# BAB 4

# ANALISA DAN PERANCANGAN

Pada perancangan sistem berbasis komputer, analisa memegang peranan yang penting dalam membuat rincian sistem baru. Analisa merupakan langkah pemahaman persoalan sebelum mengambil tindakan atau keputusan penyelesaian hasil utama, sedangkan tahap perancangan sistem adalah membuat rincian hasil dari analisa menjadi bentuk perancangan agar dapat dipahami dalam menjelaskan analisanya dalam dunia nyata sehingga mendapatkan gambaran tentang analisa dan mudah dimengerti.

## 4.1 Analisa Sistem

Analisa sistem yang akan dilakukan meliputi analisa sistem lama dan analisa sistem yang akan dibuat.

### 4.1.1 Analisa Sistem Yang Berjalan Sekarang

Analisa sistem ini dilakukan untuk menentukan langkah – langkah yang akan diambil untuk mencari solusi dari permasalahan – permasalahan yang terjadi, yaitu dengan cara mengetahui gambaran sistem yang sedang berjalan.

Permasalah – permasalahan yang ditemukan pada saat analisa sistem ini dapat diatasi dengan mengadakan perubahan – perubahan, melakukan pengembangan sistem atau dengan cara merancang suatu sistem baru, dengan tujuan dapat mengatasi, memperbaiki serta menutupi kelemahan – kelemahan dari permasalahan – permasalahan yang terdapat pada sistem sebelumnya.

Sebelumnya melakukan perubahan sistem, maka perlu mengetahui dan memahami gambaran yang membuat keterangan atau informasi mengenai sistem yang sedang berjalan pada suatu organisasi yang bersangkutan, hal ini dilakukan agar dapat mempermudah dalam menganalaisa dan merancang sistem baru.

### 4.1.2 Analisa Sistem yang akan dikembangkan

Sistem yang akan dikembangkan adalah pengenalan pola akasara Batak yang terkomputerisasi, dengan penerapan metode *Convolution Neural Network* (CNN). Yang mana pada metode *Convolution Neural Network* (CNN) terdiri dari beberapa layer untuk mengenali pola aksara Batak tersebut, terdiri dari : *convolution layer, pooling layer,* dan *fully connected layer*.

Dimana pada *convolution layer* untuk mempelajari citra yang dimasukkan kedalam program kemudian dengan layer ini akan mengekstrak citra yang dimasukkan dan diteruskan ke layer selanjutnya yaitu *pooling layer* dimana pada layer ini citra yang sudah di ekstrak pada *layer* sebelumnya akan di *resizing* atau mengubah ukuran citra tersebut menggunakan operasi *max,* yang mana operasi *max* ini akan mengambil satu nilai tertinggi sehingga menghasilkan citra yang telah diperkecil. Kemudian pada *fully connected layer* akan mengambil seluruh *neuron* pada *layer* sebelumnya.

### 4.1.3 Analisa Data Masukan ( *Input* )

Dalam membangun aplikasi pengenalan pola aksara Batak menggunakan metode *Convolution Neural Network* (CNN) diperlukan data – data agar sistem dapat berjalan sesuai dengan harapan, adapun data – data yang dibutuhkan untuk

perancangan dan implementasi sistem ini yaitu data – data yang berhubungan dengan aksara batak.

### 4.1.4 Analisa Proses

Proses yang terjadi pada sistem yaitu proses penginputan hasil scan berupa gambar dengan format JPG, proses *training* dari gambar yang di *input* kan kedalam program, dimana pada proses *training* akan menggunakan perhitungan metode *Convolusion Neural Network* (CNN), kemudian proses ekstraksi gambar yang sudah di *input* kan ke dalam program.

### 4.1.5 Analisa Fungsi Sistem

Aplikasi yang akan dibangun memiliki fungsi :

1. *Input* Gambar

   Aplikasi ini memiliki fungsi sebagai untuk memasukkan gambar yang sudah di scan sebelumnya berformat JPG.

2. *Train*

   Aplikasi ini memiliki fungsi untuk melatih atau *training* dari gambar yang sudah dimasukkan ke dalam program

3. Ekstrak

   Aplikasi ini memiliki fungsi untuk mengekstrak hasil *train* tadi sehingga menampilkan hasil dari proses *train* tersebut.

### 4.1.6 Analisa Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dalam pengembangan dan implementasi Aplikasi Pengenalan Pola Aksara Batak menggunakan metode *Convolusion Neural Network* (CNN) adalah :

1. *Windows 7*, sebagai sistem operasi yang digunakan
2. MATLAB, untuk pembuatan pembuatan dan penulisan *coding* program.

## 4.2 Analisa Kebutuhan Sistem

Dalam membangun suatu sistem data – data agar sistem dapat berjalan dengan baik maka dibutuhkan beberapa proses.

### 4.2.1 Analisa Masukan Sistem

a. Proses *Input* Gambar

   Proses *input* gambar memiliki fungsi untuk memasukkan gambar yang akan di proses dengan metode *Convolutional Neural Network* (CNN).

b. Proses Ekstrak

   Data ekstrak memiliki fungsi untuk mengekstrak citra yang di inputkan

c. Proses *Train*

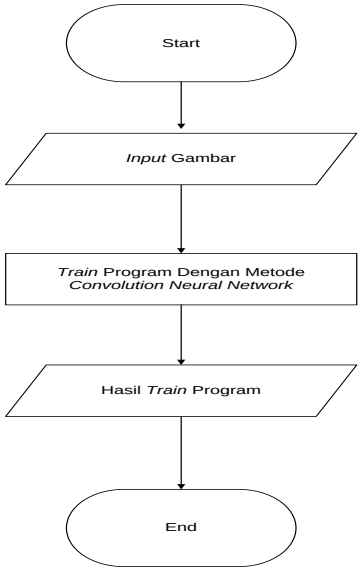
   Proses *train* memiliki fungsi untuk melakukan perhitungan pada citra sehingga hasil perhitungan nya bisa di proses.

### 4.2.2 Analisa Keluaran ( *Output* ) Sistem

Keluaran (*Output*) yang dihasilkan adalah sebuah model baru dengan kemampuan mengenali tulisan tangan huruf aksara Batak .

## 4.3 *Flowchart*

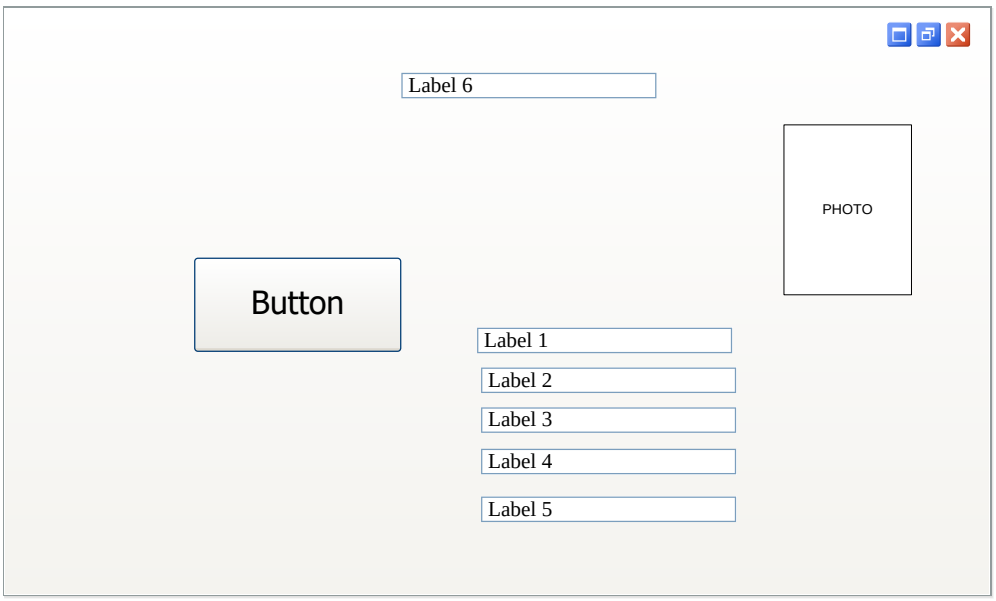
*Flowchart* sistem dapat dijelaskan proses berjalannya aplikasi dimulai dengan melakukan peng*inputan* citra atau gambar berformat jpg, kemudian citra atau gambar tersebut di *train* dengan metode *Convolutional Neural Network* (CNN) untuk pengenalan pola aksara Batak :




**Gambar 4.1 *Flowchart* Proses Pengenalan Pola aksara batak**

## 4.4 Halaman Utama

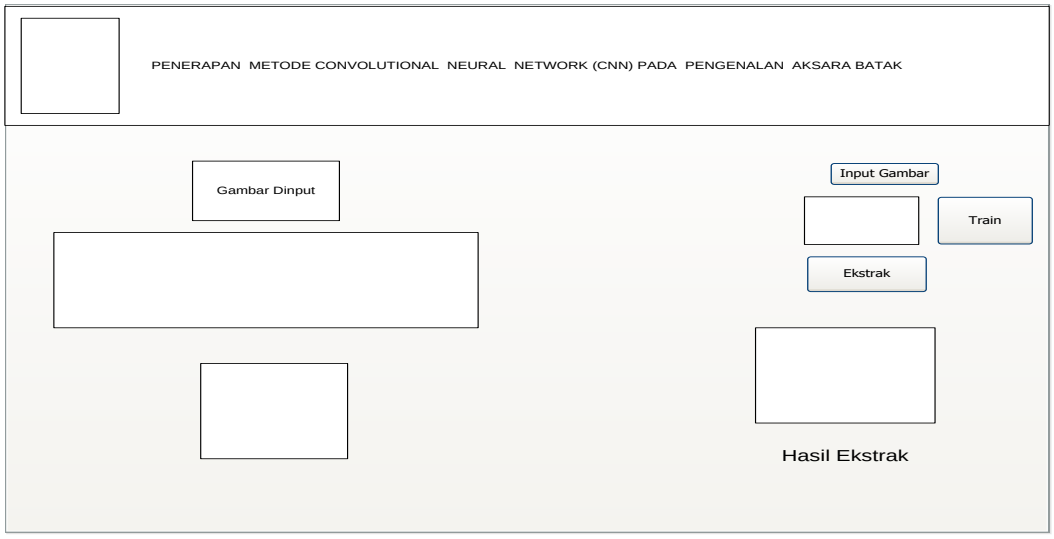
Halaman utama adalah halaman yang muncul saat aplikasi pertama dijalankan, baik dalam sistem pengambilan data training maupun dalam sistem pengenalan kata. Rancangan tampilan halaman utama dapat dilihat sebagai berikut.




