, 2700 dataset dan 13.500 dataset.

4.4.1 Pengujian Terhadap 10800 Dataset

Dilakukan pengujian 10.800 dataset dengan model terbaik yang sudaah didapat sebelumnya. Pengujian ini dilakukan menggunakan *cross validation*, untuk mengevaluasi model. Hasil Tabel 4.11 menunjukan rata-rata pengujian menghasilkan akurasi sebesar 93,59%. Akurasi tersebut hampir sama dengan pengujian yang menggunakan ekstraksi fitur PCA. Penggunakan jumlah koefisien DCT mempengaruhi citra sebelum masuk ke dalam tahap klasifikasi *backpropagation*, karena DCT memiliki fungsi untuk menyederhanakan citra dan merangkum informasi pada citra pada frekuensi citra yang rendah [24]. Fungsi tersebut kurang lebih sama seperti PCA yang membuang fitur yang berlebihan dan menyimpan ciri penting

melalui reduksi dimensi, oleh karena itu penggunaan PCA maupun tanpa PCA memiliki peforma yang bagus dalam pengujian.

Tabel 4.11 Pengujian dataset 10.800 menggunakan *cross validation* tanpa ekstraksi fitur PCA

cross validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	93,64	94	94	2.081.491
2	100	93,05	93	93	1935,115
3	100	93,88	94	94	1896,546
4	100	93,27	93	93	1920,568
5	100	93,64	94	94	1968,261
6	100	93,88	94	94	1.910.545
7	100	93,08	93	93	1.788.693
8	100	93,48	94	93	1.778.744
9	100	93,98	94	94	1.816.401
10	100	94,07	94	94	2107.66
Rata-rata	100	93,597	93,7	93,6	1.042.622

4.4.2 Pengujian Terhadap 2700 Dataset

Pengujian dataset 2700 diuji menggunaan model terbaik yang sudah didapatkan sebelumnya sesuai percobaan yaitu menggunakan citra dengan *size* 128×128, koefisien DCT 144, *learning rate* 0,005, dan 3 *hidden layer* dengan jumlah *neuron* dimasing-masing *hidden layer* sebanyak 128. Pengujian ini menggunakan *cross validation*, untuk mengevaluasi model terhadap 2700 dataset. Hasil Tabel 4.12 menunjukan rata-rata pengujian menghasilkan akurasi sebesar 78,87 sedangkan pada Tabel 4.9 pengujian dataset 2700 menggunakan ekstraksi fitur PCA menghasilkan 79,04%. Sama seperti penjelasan sebelumnya penggunaan jumlah koefisien DCT mempengaruhi hasil *testing* pada *backpropagation*. Penggunaan PCA dan metode DCT yang memiliki fungsi yang sama yaitu merangkum informasi penting pada citra maka hasil akurasi yang dihasilkan tidak jauh berbeda.

Sedangkan akurasi yang dihasilkan jika dibandingkan dengan dataset 10.800 tanpa ekstraksi fitur PCA memiliki akurasi lebih baik akibat jumlah dataset dan cara pengambilan dataset berbeda menyebabkan akurasi pada dataset 2700 menurun.

Tabel 4.12 Pengujian dataset 2700 menggunakan *cross validation* tanpa ekstraksi fitur PCA

cross validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	79,13	80	79	311,160
2	100	79,38	80	79	373,519
3	100	77,16	77	77	367,552
4	100	80,61	81	81	374,468
5	100	80,74	81	81	354,079

6	100	78,76	79	79	370,129
7	100	80,37	81	80	402,097
8	97,08	72,59	74	73	367,181
9	100	82,22	83	82	377,026
10	100	78,02	79	78	327,679
Rata-rata	97,08	78,87	79,5	78,9	362

