Pengenalan Karakter Huruf Rusia dengan Algoritma Perceptron

Michelle Zefanya Luhing[1], Kristien Margi Suryaningrum[2]

Technology and Design Department, Informatics Engineering, University of Bunda Mulia, Jakarta
Jalan Lodan Raya No.2, Ancol, Jakarta Utara
E-mail: m.zefanya.l@gmail.com

Abstract

Artificial Neural Network (ANN) is a branch science drom the fields of artificial intelligence (Artificial Intelligence) and metod which developed to solve problems by recognizing and clustering patterns. One of artificial neural network's application is in recognizing the Russian letter's (Cyrillic alphabet) pattern. Perceptron is one method of ANN which can be used to recognizing letter's pattern. In this study, before recognizing a letter pattern of image, the information about the letter's features will be extracted using pixel population matrix method. Form this study, Russian letter recognition gives percentage of success from sample data as big as 100% and percentage from testing data as big as 84,84%. System's ability in reconizing a Russian letter depends on trained sample data. Therefore, more sampel data with more kind of font trained then bigger the percentage of success in recognizing Russian letter out of the sample data.

Keywords: Artificial Neural Network, perceptron, Russian letter, pixel population matrix

Abstrak

Jaringan Syaraf Tiruan (JST) merupakan salah satu cabang ilmu kecerdasan buatan (*Artificial Intelligence*) dan merupakan metode yang dikembangkan untuk memecahkan masalah dengan cara mengenali dan mengelompokkan berbagai pola. Salah satu pengaplikasian jaringan syaraf tiruan ialah dalam mengenali pola huruf Rusia (alfabet *Silirik*). *Perceptron* ialah adalah salah satu metode JST yang dapat digunakan untuk mengenali pola huruf. Pada penelitian ini, sebelum mengenali pola citra suatu karakter huruf, informasi mengenai fitur suatu huruf akan diekstraksi menggunakan metode matriks populasi piksel. Dari penelitian yang dilakukan, pengenalan karakter huruf Rusia memberikan persentase keberhasilan terhadap data sampel sebesar 100% dan persentase terhadap data uji sebesar 84,84%. Kemampuan sistem dalam mengenali suatu karakter huruf Rusia sangat bergantung pada data sampel yang dilatih. Oleh karena itu, semakin banyak data sampel karakter huruf dengan berbagai macam font yang dilatih maka semakin besar pula persentase keberhasilan sistem dalam mengenali pola karakter huruf Rusia di luar data sampel.

Kata kunci: JST, *perceptron*, huruf Rusia, matriks populasi piksel

1. Pendahuluan

Manusia adalah makhluk yang tidak pernah berhenti mempelajari berbagai hal. Orang saling berlomba-lomba dalam mempelajari hal-hal baru dalam berbagai bidang, salah satunya ialah bahasa. Banyak orang berlomba-lomba untuk mempelajari bahasa-bahasa asing diluar bahasa ibu, seperti pada umumnya adalah bahasa Inggris, Jepang, Perancis, Jerman, Korea dan masih banyak lainnya. Salah satu bahasa asing lainnya yang dipelajari oleh orang-orang ialah bahasa Rusia. Bahkan di Indonesia sendiripun terdapat universitas yang mengadakan perkuliahan sastra Rusia, oleh karena itu orang-orang tertarik untuk mempelajari bahasa Rusia baik dengan mengikuti kursus ataupun belajar sendiri.

Biasanya dalam mempelajari bahasa asing dengan karakter huruf yang berbeda tentunya membutuhkan waktu dan haruslah terbiasa dengan pola penulisan setiap huruf yang ada. Meskipun penulisan beberapa karakter huruf Rusia (huruf *Silirik*) memiliki kemiripan dengan alfabet yang digunakan dalam bahasa Indonesia, karakter huruf Rusia tetap memiliki keunikannya tersendiri. Proses pembelajaran yang dilakukan inilah yang menjadi inspirasi untuk meneliti tentang pembelajaran pola huruf Rusia menggunakan jaringan syaraf tiruan *perceptron*.