**Gambar 4.2 Halaman Utama Pengenalan Pola Aksara Batak**

## 4.5 Tampilan Utama

Tampilan utama adalah halam tempat menjalankan fungsi untuk mengolah citra yang di*input* kedalam program

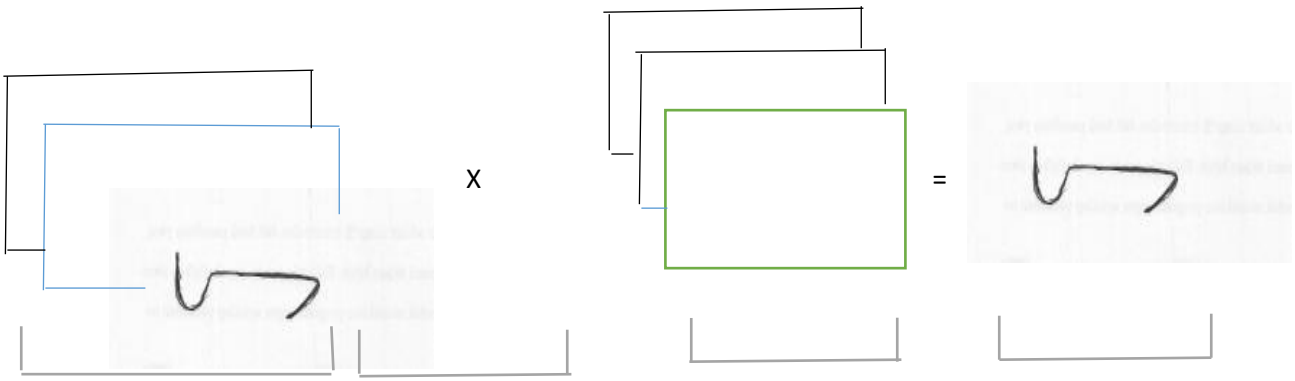



**Gambar 4.3 Tampilan Utama Pengenalan Pola Aksara Batak**

Keterangan :

1. *Input* gambar adalah untuk memasukkan gambar kedalam program dalam bentuk JPG.
2. *Train* adalah proses untuk melatih citra yang telah di*inputkan*
3. *Ekstrak* adalah proses dimana untuk mengekstrak hasil citra yang sudah di *train* sebelumnya.

## 4.6 Contoh Perhitungan Kasus




*input image* *Convolution* *kernel* 3x3 *output*

contohnya yaitu peneliti menginputkan gambar dengan ukuran 64x64x3, yang mana tinggi dan lebar dari gambar yang diinput tersebut adalah 64 dan gambar tersebut memiliki 3 *channel* yaitu *red,green, blue* atau yang biasa disebut dengan RGB setiap *channel* tersebut memiliki matriks yang berbeda – beda. Pada contoh ini peneliti di uji sebanyak 60 kali. Setelah di *input* gambar tersebut maka akan masuk pada tahan *convolution* yang mana pada konvolusi ini memiliki *filter* yang sudah ditentukan. *Filter* digunakan untuk menentukan pola apa yang dideteksi yang selanjutnya akan dikalikan dengan matriks dengan nilai matriks yang di *input*.

Jumlah *filter* pada konvolusi ini sebanyak 64 piksel dengan ukuran kernel 3x3. Yang mana akan dihasilkan gambar sebanyak 64 *fiture map*.

Agar dapat dipahami, peneliti mengambil contoh sebagian matriks pada *input image* yang berukuran 64x64 .

| | | | | |
|---|---|---|---|---|
| 23 | 25 | 0 | 21 | 22 |
| 23 | 25 | 0 | 0 | 25 |
| 25 | 25 | 0 | 20 | 20 |
| 20 | 25 | 24 | 0 | 25 |
| 25 | 24 | 0 | 25 | 24 |

$*$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| -1 | 1 | -1 |
| 1 | -1 | 1 |

| | | |
|---|---|---|
| 125 | 65 | 38 |
| 117 | 29 | 74 |
| 80 | 93 | 40 |

**Gambar 4.4 Perhitungan Konvolusi**

Gambar diatas menunjukkan proses konvolusi dengan menggunakan ukuran kernel 3x3 dengan menggunakan *stride* 1. *Stride* disini artinya jumlah pergeseran kernel terhapad matriks *input* berjumlah satu. Untuk mendapat kan hasil dari pehitungan tersebut digunakan fungsi *dot product*, perhitungan nya dapat dilihat dibawah ini :

$$a.\ Position\ 1 = (23x1) + \big(23x(-1)\big) + (25x1) + (25x1) + \big(25x(-1)\big) + (25x1) + (0x1) + (0x1) + \big(0x(-1)\big) + (0x1) = 123$$

$$b. Position\ 2 = (25x1) + (25x(-1) + (25x1) + (0x1) + (0x1) + (0x1) + (21x1) + (0x(-1) + (20x1) = 65$$

$$c. Position\ 3 = (0x1) + (0x(-1) + (0x1) + (21x1) + (0x1) + (20x1) + (22x1) + (25x(-1) + (20x1) = 38$$

$$d. Position\ 4 = (23x1) + (25x(-1) + (20x1) + (25x1) + (25x1) + (25x1) + (0x1) + \big(0x(-1)\big) + (24x1) = 117$$

$$e.Position\ 5 = (25x1) + \big(25x(-1)\big) + (25x1) + (0x1) + (0x1) + (24x1) + (0x1) + (20x(-1) + (0x1) = 29$$

$$f.Position\ 6 = (0x1) + \big(0x(-1)\big) + (24x1) + (0x1) + (20x1) + (0x1) + (25x1) + (20x(-1) + (25x1) = \ 74$$

$$g.Position\ 7 = (25x1) + (20x(-1)) + (25x1) + (25x1) + (25x1) + (24x1) + (0x1) + (24x(-1)) + (0x1) = 80$$

$$h.Position\ 8 = (25x1) + (25x(-1) + (24x1) + (0x1) + (24x1) + (0x1) + (20x1) + \big(0x(-1)\big) + (25x1) = \ 93$$

$$i.Position\ 9 = (0x1) + (24x(-1) + (0x1) + (20x1) + (0x1) + (25x1) + (20x1) + (25x(-1) + (24x1) = 40$$

Setelah hasil perhitungan dengan konvolusi dapat, untuk menghilangkan nilai negative pada hasil, maka digunakan fungsi aktivasi ReLU ( *Rectified Linear Unit*) setelah proses konvolusi. Selanjutnya masuk kedalam proses *Pooling layers* dimana pada proses ini menggunakan fungsi *max pooling* yang mana dengan cara mengambil nilai maksimum matriks dari hasil konvolusi berikut gambaran proses *pooling layer* :

| | | |
|---|---|---|
| 125 | 65 | 38 |
| 117 | 29 | 74 |
| 80 | 93 | 40 |

=

| | |
|---|---|
| 125 | 74 |
| 117 | 93 |

**Gambar 4.5 Proses *Pooling layers***

Proses *Pooling* ini menggunakan ukuran 2x2 dengan *strade* 1 dimana nilai setiap pegeseran setiap kernel matriks berjumlah satu. Untuk proses selanjutnya yaitu proses *fully connected* yang mana pada proses ini nilai *input* martiks dari layer

sebelumnya akan diubah menjadi *vector*. Dalam proses ini aktivasi fungsi *softmax*. Aktivasi ini akan membantu MLP untuk mengenal pola tersebut.

Setelah didapat hasil dari *fully connected layers* untuk menghitung nilai akurasi dari keseluruhan matriks adalah dengan menggunakan rumus berikut :

$$\frac{409}{1} = 4.09\ \% overall\ Accuracy = \frac{TTP\ all}{Tota\ Number\ of\ Testing\ Entries}\ x\ 100\ \%$$

Jadi akurasi yang dihasilkan oleh model dengan *input* gambar 64x64 piksel dengan jumlah *testing* sebanyak 1 kali didapatkan nilai akurasi sebesar 4.09%.

# BAB 5
# IMPLEMENTASI DAN PENGUJIAN

Implementasi dan pengujian merupakan tahap yang akan dilakukan setelah tahap analisa dan perancangan selesai dikerjakan.

## 5.1 Implementasi

Implementasi merupakan tahap kelanjutan dari tahap perancangan sistem yang telah didesain. Implementasi merupakan tahap pembangunan sistem menggunakan perangkat keras dan perangkat lunak yang telah ditetapkan.

Tujuan implementasi antara lain :

1. Menyelesaikan desain sistem yang ada dalam dokumen perancangan yang telah disetujui.
2. Menguji dan mendokumentasikan program – program atau prosedur – prosedur dari dokumen perancangan sistem yang telah disetujui
3. Memastikan bahwa pemakai dapat mengoperasikan sistem yakni dengan mempersiapkan secara manual pemakai serta melatih pemakai.
4. Mempertimbangkan bahwa sistem memenuhi permintaan pemakai yakni dengan menguji secara keseluruhan.

### 5.1.1 Lingkungan Implementasi

Lingkungan implementasi sistem ada 2 (dua) yaitu : lingkungan perangkat keras dan lingkungan perangkat lunak.

1. Perangkat Keras

   Pengenalan pola aksara batak dengan metode *Convolutional Neural Network* (CNN) dijalankan pada :

   a. *Processor* : *Intel* (R) *Core i3*

   b. Ram : 4 GB

   c. *Harddisk* : 500 GB

2. Perangkat Lunak

   Perangkat lunak yang digunakan dalam implementasi ini menggunakan :

   a. *Windows* 7, sebagai sistem operasi yang digunakan.

   b. MATLAB, untuk perancangan perangkat lunak dan pengkodingan aplikasi

### 5.1.2 Implementasi Penerapan Metode *Convolution Nueral Network* (CNN) untuk pengenalan pola aksara batak

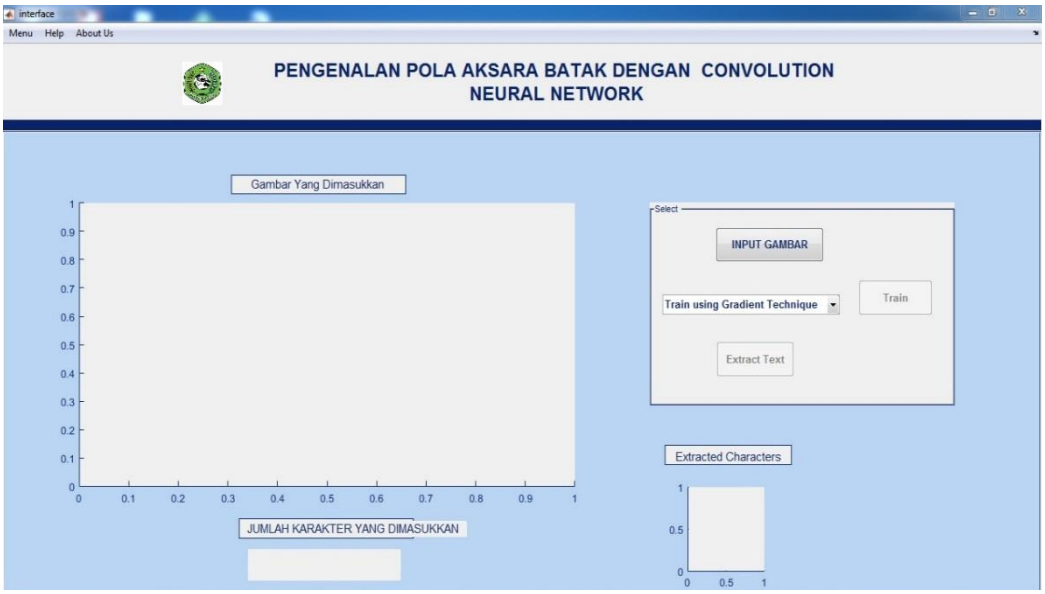
1. Menu *Home*

   Menu ini terdapat pada waktu pertama kali kita membuka aplikasi dan juga menu ini memberikan fasilitas untuk masuk ke menu utama aplikasi.