4.4.3 Pengujian Terhadap 13.500 Dataset

Pengujian dataset 13.500 bertujuan untuk melihat pengaruh dataset 2700 jika ditambahkan dengan dataset 10.800. Pengujian ini dilakukan menggunakan *cross validation*, untuk mengevaluasi model. Rata-rata akurasi pengujian yang ditujukkan pada Tabel 4.13 yaitu sebesar 90,675% dengan rata-rata waktu komputasi diatas 2720,6423s. Waktu komputasi ini adalah waktu komputasi yang terlama jika dibandingkan dengan pengujian-pengujian sebelumnya. Karena disini ekstraksi fitur PCA tidak digunakan untuk mengekstrak ciri dan hasil dari DCT langsung dieksekusi ke tahapan klasifikasi *backpropagation* dan akibat jumlah data yang sangat banyak maka proses *training* akan lebih lama. Akurasi yang didapat menurun sebanyak 1% dari pengujian dataset 13.500 menggunakan esktraksi fitur PCA.

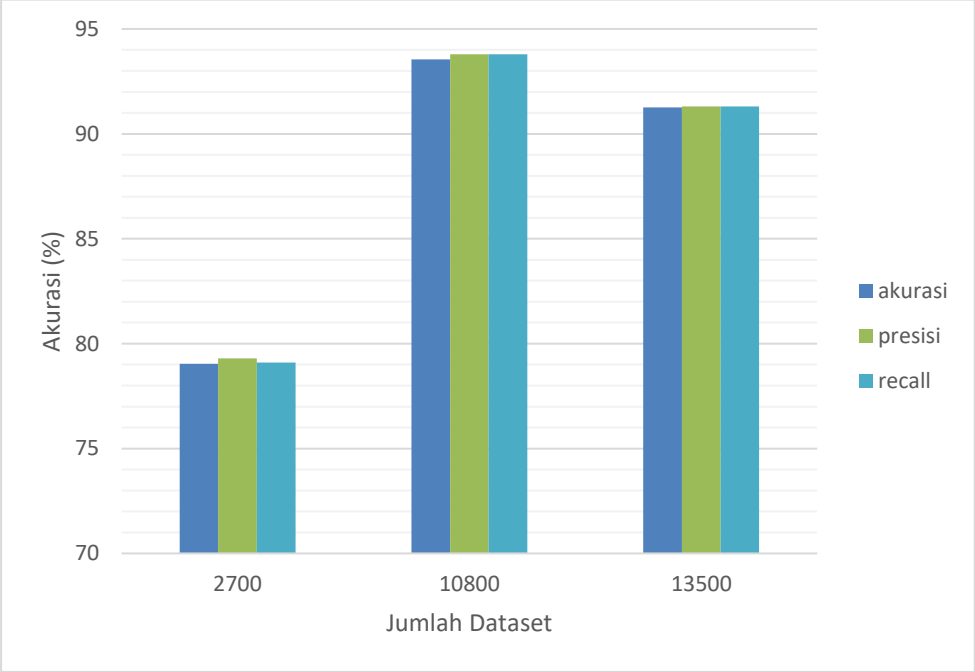
Tabel 4.13 Pengujian dataset 13.500 menggunakan *cross validation* tanpa ekstraksi fitur PCA

cross validation	Akurasi *Training* (%)	Akurasi *Testing* (%)	Presisi (%)	*Recall* (%)	Waktu komputasi (s)
1	100	91,23	91	91	2784,071
2	100	92	92	92	2673,282
3	100	91,11	91	91	2678,69
4	93,51	85,92	86	86	2671,21
5	99,98	91,11	91	91	2693,993
6	99,87	91,60	92	92	2690,717
7	99,98	90,74	91	91	2662,48
8	100	91,72	92	92	2.684,461
9	99,98	90,76	91	91	2.600,326
10	99,98	90,56	91	91	3.067,193
Rata-rata	99,33	90,675	90,8	90,8	2720,6423