Jaringan syaraf tiruan memiliki kemampuan untuk mendapatkan permasalahan yang sulit untuk didefinisikan, serta mempelajari data yang diberikan dari pengolahan terhadap inputan data yang ada. Berdasarkan hal tersebut penelitian ini merupakan lanjutan dari penelitian sebelumnya yang telah dilakukan orang menggunakan metode jaringan syaraf tiruan algoritma *perceptron* terhadap huruf *braille*, dimana penelitian ini merupakan perluasan ruang lingkup pengenalan karakter, diantaranya karakter kapital dari huruf Rusia yang akan dibahas.

2. Tinjauan Pustaka/Penelitian Sebelumnya

2.1 Alfabet Silirik

Alfabet *Silirik* dinamai berdasarkan penulis terkenal yang merupakan seorang biarawan pada abad ke-9 yaitu St.Cyril. Menggunakan alfabet *Silirik* untuk menulis bahasa Rusia adalah cara yang lebih mudah dari pada menerjemahkannya ke dalam penulisan dengan alfabet latin, karena dua atau lebih huruf latin dalam bahasa Inggris dibutuhkan untuk menulis satu huruf dalam bahasa Rusia [1].

Tabel 1. *Alfabet Silirik* [1]

No	Huruf	Transliterasi (dalam bahasa Inggris)
1	А	A dalam *father*
2	Б	B dalam *box*
3	В	V dalam *visit*
4	Г	G dalam *goat*
5	Д	D dalam *daughter*
6	Е	Ye dalam *yet*
7	Ё	Yo dalam *yonder*
8	Ж	S dalam *pleasure*
9	З	Z dalam *zoo*
10	И	Ee dalam *feet*
11	Й	Y dalam *boy*
12	К	K dalam *kit*
13	Л	L dalam *bottle*
14	М	M dalam *motor*
15	Н	N dalam *novel*
16	О	Aw/law dalam *bore*
17	П	P dalam *peach*
18	Р	R dalam *rat*
19	С	S dalam *sip*
20	Т	T dalam *tired*
21	У	Oo dalam *shoot*
22	Ф	F dalam *father*
23	Х	Ch dalam *loch*
24	Ц	Ts dalam *quits*
25	Ч	Ch dalam *chick*
26	Ш	Sh dalam *shift*
27	Щ	Shch dalam *posh china*
28	Ъ (hard sign)	“ (*silent*)
29	Ы	I dalam *ill*
30	Ь (soft sign)	‘ (*silent*)

31	Э	E dalam *let*
32	Ю	Yu dalam *yule*
33	Я	Ya dalam *yak*

2.2 Optical Character Recognition

Optical Character Recognition (OCR) adalah sebuah aplikasi komputer yang digunakan untuk mengidentifikasi citra huruf maupun angka untuk dikonversi ke dalam bentuk file tulisan [2].

2.3 Citra Digital

Secara harafiah, *image* atau citra merupakan gambar pada bidang dwimatra (dua dimensi). Sedangkan dilihat dari sudut pandang matematis, citra merupakan fungsi kontinu atau menerus dari intensitas cahaya pada bidang dwimatra. Citra terdiri dari dua macam jenis, yaitu citra kontinu dan citra diskrit. Citra kontinu berasal dari sistem optik yang menerima sinyal analog. Sedangkan citra diskrit berasal dari proses digitalisasi terhadap citra kontinu. Representasi citra dari fungsi kontinu menjadi nilai-nilai diskrit disebut sebagai digitalisasi. Citra yang dihasilkan inilah yang disebut citra digital [1].

2.4 Preprocessing

Pre-processing adalah tahap yang dimaksudkan untuk mendapatkan citra yang dianggap layak dianalisa oleh komputer. Karena penganalisa citra adalah komputer, maka citra yang memungkinkan untuk diproses harus berupa citra biner [3]. Pada tahap *pre-processing* dilakukan dengan beberapa proses yaitu binerisasi dan segmentasi.

2.5 Citra Biner

Citra biner adalah citra dimana piksel-pikselnya hanya memiliki dua buah nilai intensitas, bernilai 0 dan 1 dmana 0 menyatakan warna hitam dan 1 menyatakan warna putih. Di mana 0 didapat dari 1 titik dari rata-rata nilai citra RGB yang kurang dari 128 (sebagai nilai *threshold*) sebaliknya nilai 1 didapat dari 1 titik rata-rata nilai RGB yang lebih dari 128. Pengubahan pada citra biner ini bertujuan untuk mempermudah proses pengenalan karakter selanjutnya [4].