**Gambar 5.1 Menu *Home***

2. Tampilan Utama Aplikasi

   Tampilan utama aplikasi ini memiliki beberapa *button* yang mana fungsi dari setiap *button* berbeda – beda, *button input image* berfungsi untuk memasukkan gambar kedalam program, *button train* berfungsi untuk melatih gambar yang di masukkan ke dalam program, *button* ekstrak berfungsi untuk mengekstrak gambar yang sudah di *train* sebelumnya.

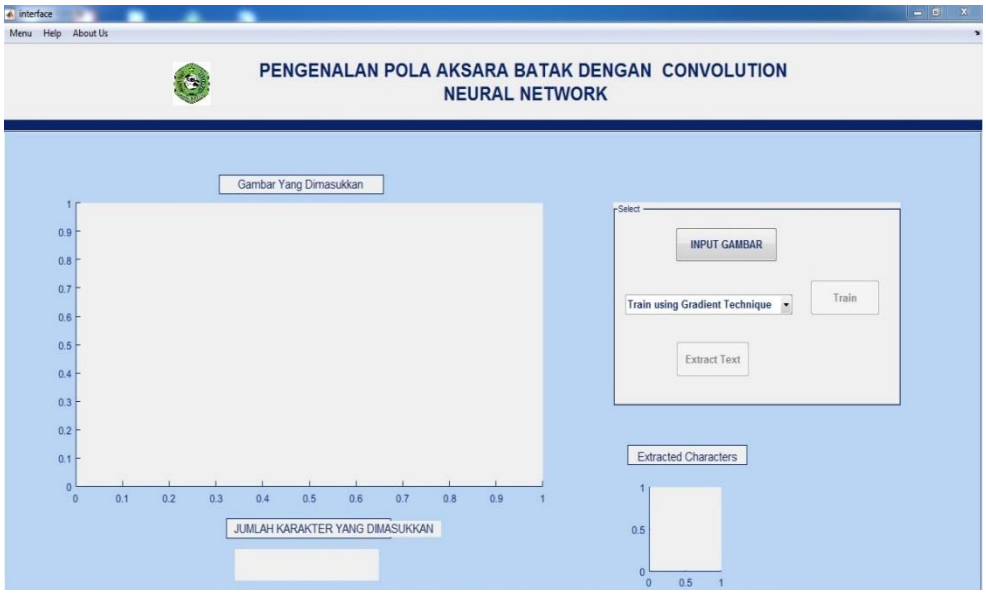



**Gambar 5.2 Tampilan Utama Aplikasi**

## 5.2 Pengujian Sistem

Pada tahap pengujian sistem akan dilakukan proses pengenalan kata degan tulisan tangan pada aksara Batak dengan menuliskan dikertas kemudian hasil tulisan tadi di *scan* menggunakan *scanner* berbentuk jpg. Pengujian dilakukan dengan 5 pola aksara Batak. Hasil pengujian sistem dengan metode *Convolutional Neural Network* (CNN) dapat dilihat melalui tabel 4.1, kolom citra *testing* berisi pola aksara batak yang ditulis dengan tangan kemudian di *scan*, kolom keluran merupakan hasil dari pengujian dengan sistem.

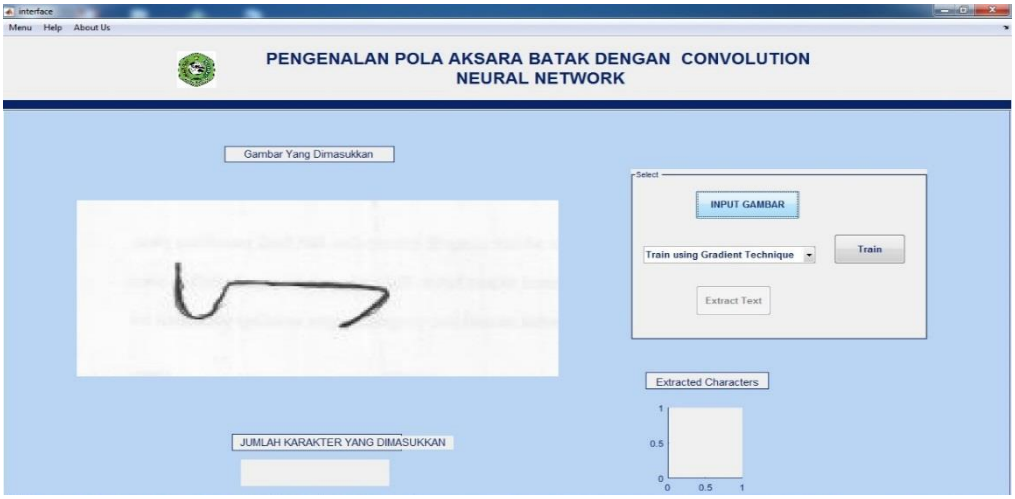
1. Pengujian Untuk Pola Aksara Batak ( a ) dengan MATLAB :




**Gambar 5.3 Tampilan Utama Aplikasi**

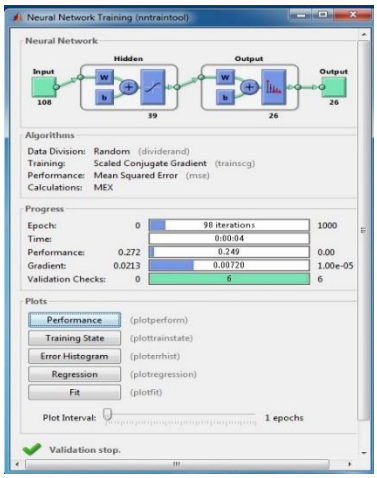
Pada tampilan ini memiliki beberap fungsi yaitu : tombol *input* gambar berfungsi untuk memasukkan gambar kedalam program, tombol *train* akan

muncul ketika gambar sudah di *input*kan kedalam program dan berfungsi untuk melatih atau *train* gambar yang sudah dimasukkan kedalam program.

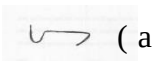



**Gambar 5.4 Tampilan Ketika Gambar Sudah di Inputkan**

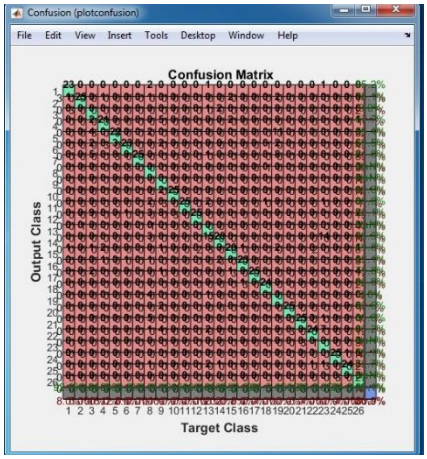
Setelah di klik tombol *input* gambar maka gambar akan muncul di dalam aplikasi, kemudian gambar akan di *train* dengan mengklik tombol *Train*.




**Gambar 5.5 Proses *Training***

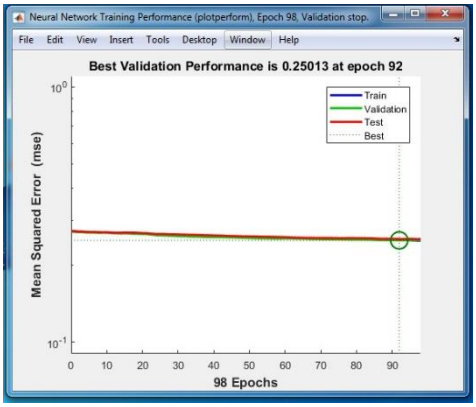
Dari gambar di atas dapat diliat untuk aksara  ( a ) untuk *epoch* nya didapat 90 dari 1000 *iterations* dan untuk lama proses *training* nya yaitu 4 detik.

Setelah muncul *layer training* maka akan muncul hasil matriks dari gambar yang di*input*kan ke dalam program untuk gambar matriknya dapat dilihat dibawah ini :



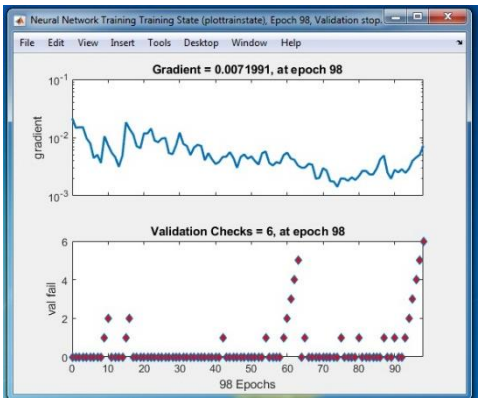

**Gambar 5.6 Matriks**

Untuk melihat *validation performance* dapat dilihat pada gambar dibawah:




**Gambar 5.7 *Validation Performance***

Dari gambar diatas untuk tingkat *validation performance* nya adalah 0.25013 pada *epoch* 92 dari 98 *epoch*. dan untuk *check validation* dapat di lihat pada gambar dibawah ini :




***5.8 Check Validation***

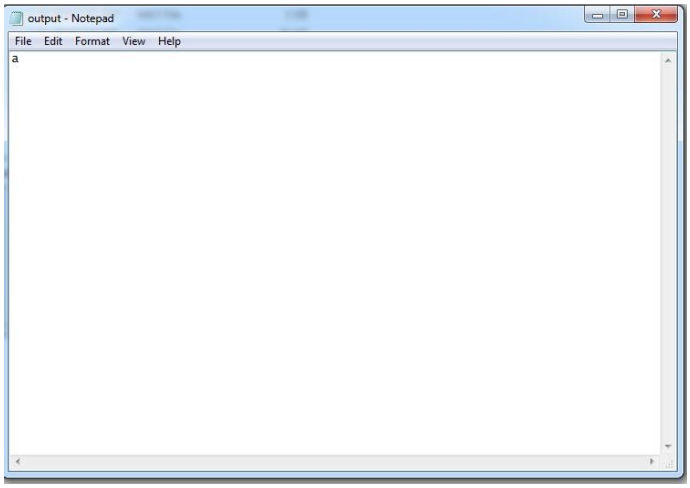
Dari gambar diatas dapat disimpulkan bahwa untuk *gradient* aksara tersebut adalah 0.0071991 terdapat pada *epoch* 98, dan untuk *validation checks* terdapat 6 pada *epoch* 98. Setelah didapat tingkat *validation* dan *performance* langkah selanjutnya adalah dengan menekan tombol ekstrak untuk mengektrak gambar yang telah dimasukkan kedalam program tersebut, untuk proses ektrak akan muncul beberapa *layer* sehingga setelah diekstrak maka akan muncul hasil dari ekstrak tersebut seperti gambar di bawah ini :


**Gambar 5.9 Proses Ekstrak**

Dari gambar *layer figure* 1 merupakan gambar yang di *input* kan kedalam program kemudian akan di proses sehingga muncul hasil ekstrak tersebut pada *layer*