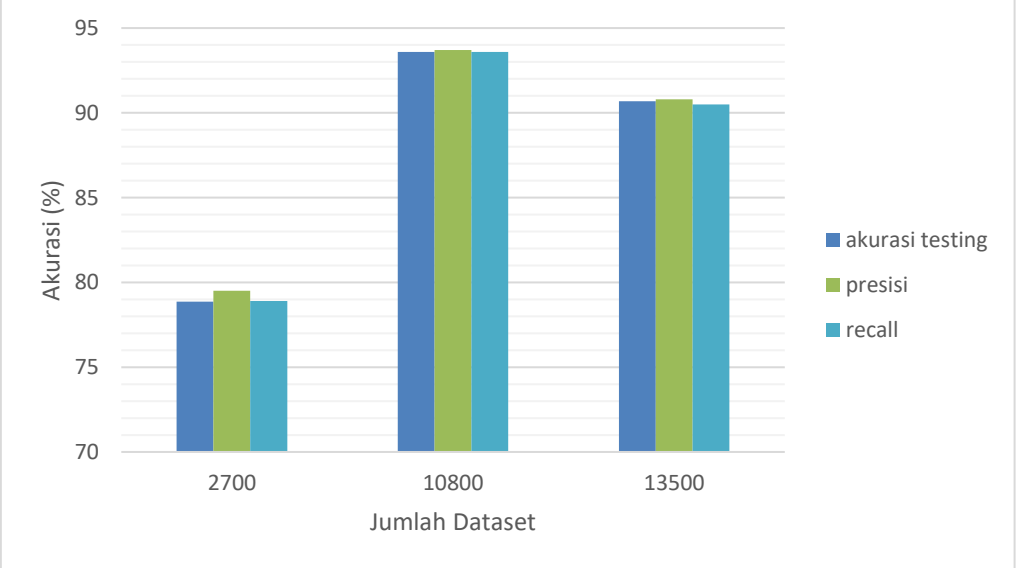
4.5 Perbandingan Grafik Ekstraksi Fitur PCA dan Tanpa PCA

Berdasarkan pengujian-pengujian sebelumnya, dapat dilihat perbedaan bahwa pengujian dengan akurasi tertinggi didapat menggunakan dataset 10.800 yang akurasinya mencapai rata-rata 93% menggunakan ekstraksi fitur PCA maupun tanpa ekstraksi fitur PCA. Sedangkan menggunakan dataset 2700 menghasilkan akurasi tertinggi 79,04% yang menggunakan ekstraksi fitur sedangkan yang tidak menggunakan ekstraksi fitur menghasilkan 78,87%.

. Kemudian dataset tersebut digabungkan menjadi 13.500 dataset dan diuji menghasilkan akurasi tertinggi 91,265% dengan ekstraksi fitur PCA dan 90,675% yang tidak menggunakan ekstraksi fitur PCA. Grafik hasil dari ekstraksi fitur PCA dan tanpa ekstraksi fitur PCA dapat dilihat pada Gambar 4.4 dan Gambar 4.5.



Gambar 4.4 Grafik batang perbandingan akurasi dari 3 dataset berbeda menggunakan ekstraksi fitur PCA



Gambar 4.5 Grafik batang perbandingan akurasi dari 3 dataset berbeda tanpa estraksi fitur PCA

Penurunan akurasi membuktikan bahwa perbedaan jumlah dataset dan pengambilan dataset mempengaruhi akurasi dari pengujian aksara Sasak. Dataset 2700 pada penelitian

sebelumnya diambil menggunakan template berbeda dengan dataset 10.800. Jika dilihat dataset 2700 memiliki warna *background* kebiruan sedangkan dataset 10.800 memiliki data dengan *background* putih. Perbedaannya dapat dilihat dari ketebalan tulisan yang dapat dilihat pada Gambar 4.6.

Gmbar 4.6 Dataset 2700 dan dataset 10800

Jumlah dataset mempengaruhi kerja model, karena dataset yang banyak memiliki variasi tulisan yang lebih beragam. Hal ini akan digunakan oleh sistem untuk mempelajari karakter-karakter tulisan yang berbeda dan proses *pre-processing* suatu citra memiliki pengaruh penting juga dalam pengambilan fitur yang ada pada suatu citra sebelum masuk ke tahapan klasifikasi. Sesuai dengan penggunakan ekstraksi fitur PCA yang mampu mengatasi dataset dalam jumlah yang banyak dan klasifikasi *backpropagation* yang didalamnya harus melalui tahapan *training* (mempelajari) data dengan menyesuaikan *weight* (bobot) dan *cost* agar mencapai target yang diinginkan. Jadi semakin banyak sistem belajar maka semakin baik sistem dalam mengenali targetnya dengan dataset yang memiliki *pre-processing* yang baik.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang sudah dilakukan, berikut kesimpulan yang bisa disimpulkan oleh penulis.