2.6 Segmentasi Citra

Segmentasi merupakan suatu proses pemisahan objek yang satu dengan objek yang lain dalam suatu citra, berdasarkan sifat-sifat tertentu dari citra yang dapat dijadikan sebagai pembeda [5]. Segmentasi pada penelitian ini hanya digunakan untuk memperoleh objek berupa huruf Rusia yang dipisahkan dari latar belakang atau diluar objek yang tidak diperlukan.

2.7 Matriks Populasi Piksel

Matriks populasi piksel adalah sekumpulan sel yang terbentuk dari pembagian citra-citra membentuk kolom dan baris. Masing-masing sel berisi populasi piksel pada bagian-bagian citra yang diwakilinya. Matriks populasi piksel dapat tersusun dari matriks 2x2, 3x3, 4x4, 5x5, 6x6 dan seterusnya. Matriks populasi piksel merupakan sebuah cara untuk mendapatkan fitur atau ekstrasi fitur dimana fitur yang didapatkan diproses untuk pengenalan karakter [3].

2.8 Fungsi Aktivasi

Fungsi aktivasi yang sering digunakan dalam arsitektur jaringan *perceptron* adalah Fungsi undak biner (*Hard Limiter*). Jaringan dengan lapisan tunggal sering menggunakan fungsi undak (*step function*) untuk mengkonversikan input dari suatu variabel yang bernilai kontinu ke suatu *output* biner (0 atau 1) [6]. Fungsi undak biner (*Hard Limiter*) dapat dilihat pada rumus 1.

$$y = \begin{cases} 0, jika\ x\ \leq 0 \\ 1, jika\ x > 0 \end{cases}$$

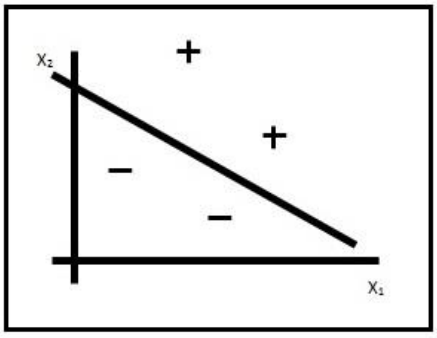
Dimana:
y : fungsi undak biner (*Hard Limiter*)
x : variabel masukkan (1)

2.9 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah paradigma pengolahan informasi yang terinspirasi oleh sistem syaraf secara biologis, seperti proses informasi pada otak manusia. Elemen kunci dari paradigma ini adalah struktur dari sistem pengolahan informasi yang terdiri dari sejumlah besar pemrosesan yang saling berhubungan (neuron), bekerja serentak untuk menyelesaikan masalah tertentu. Cara kerja jaringan syaraf tiruan seperti cara kerja manusia, yaitu bekerja melalui contoh. Sebuah jaringan syaraf tiruan dikonfigurasikan untuk aplikasi tertentu, seperti pengenalan pola atau klasifikasi data, melalui proses pembelajaran. Belajar dalam sistem biologis melibatkan penyesuaian terhadap koneksi *synaptic* yang ada antara neuron. Hal ini berlaku juga untuk jaringan syaraf tiruan [7].

2.10 Perceptron

Perceptron termasuk salah satu bentuk jaringan syaraf sederhana. *Perceptron* biasanya digunakan untuk mengklasifikasikan tipe pola tertentu yang sering dikenal dengan pemisahan secara linear. Pada dasarnya, *perceptron* pada jaringan syaraf dengan satu lapisan memiliki bobot yang bisa diatur dan suatu nilai ambang (*threshold)*. Algoritma yang digunakan oleh aturan *perceptron* ini akan mengatur parameter-parameter bebasnya melalui proses pembelajaran. *Perceptron* menggunakan fungsi aktivasi untuk memisahkan daerah positif dan negatif secara linear [6].