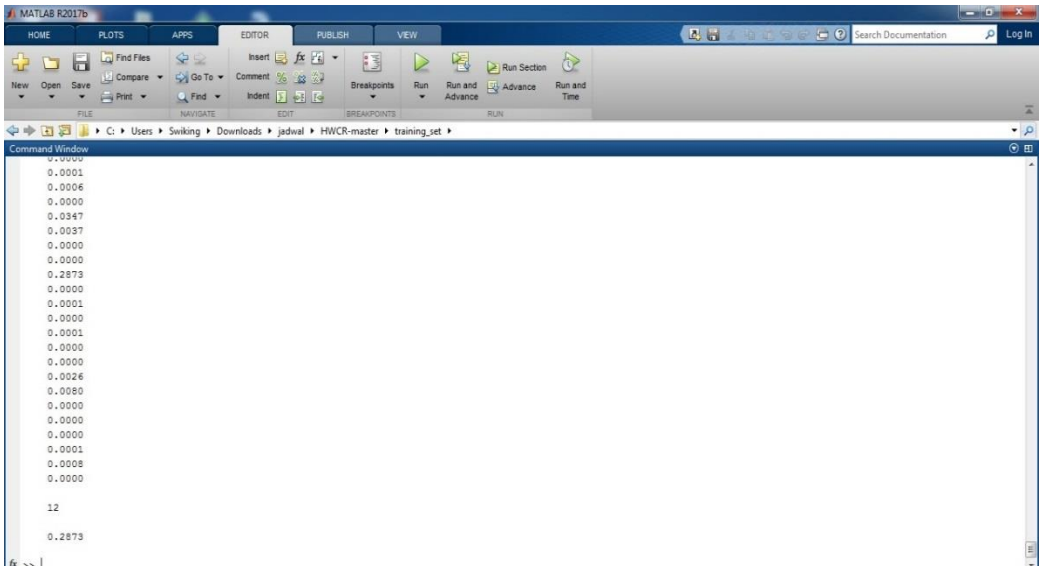
*figure* 5. Sehingga pada program akan tampil hasil *output* dari ekstrak tersebut seperti gambar dibawah ini :

**Gambar 5.10 Hasil *Output***

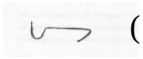



**Gambar 5.11 Hasil *Ouput* Aksara**

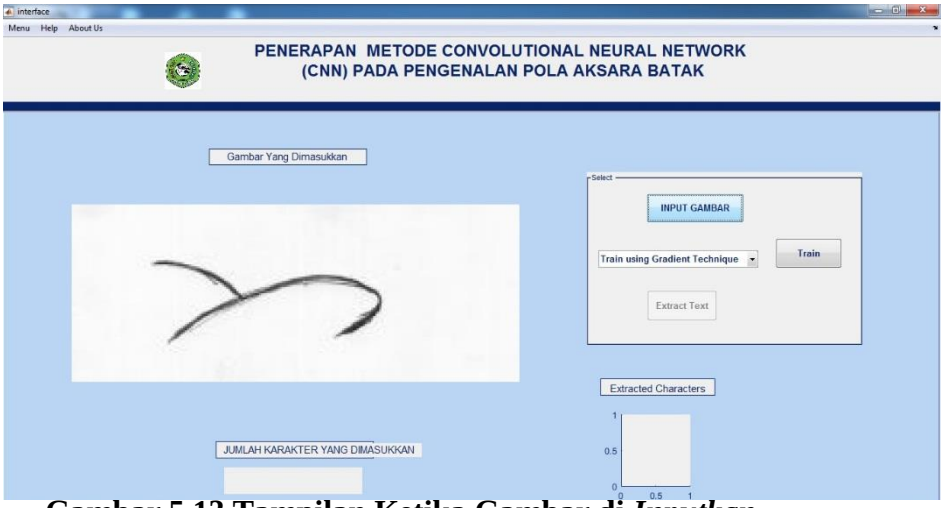
Setelah hasil *output* tampil maka program akan menampilkan tingkat akurasi dalam pengenalan pola tersebut, untuk hasil akurasi dapat dilihat pada gambar dibawah ini :




**Gambar 5.12 Hasil Akurasi**

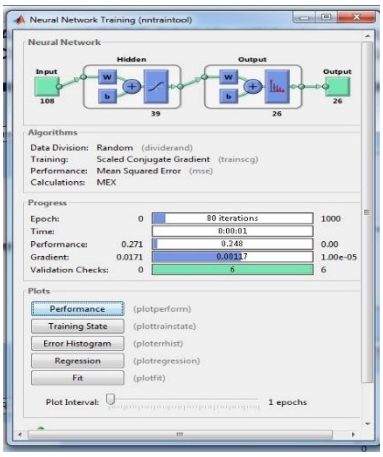
Pada gambar diatas hasil akurasi nya yaitu 0.2873 untuk pengenalan pola akasara Batak  ( a ).

1. Pengujian Untuk Pola Aksara Batak  ( ha ) dengan MATLAB



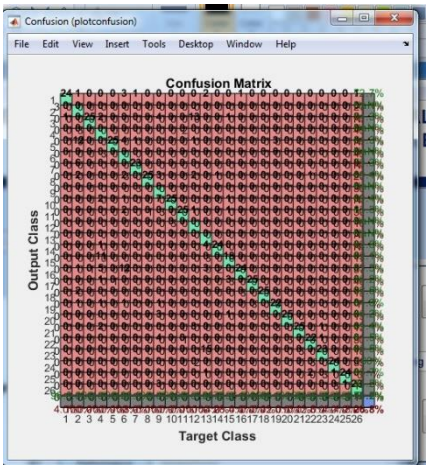

**Gambar 5.13 Tampilan Ketika Gambar di *Inputkan***

Setelah gambar di masukkan kedalam program dengan menekan tombol *Input* gambar_kemudian akan dilanjutkan dengan proses *train* dengan menekan tombol *Train* maka akan muncul *layer training* seperti gambar dibawah ini:



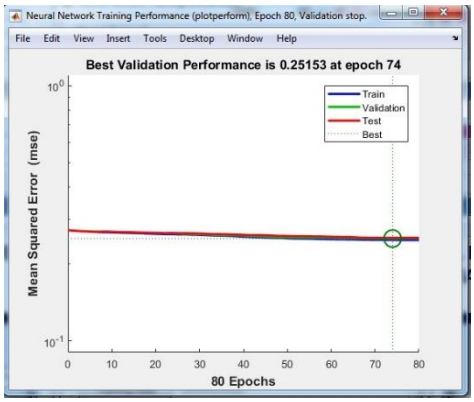

**Gambar 5.14 *Layers Training***

Pada gambar tersebut *progress* untuk *training* aksara tersebut diperoleh 80 *iterations epoch* dari 1000 *epoch* serta membutuhkan waktu untuk melakukan *train* selama 1 detik dan mendapat 6 *validation check* untuk proses *train* aksara (ha). Setelah muncul *layers training* maka akan muncul matrik dari hasil *train* tersebut seperti gambar dibawah ini:



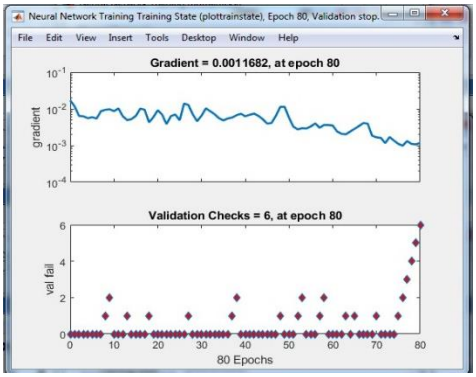

**Gambar 5.15 *Confusion Matrix***

Setelah muncul *confusion matrix* untuk melihat *performance* dari hasil *train* bisa dilihat pada *layer training* dengan menekan tombol *performance* maka akan muncul seperti gambar dibawah ini :



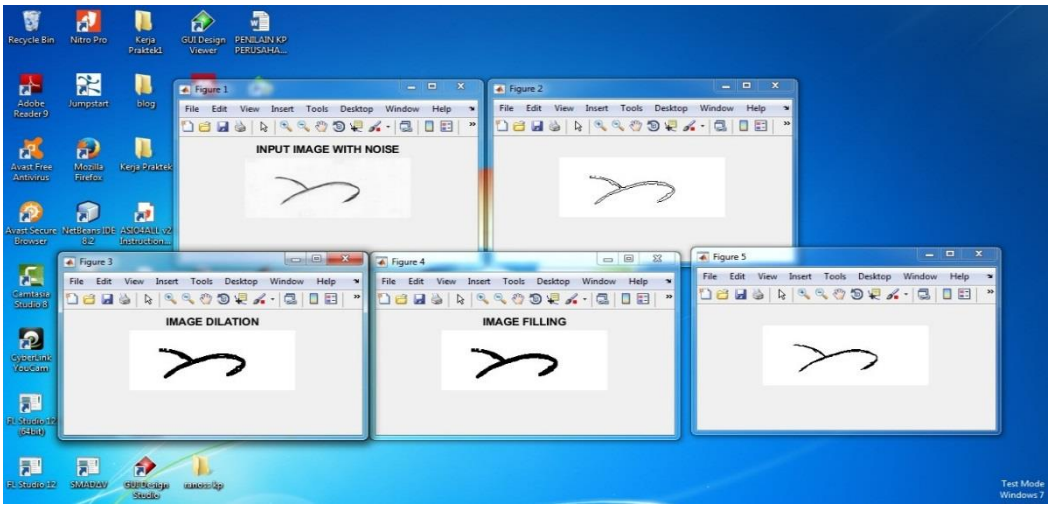

**Gambar 5.16 *Layers Training Performance***

Dari gambar di atas dapat disimpulkan bahwa untuk hasil *train* aksara (ha) *validation* yang bagus terdapat pada *epoch* 74 dari 80 *epoch* yaitu 0.25153. untuk melihat *Gradient* dan *validation check* nya seperti gambar dibawah ini :




**Gambar 5.17 *Training State***

dari gambar diatas dapat disimpulkan untuk *gradient* nya adalah 0.0011682 terdapat pada *epoch* 80 dan untuk *validation check* nya ada 6 yang terletak pada *epoch* 80. Untuk proses pengenalan aksara dapat dilihat pada gambar di bawah ini :

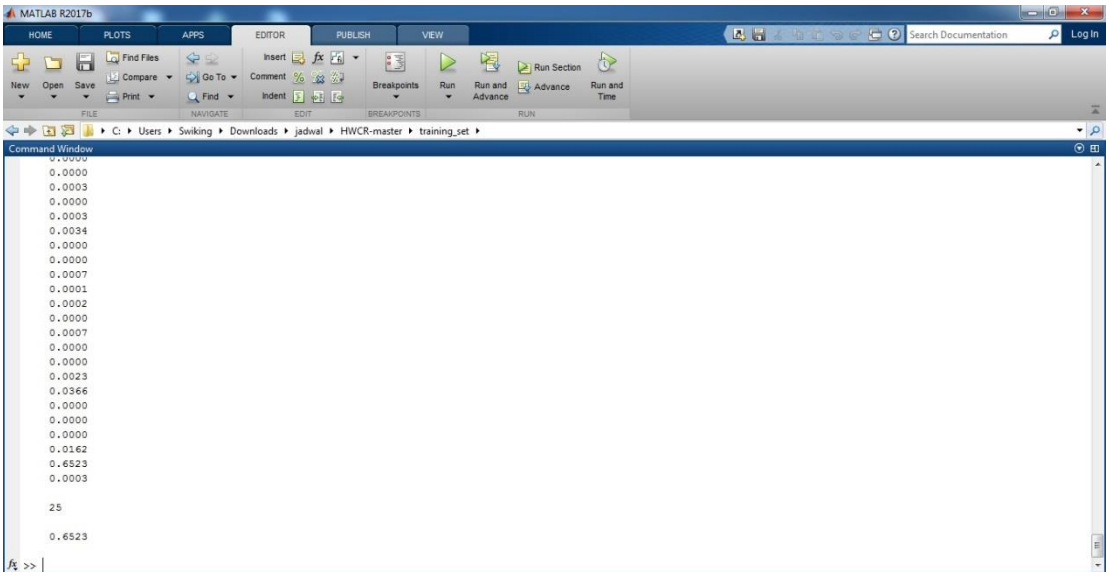



**Gambar 5.18 *Training Noise***

Pada gambar tersebut pada layer *figure* 1 merupakan gambar yang telah dimasukkan kedalam program kemudian akan diproses dengan *image dilation*, *image filling* sehingga menghasilkan *output* nya pada *layer figure* 5. Untuk hasil *output* nya dapat dilihat pada gambar dibawah :