1. Model terbaik yang dihasilkan pada penelitian ini yaitu menggunakan size citra 128, jumlah koefisien DCT 144, jumlah *eigen value* 128, 3 *hidden layer* dengan masing-masing *neuron* di *hidden layer* bernilai 32 dan *learning rate* 0,005.
2. Persentasi tertinggi yang didapat pada pengujian ini yaitu sebesar 93,547% menggunkan dataset 10800 berdasarkan model dan arsitektur terbaik.
3. Hasil pengujian menggunakan ekstraksi fitur PCA dan tanpa ekstraksi fitur PCA hampir sama, yaitu dengan akurasi 93,54% menggunakan PCA dan 93,59% untuk pengujian tanpa PCA pada dataset 10.800.
4. Semakin tinggi nilai *learning rate* yang digunakan maka akurasi yang didapat semakin tinggi.
5. Pengujian terhadap 3 dataset memliki akurasi yang berbeda. Dataset yang memiliki akurasi tertinggi menggunakan dataset 10800 sedangkan akurasi terendah yang didapat sebesar 75% saat menggunakan dataset 2700. Hal ini disebabkan perbedaan cara pengambilan data yang dapat mempengaruhi akurasi. Semakin bagus dataset maka akurasi yang didapat semakin tinggi.

5.2 Saran

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

1. *Pre-processing* pada pengolahan data dapat ditambahkan.
2. Penggunaan metode klasifikasi dapat menggunakan metode yang lain seperti KNN.

DAFTAR PUSTAKA

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

[2] Jamaluddin, "Sejarah Tradisi Tulis Dalam Masyarakat Sasak Lombok" Ulumuna, vol IX edisi 16 no 2. 369-384, 2005.

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

[4] Ms, E. D. J. U., Wijaya, I. G. P. S., & Bimantoro, F. "Pengenalan Pola Tulisan Tangan Huruf Sasak Menggunakan Metode *Integral Projection* dan *Neural Network*".J-COSINE, Vol 3 no 1, 2019.

[5] Wijaya, N., & Susanto, K.. "Pengenalan Pola Huruf pada Kata dengan Menggunakan Algoritma *Backpropagation* dan Hybrid *Feature*". Teknomatika, Vol 09 no 02. 2019.

[6] Gulati, I., Vig, G., & Khare, V. "Real Time Handwritten Character Recognition Using ANN". IJESRT hal 357-362. 2018

[7] Wibowo, A., Indriati, R., & Wulanningrum, R. "Sistem Pengenalan Pola Motif Batik Kediri". Artikel Skripsi Universitas Nusantara PGRI Kediri. 2017.

[8] Faturrahman, I., Arini & Fitri, M. "Pengenalan Pola Huruf Hijaiyah Khat Kufi Dengan Metode Deteksi Tepi Sobel Berbasis Jaringan Syaraf Tiruan *Backpropagation*". Jurnal Teknik Informatika, vol 11 no 1. 2018.

[9] Taye, O. A.., Abdullahi.Y. M., & Ikeola, S. A. "*Recognition of Alphabet Characters and Arabic Numerals Using Backpropagation Neural*". *Jurnal of Science and Control Systems*, vol 11 no 2. 2018.

[10] Syamsudin, Zubair & Solichin, Achmad. "Pengenalan Karakter Sandi Rumput Pramuka Menggunakan Jaringan Saraf Tiruan Dengan Metode *Backpropagation*". Yogyakarta: Seminar Nasional Teknologi Informasi dan Multemedia 2017, hal 3.9. 2017.