Gambar 1. *Pembatasan linear dengan perceptron*

Algoritma pembelajaran *perceptron* menggunakan metode dimana proses pembelajaran akan terus berlanjut sampai target *output* tercapai atau sudah tidak terjadi kesalahan. Namun apabila proses pembelajaran sudah dilakukan berulang-ulang namun masih terdapat kesalahan, maka proses pembelajaran harus dihentikan. Untuk menghitung hasil keluaran (*output*) dapat dilihat pada rumus 2.

$$Sij = \sum_{i=0}^{n} x_i \, w_{ij} + bias \quad (2)$$

Dimana:
i, j : data ke-n
s : hasil *ouput* pada data ke-n
x : input pada data ke-n
w : bobot data ke-n
bias : bias data ke-n

Dalam algoritma pembelajaran *perceptron*, perbaikan bobot tidak hanya dipengaruhi oleh nilai *input* dan *ouput* saja, tetapi dipengaruhi juga oleh laju pembelajaran. Nilai laju pembelajaran biasanya berada

diantara 0 sampai 1, semakin mendekati 1 maka proses pembelajaran akan semakin cepat. Sehingga rumus untuk memperbaiki bobot dapat dilihat pada rumus 3.
Selain bobot, laju pembelajaran juga mempengaruhi bias selama proses pembelajaran berlangsung. Rumus yang digunakan untuk memperbaiki bias dapat dilihat pada rumus 4.

$$w_{ij} = w_{ij} + \alpha x_i t_j \quad (3)$$

$$b_{ij} = b_{ij} + \alpha x_i t_j \quad (4)$$

Dimana:

i,j : data ke-n
w : bobot ke-ij
b : bias
α : laju pembelajaran
x : target *output* yang diinginkan
t : hasil *ouput* pada ij

3. Metodologi

Adapun metodologi yang digunakan dalam pengembangan sistem pada penelitian ini adalah model *waterfall*. Penggunaan model *waterfall* ini karena proses dilakukan secara bertahap, sehingga setiap proses tidak saling tumpang tindih dalam pelaksanaannya. Model *Waterfall* ini terdiri dari beberapa tahap, antara lain:

a. Analisis dan Definisi Persyaratan
   Informasi yang dapat digunakan untuk mendukung pengembangan perangkat lunak diperoleh melalui konsultasi dengan pengguna sistem. Pada tahapan ini diperoleh SRS (*Software Requirements Specification*) yang kemudian menjadi fungsionalitas dari perangkat lunak yang dibangun.
b. Perancangan Sistem dan Perangkat Lunak
   Proses perancangan sistem membagi persyaratan dalam sistem perangkat keras atau lunak. Kegiatan ini menentukan arsitektur sistem secara keseluruhan.
c. Implementasi dan Pengujian Unit
   Pada tahap ini perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Pengujian unit melibatkan verifikasi bahwa setiap unit telah memenuhi spesifikasinya.
d. Integrasi dan Pengujian
   Unit program atau program individual diintegrasikan dan diuji sebagai sistem yang lengkap untuk menjamin bahwa persyaratan sistem telah dipenuhi. Pengujian sistem menggunakan metode *blackbox* yaitu pengujian yang difokuskan pada dungsionalitas perangkat lunak tanpa pengetahuan struktur internal program (*source code*).
e. Operasi dan Pemeliharaan
   Tahap ini merupakan tahap siklus hidup yang paling lama. Pada tahap ini dilakukan instalasi dan penggunaan sistem. Pemeliharaan mencakup koreksi dari berbagai *error* yang tidak ditemukan pada tahap-tahap terdahulu, perbaikan atas implementasi unit sistem, dan pengembangan pelayanan sistem