**Gambar 5.19 *Output***

Setelah *output* nya ditampilkan untuk melihat tingkat akurasinya adalah 0.6523. untuk lebih jelasnya dapat dilihat pada gambar dibawah ini :

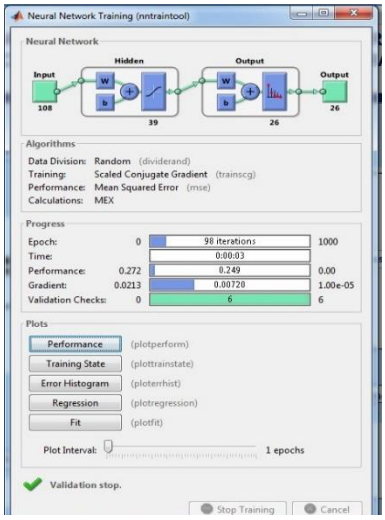



**Gambar 5.20 Akurasi Pengenalan Pola Aksara**

2. Pengujian untuk pola aksara Batak  ( na ) dengan MATLAB

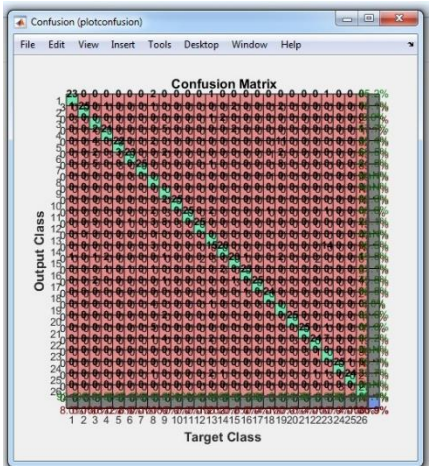

**Gambar 5.21 *Input* Aksara**

Setelah gambar di masukkan kedalam program dengan menekan tombol *Input* gambar kemudian akan dilanjutkan dengan proses *train* dengan menekan tombol *Train* maka akan muncul *layer training* seperti gambar dibawah ini:



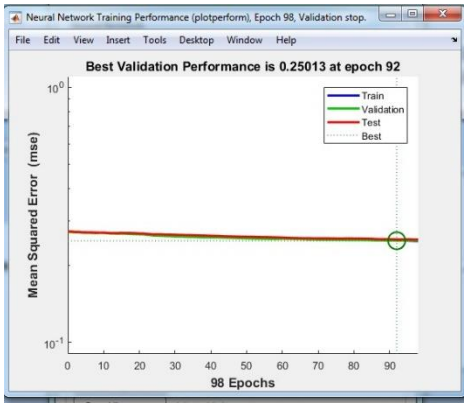

**Gambar 5.22 *Layer Training***

Pada gambar tersebut *progress* untuk *training* aksara tersebut diperoleh 98 *iterations epoch* dari 1000 *epoch* serta membutuhkan waktu untuk melakukan *train* selama 3 detik dan mendapat 6 *validation check* untuk proses *train*. Setelah muncul *layers training* maka akan muncul matrik dari hasil *train* tersebut seperti gambar dibawah ini:



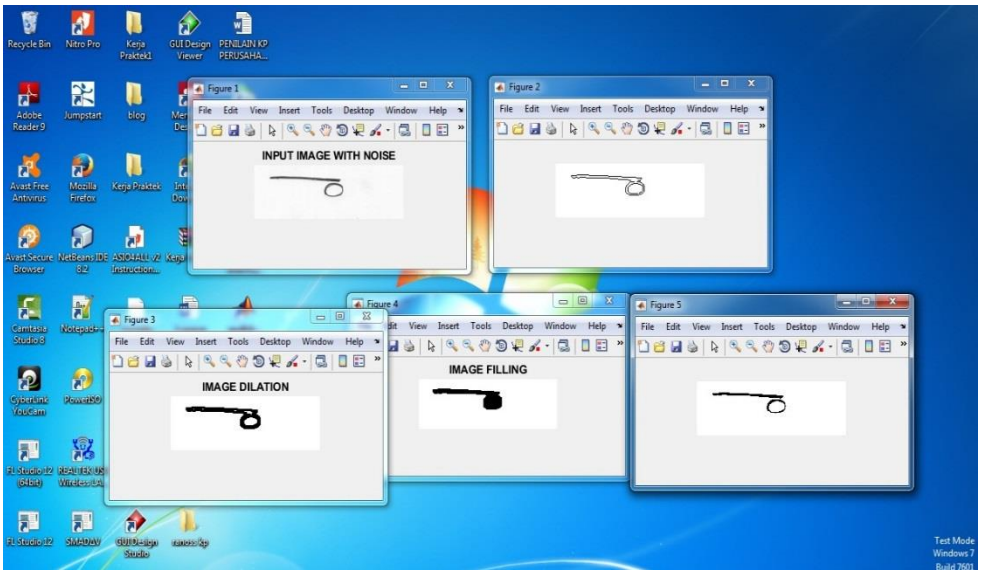

**Gambar 5.23 *Layer Matriks***

Setelah muncul *confusion matrix* untuk melihat *performance* dari hasil *train* bisa dilihat pada *layer training* dengan menekan tombol *performance* maka akan muncul seperti gambar dibawah ini :



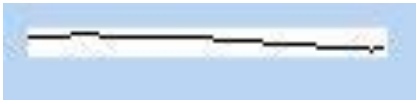

**Gambar 5.24 *Layers Training Performance***

Dari gambar di atas dapat disimpulkan bahwa untuk hasil *train validation* yang bagus terdapat pada *epoch* 74 dari 80 *epoch* yaitu 0.25013. untuk proses *train* aksara dapat dilihat pada gambar dibawah ini :

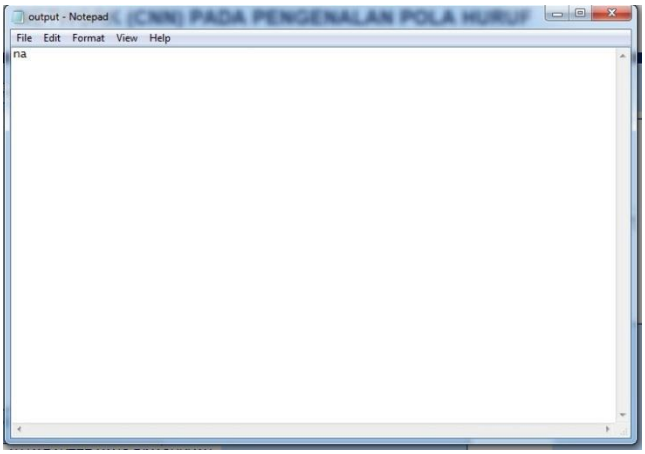



**Gambar 5.25 Proses *Train***

Pada gambar tersebut pada layer *figure* 1 merupakan gambar yang telah dimasukkan kedalam program kemudian akan diproses dengan *image dilation*, *image filling* sehingga menghasilkan *output* nya pada *layer figure* 5. Untuk hasil *output* nya dapat dilihat pada gambar dibawah :

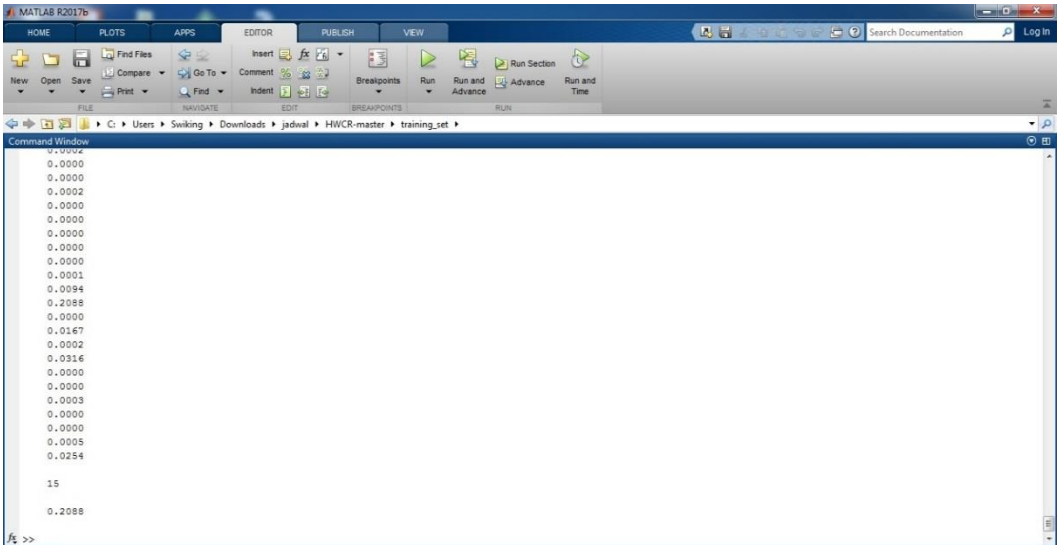


**Gambar 5.26 *Output***



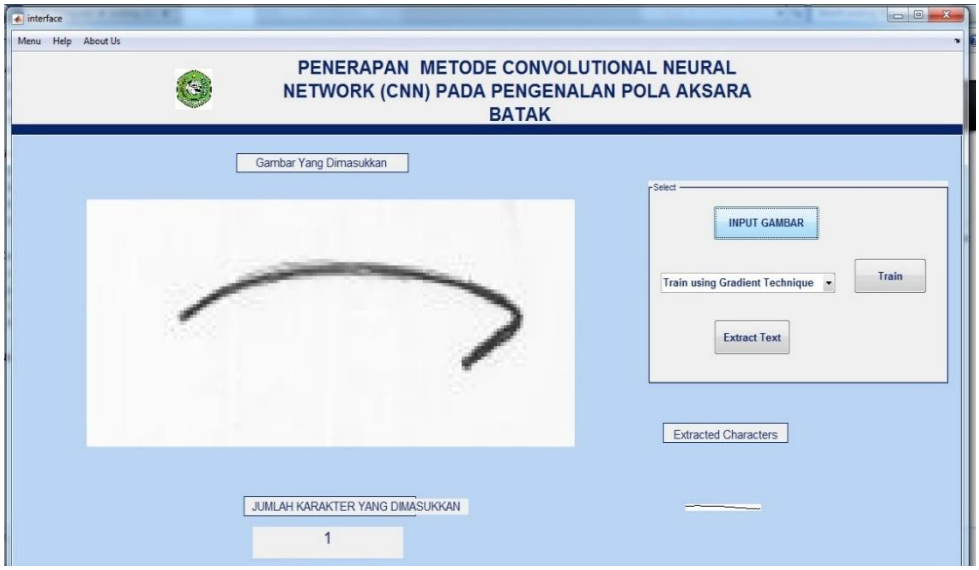

**Gambar 5.27 *Output* Latin**

Setelah *output* nya ditampilkan untuk melihat tingkat akurasinya adalah 0.2088. untuk lebih jelasnya dapat dilihat pada gambar dibawah ini :