[11] Hara, E., Fitriawan, H., & Mulyani, Y. "Penggunaan Deteksi Tepi (Canny) pada Sistem Pengenalan Tulisan Tangan Aksara Lampung Berbasis Jaringan Syaraf Tiruan". Jurnal Rekayasa dan Teknologi Elektro, vol 10, no 3. 2016.

[12] Wahyuningrum, R. T. "Pengenalan Pola Senyum Menggunakan *Backpropagation* Berbasis Ekstraksi Fitur *Principal Component Analysis* (PCA)". Rekayasa, vol 4 no 1. 2011.

[13] Nafan, M. Z., Anggoro, A. W., & Usada, E. "Identifikasi Citra Tanda Tangan Berdasarkan

Grid Entropy dan PCA Menggunakan *Multi Layer Perceptron*". J.Of INISTA, 1(2) hal 89-96. 2019.

[14] Diyah, P., Dyan, K. S., & Boko, S. "Dampak Reduksi Sampel Menggunakan PCA Pada Pelatihan Jaringan Syaraf Tiruan Terawasi (Studi Kasus : Pengenalan Angka Tulisan Tangan)". Jurnal Pseudocode, vol 2 no 1. 2014.

[15] Ismawan, F. "Hasil Ekstraksi Algoritma Principal Component Analysis (PCA) untuk Pengenalan Wajah dengan Bahasa Pemograman Java Eclipse IDE". Jurnal Sisfotek global vol 5 no 1. 2015.

[16] Haumahu, J. P."Implementasi Jaringan Syaraf Tiruan Untuk Pengenalan Pola Notasi Balok Menggunakan Metode *Backpropagation.*" JURIKOM (Jurnal Riset Komputer) vol 6 no 3: 255-259. 2019.

[17] D. R. Taningrum, B. Hidayat, And Y. S. Hariyani, "Sistem Pengidentifikasian Plat Nomor Kendaraan Mobil Menggunakan Principal Component *Analysis* Dan Klasifikasi K-NN".*E-Proceeding Eng.*, Vol. 3, No. 2, Pp. 1868–1876, 2016.

[18] I. Setiawan And W. Iskand, "Implementasi Pengenalan Citra Wajah Dengan Algoritma *Eigenface* Pada Metode *Principal Component Analysis* (PCA)," Vol. 2016, 2016.

[19] H. Jaya, dkk., "Kecerdasan Buatan". Makassar: Fakultas MIPA Universitas Negeri Makassar, 2018.

[20] R. S. Suhartanto, C. Dewi, and L. Muflikhah. "Implementasi Jaringan Syaraf Tiruan Backpropagation untuk Mendiagnosis Penyakit Kulit pada Anak," J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya, vol. 1, no. 7, pp. 555–562, 2017.