4. Hasil dan Pembahasan

4.1 Implementasi

a. Proses Binerisasi

Proses binerisasi merupakan proses dimana citra digital diubah menjadi citra biner. Citra biner sendiri adalah citra dimana piksel-pikselnya hanya memiliki dua kemungkinan nilai yaitu 0 atau 1. Dalam mengubah citra digital menjadi citra biner, citra digital akan dikonversi terlebih dahulu menjadi citra *grayscale* dengan mengambil nilai dari tiga channel warna yaitu merah (*red)*, hijau (*green*) dan biru (*blue*) dari setiap piksel citra. Nilai dari masing-masing warna akan dimasukkan ke dalam variable R, G, B yang akan dijumlahkan dan dibagi 3 untuk menghasilkan citra *grayscale*, seperti yang sudah diimplementasikan dalam gambar 2.

```
int maxX=0, maxY=0, minX=lebar, minY=tinggi;
int treshold = 150;
for(int j=0; j<tinggi; j++){
   for(int i=0; i<lebar; i++){
      warna=pixels[j*lebar+i];
      alpha=(warna>>24)&0xff;
      red=(warna>>16)&0xff;
      green=(warna>>8)&0xff;
      blue=(warna)&0xff;
      abuabu=(red+green+blue)/3;

      if(abuabu <= treshold) {
            biner[i][j]=0;
            if(maxX<i) maxX=i;
            if(maxY<j) maxY=j;
            if(minX>i) minX=i;
            if(minY>j) minY=j;
      }else{
            biner[i][j]=1;
      }

      nilai = binerisasi(biner[i][j]);
      Color newColor = new Color(nilai, nilai, nilai);
      tampung.setRGB(i, j, newColor.getRGB());
   }
```

Gambar 2. *Proses menentukan nilai keabuan*

Citra *grayscale* akan disimpan dalam variabel penampung nilai gray. Jika nilai gray lebih besar dari nilai threshold yang ditentukan, maka piksel berwarna putih. Dan untuk sebaliknya maka piksel akan berwarna hitam.

b. Proses Ekstraksi Fitur

Proses ekstraksi fitur dilakukan setelah citra digital telah dibinerisasi dan disegmentasikan, sehingga objek dan background telah terpisah. Citra biner terlebih dahulu dicari nilai batas atas, batas bawah, batas kanan dan batas kirinya sepert yang sudah diterapkan dalam gambar 3.

```
for(int j=0; j<tinggi; j++){
   for(int i=0; i<lebar; i++){
      warna=pixels[j*lebar+i];
      alpha=(warna>>24)&0xff;
      red=(warna>>16)&0xff;
      green=(warna>>8)&0xff;
      blue=(warna)&0xff;
      abuabu=(red+green+blue)/3;

      if(abuabu <= treshold) {
            biner[i][j]=0;
            if(maxX<i) maxX=i;
            if(maxY<j) maxY=j;
            if(minX>i) minX=i;
            if(minY>j) minY=j;
      }else{
```

Gambar 3. *Pembatasan citra*

Selanjunya, citra huruf akan dibagi menjadi matriks 5x5. Untuk setiap sel matriksnya, jumlah piksel berwarna akan dihitung dan juga jumlah seluruh piksel pada satu sel tersebut akan dihitung. Untuk mendapatkan nilai fitur untuk setiap sel tersebut, bagi jumlah piksel hitam dengan jumlah seluruh piksel pada satu sel tersebut, seperti yang telah diterapkan pada gambar 4. Misalnya pada sel pertama matriks jumlah piksel hitam ada 50, sedangkan jumlah seluruh piksel pada sel pertama ialah 100. Maka bagi 50 dengan 100, dan didapatkan hasil 0,5 yang merupakan fitur dari sel.

```
public void matriksPopulasiPiksel(int l, int p, int[][] karakter){
   for(int j=0; j<5; j++){
      for(int i=0; i<5; i++){
         populasi[i][j]=0;
         jmlpsel[i][j] = 0;
      }
   }

   double rangeL = (float)(l)/5;
   double rangeP = (float)(p)/5;

   for(int j=0; j<5; j++){
      for(int i=0; i<5; i++){

         for(int y=((int) (Math.round(j*rangeL)));
               y<((int) Math.round((j+1)*rangeL));
               y++){
            for(int x=((int) Math.round(i*rangeP));
                  x<((int) Math.round((i+1)*rangeP));
                  x++){
               jmlpsel[i][j] += 1;
               if(karakter[x][y]==0){
                  populasi[i][j] += 1;
               }
            }
         }
         ppop[i][j] = (float)populasi[i][j]/ jmlpsel[i][j] ;
      }
   }
```