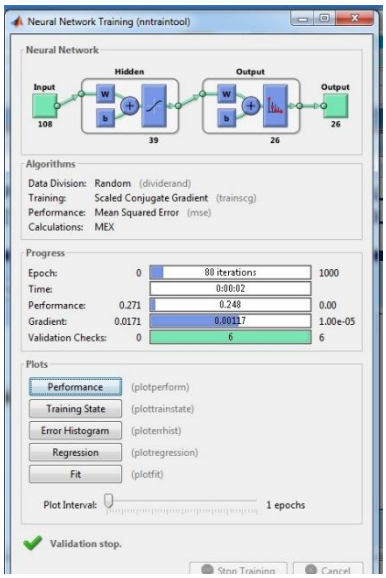

**Gambar 5.28 Akurasi Pengenalan Pola Aksara**

3. Pengujian Untuk Pola Aksara Batak  ( nga ) Dengan MATLAB



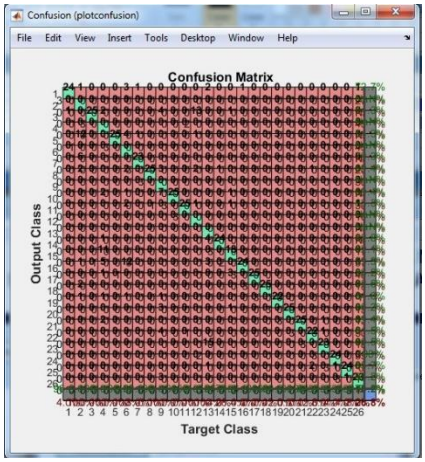

**Gambar 5.29 *Input* Aksara**

Setelah gambar di masukkan kedalam program dengan menekan tombol *Input* gambar_kemudian akan dilanjutkan dengan proses *train* dengan menekan tombol *Train* maka akan muncul *layer training* seperti gambar dibawah ini.



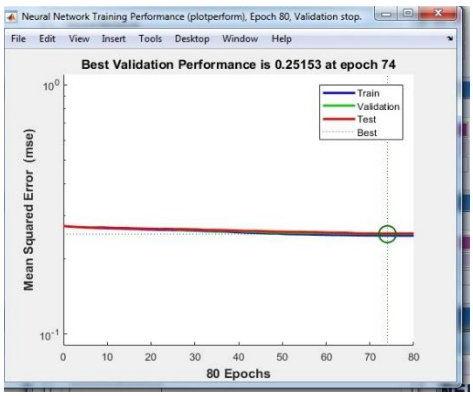

**Gambar 5.30 *Layer Training***

Pada gambar tersebut *progress* untuk *training* aksara tersebut diperoleh 80 *iterations epoch* dari 1000 *epoch* serta membutuhkan waktu untuk melakukan *train* selama 2 detik dan mendapat 6 *validation check* untuk proses *train*. Setelah muncul *layers training* maka akan muncul matrik dari hasil *train* tersebut seperti gambar dibawah ini:



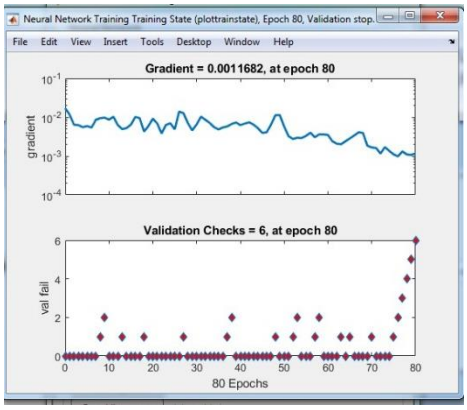

**Gambar 5.31 *Layer Matriks***

Setelah muncul *confusion matrix* untuk melihat *performance* dari hasil *train* bisa dilihat pada *layer training* dengan menekan tombol *performance* maka akan muncul seperti gambar dibawah ini :



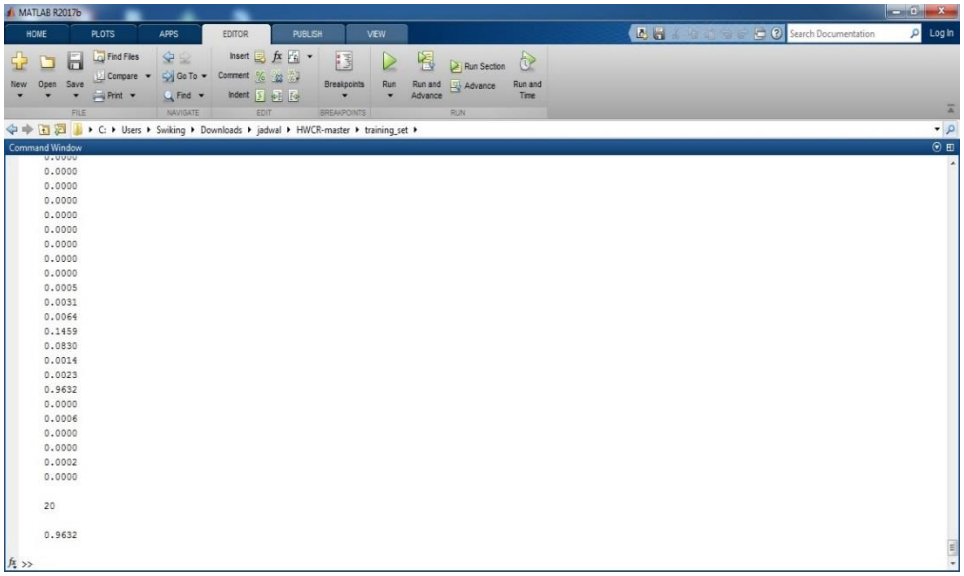

**Gambar 5.32 *Layers Training Performance***

Dari gambar di atas dapat disimpulkan bahwa untuk hasil *train validation* yang bagus terdapat pada *epoch* 74 dari 80 *epoch* yaitu 0.25153. untuk melihat *Gradient* dan *validation check* nya seperti gambar dibawah ini :



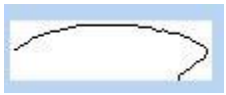

**Gambar 5.33 *Training State***

dari gambar diatas dapat disimpulkan untuk *gradient* nya adalah 0.0011682 terdapat pada *epoch* 80 dan untuk *validation check* nya ada 6 yang terletak pada *epoch* 80. Untuk melihat akurasi dapat dilihat pada gambar di bawah:

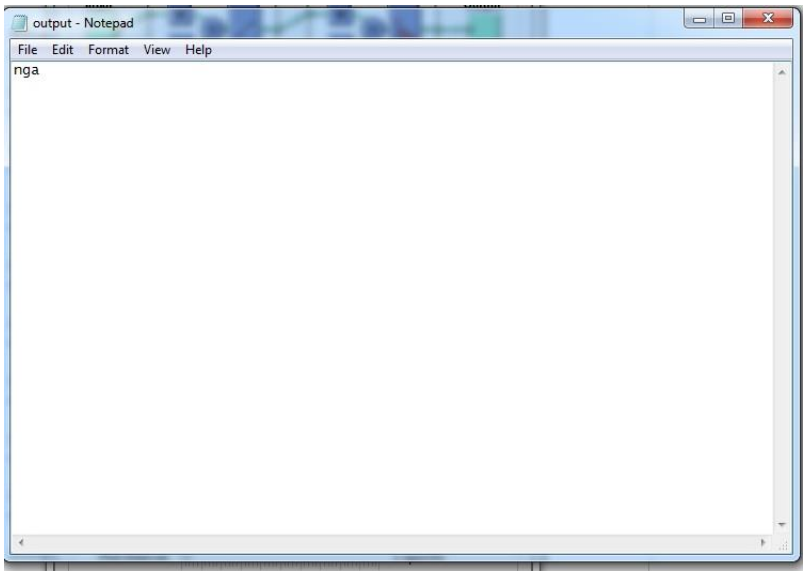



**Gambar 5.34 Akurasi *Training***

Pada gambar diatas hasil akurasi nya yaitu 0.2873, hasil *output* nya dapat dilihat pada gambar dibawah ini:

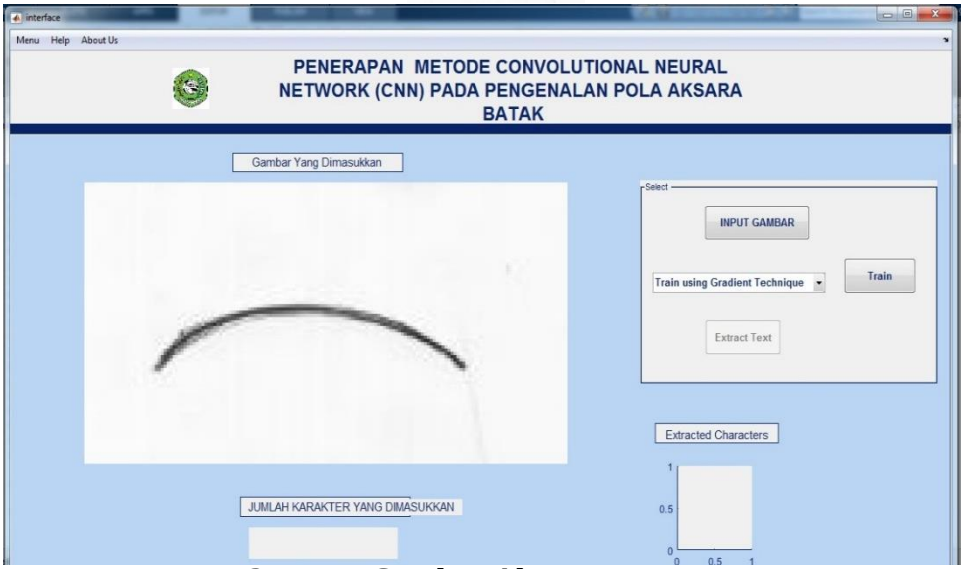


**5.35 Hasil *Output***



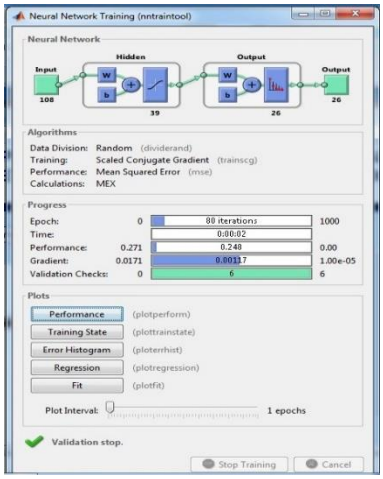

**Gambar 5.36 *Output* Latin**

4. Pengenalan Pola Aksara Batak  ( pa ) dengan MATLAB




**5.37 *Input* Gambar Aksara**

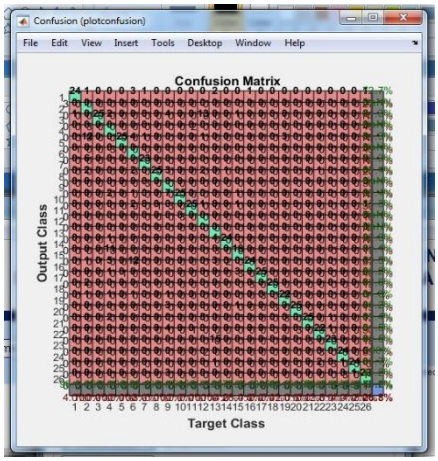
Setelah gambar di masukkan kedalam program dengan menekan tombol *Input* gambar_kemudian akan dilanjutkan dengan proses *train* dengan menekan tombol *Train* maka akan muncul *layer training* seperti gambar dibawah ini:




**5.38 *Layer Training***

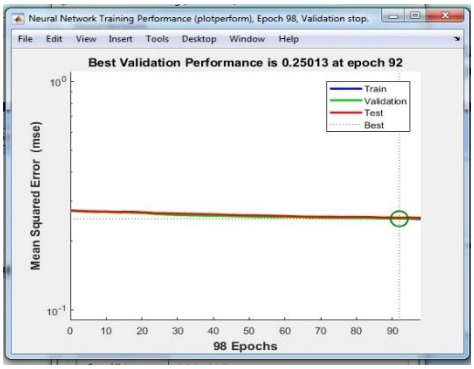
Pada gambar tersebut *progress* untuk *training* aksara tersebut diperoleh 98 *iterations epoch* dari 1000 *epoch* serta membutuhkan waktu untuk

melakukan *train* selama 4 detik dan mendapat 6 *validation check* untuk proses *train*. Setelah muncul *layers training* maka akan muncul matrik dari hasil *train* tersebut seperti gambar dibawah ini:



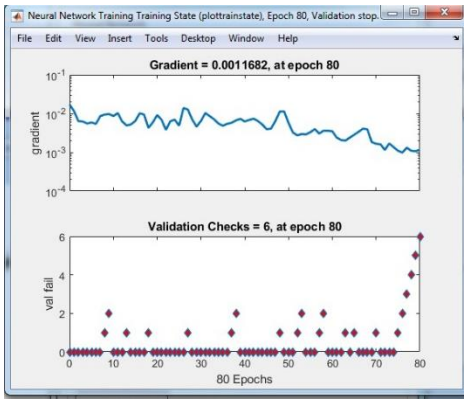

**5.39 *Confusion* Matriks**

Setelah muncul *confusion matrix* untuk melihat *performance* dari hasil *train* bisa dilihat pada *layer training* dengan menekan tombol *performance* maka akan muncul seperti gambar dibawah ini :



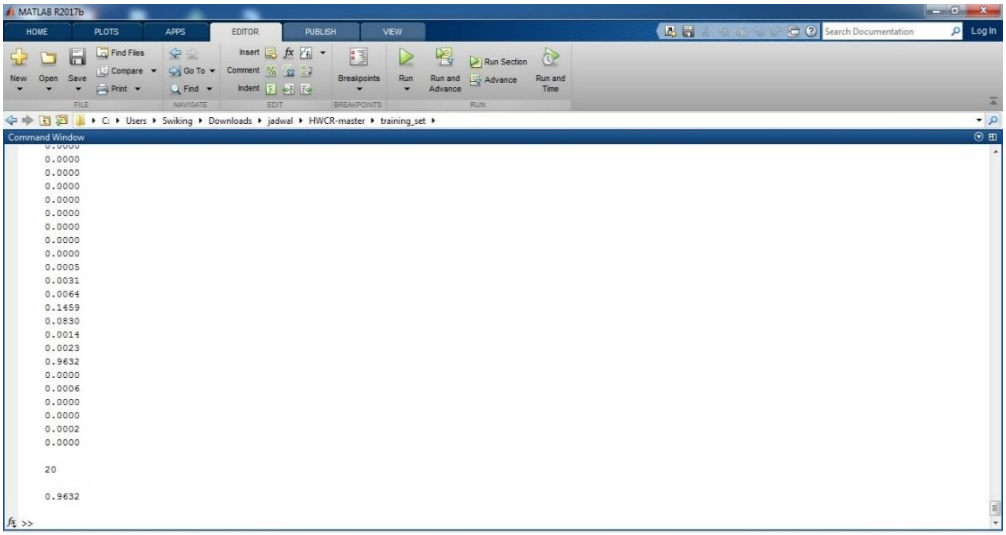

**Gambar 5.40 *Layers Training Performance***

Dari gambar di atas dapat disimpulkan bahwa untuk hasil *train validation* yang bagus terdapat pada *epoch* 92 dari 98 *epoch* yaitu 0.25013. untuk melihat *Gradient* dan *validation check* nya seperti gambar dibawah ini :



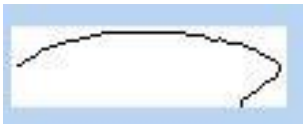

**Gambar 5.41 *Training State***

dari gambar diatas dapat disimpulkan untuk *gradient* nya adalah 0.0071991 terdapat pada *epoch* 98 dan untuk *validation check* nya ada 6 yang terletak pada *epoch* 98. Untuk melihat akurasi dapat dilihat pada gambar di bawah:



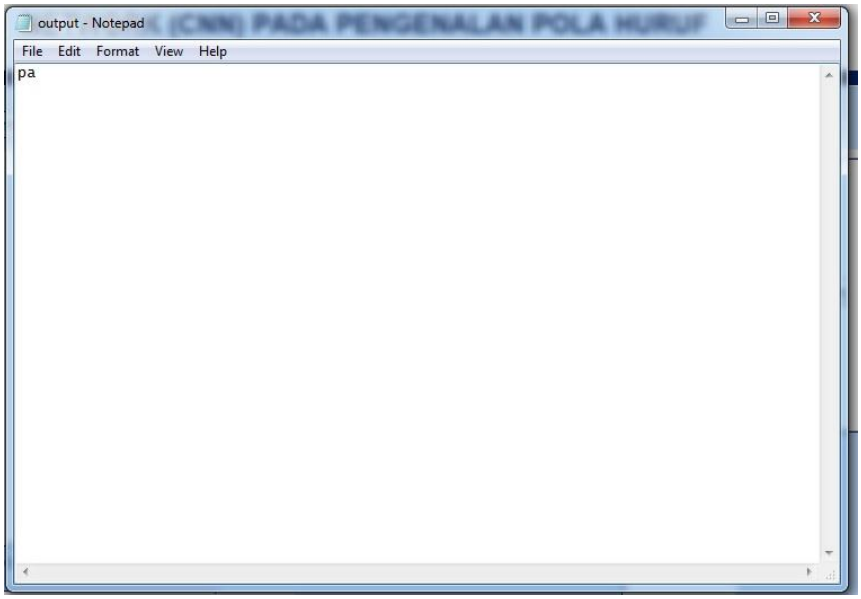

**Gambar 5.42 Akurasi *Training***

Pada gambar diatas hasil akurasi nya yaitu 0.6548, hasil *output* nya dapat dilihat pada gambar dibawah ini:



**Gambar 5.43 Hasil *Output***

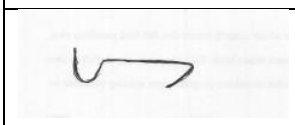
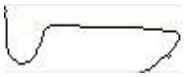
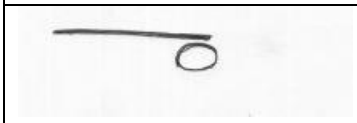
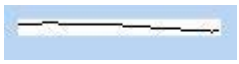
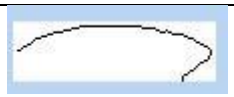
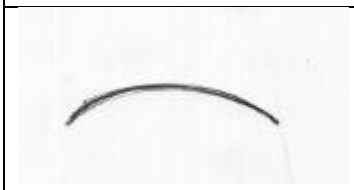
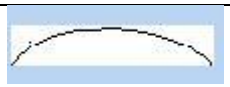
Untuk *coding train* dan *coding Convolutional Neural Network* (CNN ) pengenalan pola aksara Batak dapat dilihat pada tabel dibawah ini:




**Gambar 5.44 *Output* Latin**

Dari pengujian diatas pegujian dilakukan dengan sampel 5 pola aksara Batak hasil pengujian dengan metode *Convolutional Neural Netwrok* ( CNN ) dapat dilihat pada tabel dibawah :

**Tabel 5.3 Hasil Pengujian Sistem**

| ***Inputan* Citra** | ***Output* Citra** | **Kata** | **Nilai** | **Akurasi** |
|---|---|---|---|---|
|  |  | a | Benar | 0.2873 |
|  |  | na | Salah | 0.2088 |
|  |  | ha | benar | 0.6523 |
|  |  | Nga | benar | 0.9632 |
|  |  | pa | benar | 0.6548 |

Dari hasil pengujian sistem yang dilakukan, nilai akurasi citra dipengaruhi oleh bentuk goresan pola pada saat menulis pola aksara Batak tersebut dengan tangan. Goresan tersebut akan disesuaikan dengan data training

# BAB 6

# PENUTUP

Bab ini akan membahas tentang kesimpulan dari implementasi metode yang diajukan untuk pengenalan pola tulisan tangan aksara Batak, serta saran – saran yang dapat dilakukan selanjutnya.

## 6.1 Kesimpulan

Berdasarkan pengujian sistem menggunakan metode *Convolutional Neural Network* (CNN) pada pengenalan pola aksara Batak didapatkan beberapa kesimpulan sebagai berikut :

a. Nilai akurasi citra yang dicapai pada citra dipengaruhi oleh goresan kata yang dituliskan. Goresan kata pada data testing akan disesuaikan dengan data *training*, sehingga goresan yang mendekati dengan data *training* akan mendapatkan hasil yang lebih baik.
b. Hasil *output* dari pengujian sistem adalah berupa ektrak perkata dari aksara Batak serta bacaan dari aksara Batak tersebut yang telah ditulis dan di *scan* sebelumnya.

## 6.2 Saran

Saran penulis untuk penelitian selanjutnya adalah sebagai berikut :

1. Sistem dapat mendukung pengenalan kata dengan jumlah suku kata lebih dari dua
2. Sistem lebih baik dari sebelumnya dan bisa diintegrasikan ke Android agar lebih mudah digunakan.

# DAFTAR PUSTAKA


[1] Rouza,Erni.(2015). *Penerapan Jaringan Syaraf Tiruan Untuk Prediksi Jenis Cacing Nemotada Usus Yang Menginfeksi Siswa Dengan Metode Learning Vector Quantization*. Skripsi tidak di publikasikan.

[2] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "*Imagenet Classification Wideep Convolutional Neural Networks*." In Advances in neural information processing

[3] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z. Karpathy, A.,Khosla, A., Bernstein, M. and Berg, A.C., 2015. *Imagenet Large Scalevisual Recognition Challenge. International Journal of Computer Vision*, 115(3), pp.211-252.sytems, pp. 10971105.2012

[4] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich."*Goingdeeper With Convolutions.*" In Proceedings of the IEEE Conference on Computer.