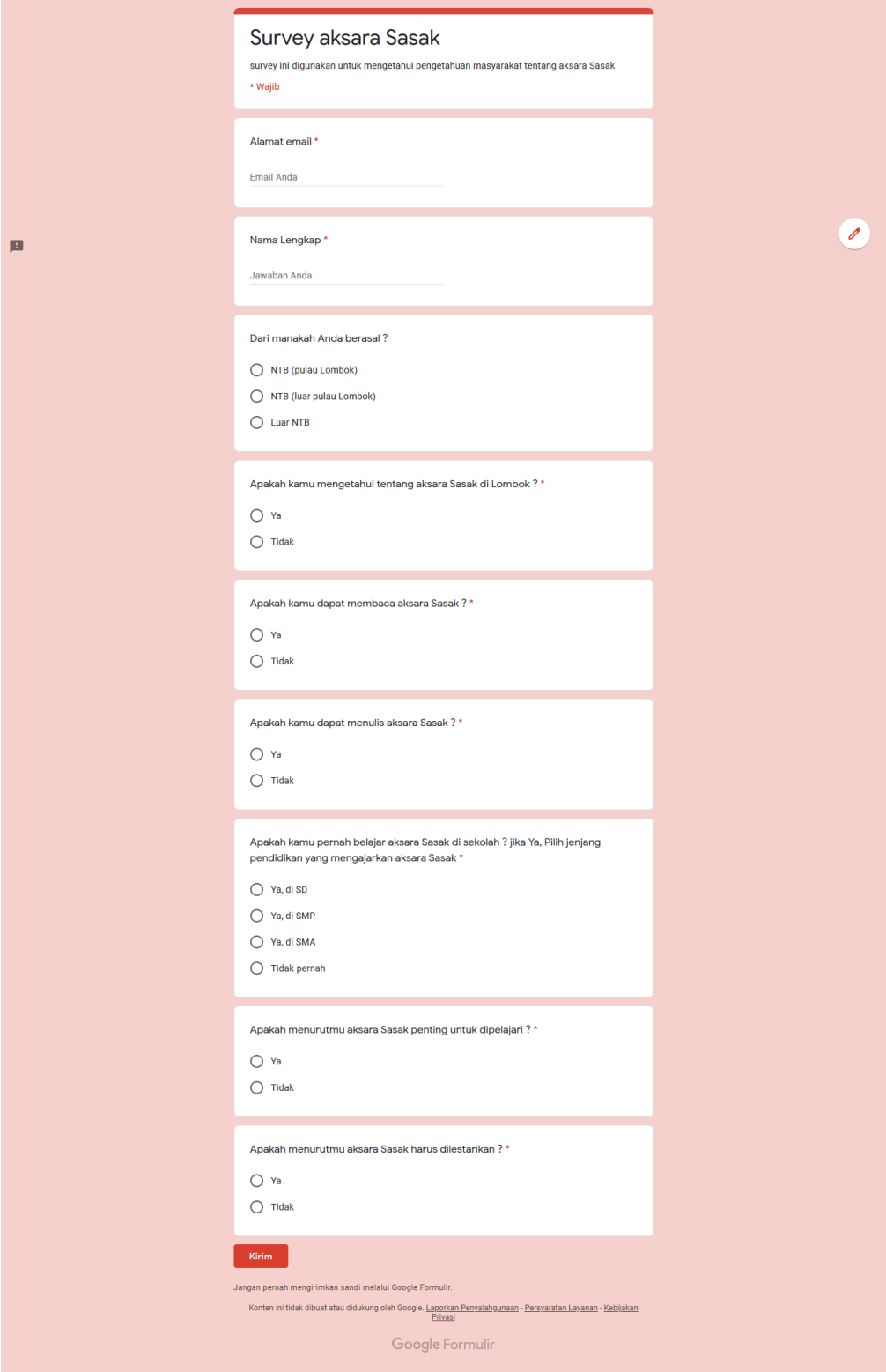
[21] A. Jumarwanto, R. Hartanto, and D. Prastiyanto. "Aplikasi Jaringan Saraf Tiruan Backpropagation Untuk Memprediksi Penyakit THT Di Rumah Sakit Mardi Rahayu Kudus," J. Tek. Elektro, vol. 1, no. 1, pp. 11–21, 2009.

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

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

[24] 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.

LAMPIRAN



Survey aksara Sasak

survey ini digunakan untuk mengetahui pengetahuan masyarakat tentang aksara Sasak

* Wajib

Alamat email *

Email Anda

Nama Lengkap *

Jawaban Anda

Dari manakah Anda berasal ?

- NTB (pulau Lombok)
- NTB (luar pulau Lombok)
- Luar NTB

Apakah kamu mengetahui tentang aksara Sasak di Lombok ? *

- Ya
- Tidak

Apakah kamu dapat membaca aksara Sasak ? *

- Ya
- Tidak

Apakah kamu dapat menulis aksara Sasak ? *

- Ya
- Tidak

Apakah kamu pernah belajar aksara Sasak di sekolah ? jika Ya, Pilih jenjang pendidikan yang mengajarkan aksara Sasak *