Gambar 4. *Matriks populasi piksel*

c. Proses Pembelajaran

Proses pembelajaran dimulai dengan inisialisasi bobot dan bias dengan nilai 0. Selanjutnya nilai kecepatan belajar diisi dengan 0.1 seperti yang ditunjukan pada gambar 5.

```
for (int i =0; i<25; i++ )
{
   ww[i] = 0;
}
bias = 0;
cc = (float) 0.1;
e = 1;
max = 0;
hrf = labelHruf.getText();
```

Gambar 5. *Proses inisialisasi awal variable sebelum proses pembelajaran*

Setelah itu, proses pembelajaran dimulai dengan mengalihkan fitur yang telah didapat dari proses ekstraksi fitur dengan bobot yang ada dengan rumus 1. Dalam mengekstraksi fitur sebelumnya digunakan matriks 5x5, sehingga terdapat 25 sel matriks yang mana setiap selnya merupakan fitur dari huruf. Setiap nilai fitur tersebut akan dialihkan ke variabel baru, kemudian dikalikan dengan bobot yang sudah disiapkan. Nilai dari sel matriks pertama akan dikalikan dengan bobot yang pertama, sel ke-2 dikalikan dengan bobot ke-2 dan seterusnya hingga sel ke-25 dan bobot ke-25. Hasil perhitungan kemudian dijumlahkan dan dibandingkan dengan target *output* yang telah ditentukan, implementasinya dapat dilihat pada gambar 6 dan 7.

```
        try{
            Connection c = dbs.connect();
            Statement s = c.createStatement();
            String sql = "SELECT * FROM tblftur ";
            ResultSet r = s.executeQuery(sql);
            int row = 0;

    while (e>0){
        e = 0;
        r.beforeFirst();
        while(r.next()){
            row = r.getRow();
            ss=0;
            b[0] = r.getFloat("sel 1");
            b[1] = r.getFloat("sel 2");
            b[2] =  r.getFloat("sel 3");
            b[3] = r.getFloat("sel 4");
            b[4] =  r.getFloat("sel 5");
            b[5] = r.getFloat("sel 6");
            b[6] = r.getFloat("sel 7");
            b[7] = r.getFloat("sel 8");
            b[8] = r.getFloat("sel 9");
            b[9] = r.getFloat("sel 10");
            b[10] = r.getFloat("sel 11");
            b[11] = r.getFloat("sel 12");
            b[12] = r.getFloat("sel 13");
            b[13] = r.getFloat("sel 14");
            b[14] = r.getFloat("sel 15");
            b[15] = r.getFloat("sel 16");
            b[16] = r.getFloat("sel 17");
            b[17] = r.getFloat("sel 18");
            b[18] = r.getFloat("sel 19");
            b[19] = r.getFloat("sel 20");
            b[20] = r.getFloat("sel 21");
            b[21] = r.getFloat("sel 22");
            b[22] = r.getFloat("sel 23");
            b[23] = r.getFloat("sel 24");
            b[24] = r.getFloat("sel 25");

            for (int i=0; i<25; i++)
                { ss = ss + (b[i]*ww[i]); }
            ss = ss+ bias;
            if (ss>0)
                { ss = 1; }
            else
                { ss = 0; }

            hrf2 = r.getString("id_hruf");
            if ((hrf).equals(hrf2) )
                { t = 1; }
            Else
                { t = 0; }
            x = t-ss;
```

Gambar 6. *Proses pembelajaran*

Apabila terdapat perbedaan (*error*) maka, bobot dan bias akan disesuaikan menggunakan rumus 2 (rumus perbaikan bobot), sedangkan bias disesuaikan menggunakan rumus 3 (rumus perbaikan bias). Setelah bobot dan bias diperbaiki, *output* akan dihitung kembali menggunakan rumus 1. Apabila dengan nilai bobot dan bias yang telah diperbaiki masih memberikan *ouput* yang tidak sesuai dengan target, maka proses akan terus diulang sampai target *ouput* tercapai (tidak terjadi error) atau pengulangan telah mencapai nilai maksimal.

```
while (x != 0){
   ss = 0;
   e++;
   //perbaiki bobot
   for (int i = 0; i<25; i++)
   { ww[i]= (float) ww[i]+(cc*x*b[i]) ; }

   //perbaiki bias
   bias = bias + (cc*x);

   for (int i=0; i<25; i++)
   { ss = ss + (b[i]*ww[i]); }
   ss = ss+ bias;
   if (ss>0)
   { ss = 1; }
   else
   { ss = 0; }

   x = t-ss;
}
```