[5] He, K., Zhang, X., Ren, S., & Sun, J. "*Deep Residual Learning for Image Recognition*".IEEE Conference on Computer Vision and Pattern Recognition.2015 Vision and Pattern Recognition, pp. 1-9. 2015.

[6] Hu, et al. (2015). *Transferring Deep Convolutional Neural Networks For The Scene Classificationof Hight-Resolution Remote Sesing Imagery.Remote Sensing*, /(11), 14680- 14707. https://doi.org/10.3390/rs71114680

[7] Goodfellow, I., Bengio, Y. & Courville, A. 2016. *Deep Learning (Adaptive Computation and Machine Learning series)*. The MIT Press: Cambridge

[8] Bui, V & Chang, L. 2016. *Deep Learning Arsitectures for hard Character*

[9] Devikar, P. 2016. *Transfer Learning for Image Classification of Various Dog Breeds International Journal of Advanced Research in Computer Engineering and Technology (IJARCET),* Vol.5: 2707-2715

[10] Haykin, S. (1999) *Neural Networks A Comprehensive Foundation*. 2nd Edition, Prentice- Hall, Englewood Cliffs.

[11] Fausett, Laurene. 1994. *Fundamentals of Neural Networks Architectures, Algorithms, and Applications*. London: Prentice Hall, Inc.

[12] Azcarraga, A (1999) *Artificial Neural Network http:www.comp.nus.edu.sg/ Artificial Neural Network*/index.html.

[13] Richad, Goering (2004). *Matlab edges closer to electronic design automation world EE Times*

[14] Sitorus, Johanes (2015). Perancangan aplikasi pengenalan pola huruf aksara batak toba menerapkan metode *Direction Feature Extraction* (DFE).jl.Sisingamangaraja

[15] Sitinjak, Suriski ( 2012 ). Pengenalan tulisan tangan aksara batak Toba menggunakan *BACKPROPAGATION*

[16] Sena, S. (2018, Mei 27*). Pengenalan Deep Learning Part 7 : Convolutional NeuralNetwork* (CNN).

# LAMPIRAN

**Tabel 5.1 Tabel *Source Code Train***

```
rng('default');

load('input108.mat');
load('target650.mat');

inputs = input108';
targets = target650';


% Create a Pattern Recognition Network
hiddenLayerSize = 39;
net = patternnet(hiddenLayerSize);

% Choose Input and Output Pre/Post-Processing Functions
% For a list of all processing functions type: help nnprocess
net.inputs{1}.processFcns = {'removeconstantrows','mapminmax'};
net.outputs{2}.processFcns = {'removeconstantrows','mapminmax'};


% Setup Division of Data for Training, Validation, Testing
% For a list of all data division functions type: help nndivide
net.divideFcn = 'dividerand';  % Divide data randomly
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 80/100;
net.divideParam.testRatio = 20/100;

% For help on training function 'trainscg' type: help trainscg
% For a list of all training functions type: help nntrain
net.trainFcn = 'trainscg';  % Scaled conjugate gradient

% Choose a Performance Function
% For a list of all performance functions type: help nnperformance
net.performFcn = 'mse';  % Mean squared error

% Choose Plot Functions
% For a list of all plot functions type: help nnplot
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
  'plotregression', 'plotfit'};

net.efficiency.memoryReduction = 100;
net.trainParam.max_fail = 6;
net.trainParam.min_grad=1e-5;
net.trainParam.show=10;
net.trainParam.lr=0.9;
net.trainParam.epochs=1000;
net.trainParam.goal=0.00;
```

```
% Train the Network
[net,tr] = train(net,inputs,targets);

% Test the Network
outputs = net(inputs);
errors = gsubtract(targets,outputs);
performance = perform(net,targets,outputs)

% Recalculate Training, Validation and Test Performance
trainTargets = targets .* tr.trainMask{1};
valTargets = targets  .* tr.valMask{1};
testTargets = targets  .* tr.testMask{1};
trainPerformance = perform(net,trainTargets,outputs)
valPerformance = perform(net,valTargets,outputs)
testPerformance = perform(net,testTargets,outputs)

% View the Network
view(net)


disp('after training')
y1 = sim(net,inputs);
y1=abs(y1);
y1=round(y1);

save   d:\training_set\net net;

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
figure, plotconfusion(targets,outputs)
%figure, plotroc(targets,outputs)
%figure, ploterrhist(errors)
```

**Tabel 5.2 *Source Code Convolutional Neural Network* (CNN)**

```
clear all;

train_images = reshape((loadMNISTImages('train-images.idx3-
ubyte')),[28,28,1,60000]);
train_labels = ((loadMNISTLabels('train-labels.idx1-ubyte')));
```

```
test_images = reshape((loadMNISTImages('t10k-images.idx3-
ubyte')),[28,28,1,10000]);
test_labels = ((loadMNISTLabels('t10k-labels.idx1-ubyte')));


%% Code to generate 10 fold train data

train_images_fold=zeros(28,28,1,6000,10);
train_labels_fold=zeros(6000,1,10);

for i=(1:10)
train_images_fold(:,:,:,:,i)=train_images(:,:,:,(6000*(i-
1))+1:6000*i);
train_labels_fold(:,1,i)=train_labels((6000*(i-1))+1:6000*i,1);
end

train_images_data=zeros(28,28,1,54000,10);
train_labels_data=zeros(54000,1,10);
cross_valid_images=zeros(28,28,1,6000,10);
cross_valid_labels=zeros(6000,1,10);

for (i=1:10)
    cross_valid_images(:,:,:,:,i)=train_images_fold(:,:,:,:,i);
    cross_valid_labels(:,:,i)=train_labels_fold(:,:,i);
    k=1;
    for j=(1:10)
        if (j~=i)
           train_images_data (:,:,:,(6000*(k-
1))+1:6000*k,i)=train_images_fold(:,:,:,:,j);
           train_labels_data ((6000*(k-
1))+1:6000*k,1,i)=train_labels_fold(:,:,j);
           k=k+1;
        end
    end
end


%% Code to display first 100 training images
figure(1)
for i=1:100
        subplot(10,10,i);
        imshow(train_images(:,:,:,i));
end

%% Code to generate the ANN layers
```

```
inputlayer = imageInputLayer([28 28
1],'DataAugmentation','none',...
    'Normalization','none','Name','input');

%%volume 28*28*1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%                  Layer 1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%

convlayer1 = convolution2dLayer(4,32,'Stride',1,'Padding',0, ...
    'BiasLearnRateFactor',2,'NumChannels',1,...
    'WeightLearnRateFactor',2, 'WeightL2Factor',1,...
    'BiasL2Factor',1,'Name','conv1');

%%volume 25*25*32

convlayer1.Weights = randn([4 4 1 32])*0.1;
convlayer1.Bias = randn([1 1 32])*0.1;

relulayer1 = reluLayer('Name','relu1');

localnormlayer1 = crossChannelNormalizationLayer(3,'Name',...
    'localnorm1','Alpha',0.0001,'Beta',0.75,'
K',2);

maxpoollayer1 =
maxPooling2dLayer(3,'Stride',3,'Name','maxpool1','Padding',1);

%volume 9*9*32

droplayer1 = dropoutLayer(0.35);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%                    Layer 2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
```

```
convlayer2 = convolution2dLayer(3,16,'Stride',1, 'Padding',0,...
    'BiasLearnRateFactor',1,'NumChannels',32,...
    'WeightLearnRateFactor',1, 'WeightL2Factor',1,...
    'BiasL2Factor',1,'Name','conv2');

%7*7*16

convlayer2.Weights = randn([3 3 32 16])*0.0001;
convlayer2.Bias = randn([1 1 16])*0.00001;

relulayer2 = reluLayer('Name','relu2');

localnormlayer2 = crossChannelNormalizationLayer(3,'Name',...
    'localnorm2','Alpha',0.0001,'Beta',0.75,'K',2);

%maxpoollayer2 =
maxPooling2dLayer(2,'Stride',2,'Name','maxpool1','Padding',1);

droplayer2 = dropoutLayer(0.25);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%%                    Output Layers
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%

fullconnectlayer =
fullyConnectedLayer(10,'WeightLearnRateFactor',1,...
    'BiasLearnRateFactor',1,'WeightL2Factor',1,'BiasL2Factor',1,...
    'Name','fullconnect1');
fullconnectlayer.Weights = randn([10 784])*0.0001;
fullconnectlayer.Bias = randn([10 1])*0.0001+1;

smlayer = softmaxLayer('Name','sml1');

coutputlayer = classificationLayer('Name','coutput');


%% Code to define training parameters

options = trainingOptions('sgdm',...
      'LearnRateSchedule','piecewise',...
      'LearnRateDropFactor',0.75,...
      'LearnRateDropPeriod',1,'L2Regularization',0.0001,...
```

```
      'MaxEpochs',16,'Momentum',0.9,'Shuffle','once',...
      'MiniBatchSize',15,'Verbose',1,...

'CheckpointPath','E:\ccfuser3\checkpoints','InitialLearnRate',0.043
);

 %% Code to make the network
    layers =[inputlayer, convlayer1, relulayer1,localnormlayer1,
...
       maxpoollayer1, droplayer1,...
       convlayer2, relulayer2, localnormlayer2,droplayer2,...
       fullconnectlayer, smlayer, coutputlayer];


   %% Train the CNN
   train_im=zeros(28,28,1,54000);
   train_lb=categorical(zeros(54000,1));
   cross_valid_im=zeros(28,28,1,6000);
   cross_valid_lb=zeros(6000,1);
   for (i=1:10)
       train_im=train_images_data(:,:,:,:,i);
       train_lb=categorical(train_labels_data(:,:,i));
       cross_valid_im=cross_valid_images(:,:,:,:,i);
       cross_valid_lb=categorical(cross_valid_labels(:,:,i));
trainedNet(i) = trainNetwork(train_im,train_lb,layers,options);

[Ypred,scores] = classify(trainedNet(i),cross_valid_im);
score(i) = sum((Ypred==cross_valid_lb))/numel(cross_valid_lb)
   end

   [max,index]=max(score);
   best_model=trainedNet(index);
```

# DAFTAR RIWAYAT HIDUP

Penulis dilahirkan di Pasir Pengaraian Kecamatan Rambah Kabupaten Rokan Hulu pada tanggal 26 Juni 1994 dari Ayahanda Makmur dan Ibunda Nurmiati. Penulis merupakan anak pertama dari dua bersaudara yang bernama Nurislamiati. Pada tahun 2000, penulis masuk Sekolah Dasar Negeri 002 Negeri Rambah, Rokan Hulu dan menyelesaikannya pada tahun 2006. Kemudian melanjutkan di tingkat Sekolah Menengah Pertama di MTS Negeri Rambah, Rokan Hulu dan menamatkannya pada tahun 2009. Tahun 2013 berhasil menamatkan Sekolah Menengah Atas di SMA IT Bangkinang.

Setelah menamatkan pendidikan formal di tahun 2013, penulis masuk ke Universitas Islam Negeri Sultan Syarif Kasim ( UIN Suska ) Riau, Pekanbaru sebagai mahasiswa Jurusan Sistem Informasi

kemudian pada tahun 2015 penulis masuk ke Universitas Pasir Pengaraian, sebagai mahasiswa jurusan Teknik Informatika hingga menyelesaikan Laporan Tugas Akhir.