- Ya, di SD
- Ya, di SMP
- Ya, di SMA
- Tidak pernah

Apakah menurutmu aksara Sasak penting untuk dipelajari ? *

- Ya
- Tidak

Apakah menurutmu aksara Sasak harus dilestarikan ? *

- Ya
- Tidak

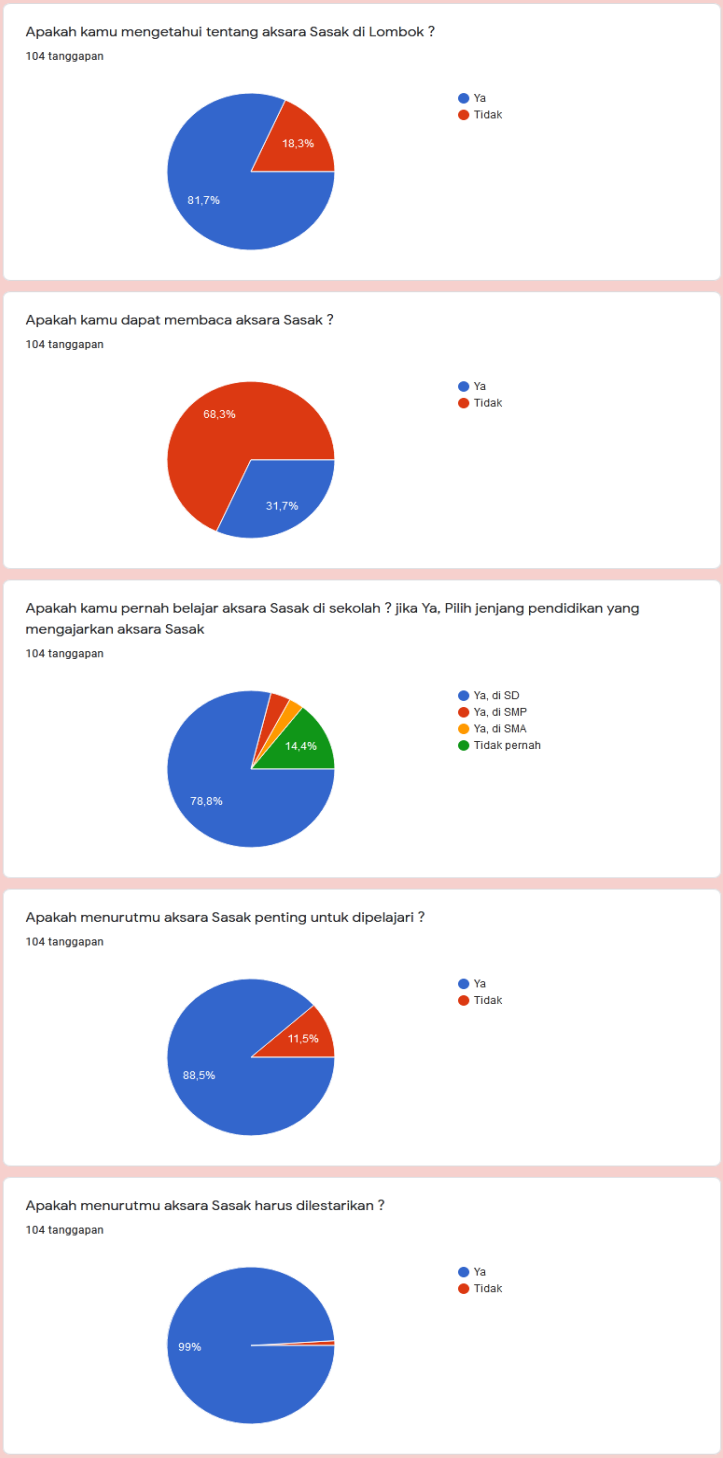
Kirim

Jangan pernah mengirimkan sandi melalui Google Formulir.

Konten ini tidak dibuat atau didukung oleh Google. Laporkan Penyalahgunaan - Persyaratan Layanan - Kebijakan Privasi

Google Formulir

Gambar survey aksara Sasak



Gambar hasil survey aksara Sasak

Tabel template pengambilan data aksara Sasak

Nama :

Pendidikan :

1. HA	2. NA	3. CA	4. RA	5. KA	6. DA
7.TA	8. SA	9. WA	10. LA	11. MA	12. GA
13. BA	14. NGA	15. PA	16. JA	17. YA	18. NYA