Gambar 7. *Perbaikan bobot dan bias*

Bila hasil *output* sesuai dengan target, maka bobot tidak akan diperbaiki. Proses pembelajaran akan dianggap selesai apabila hasil *output* scsuai dengan target *output*, atau bila pengulangan yang dilakukan telah mencapai batas maksimal. Setelah proses pembelajaran selesai, nilai bobot dan bias akan disimpan.

d. Proses Pengenalan Karakter

Proses pengenalan karakter dilakukan dengan menguji apakah bobot yang didapat dari hasil pembelajaran mampu digunakan untuk mengenali karakter. Proses dalam pengenalan karakter hampir sama dengan proses yang dilakukan dalam pembelajaran karakter. Perbedaannya terletak pada proses pengujian atau pengenalan karakter, di mana tidak ada tahap perbaikan bobot dalam proses pengujian.

```
while(r.next()){
   ss=0;
   idhruf = r.getString("id_hruf");

   System.out.println(idhruf);

   ww[0][0] = r.getFloat("bbt1");
   ww[0][1] = r.getFloat("bbt2");
   ww[0][2] =  r.getFloat("bbt3");
   ww[0][3] = r.getFloat("bbt4");
   ww[0][4] = r.getFloat("bbt5");
   ww[1][0] =  r.getFloat("bbt6");
   ww[1][1] = r.getFloat("bbt7");
   ww[1][2] = r.getFloat("bbt8");
   ww[1][3] = r.getFloat("bbt9");
   ww[1][4] = r.getFloat("bbt10");
```

Gambar 8. *Proses pengenalan karakter (1)*

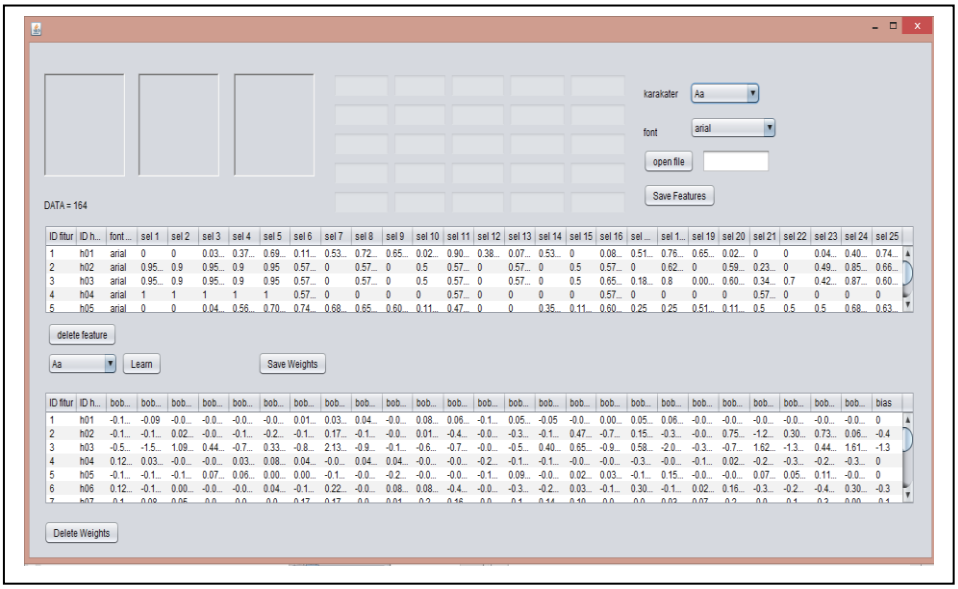
```
ww[2][0] = r.getFloat("bbt11");
ww[2][1] = r.getFloat("bbt12");
ww[2][2] = r.getFloat("bbt13");
ww[2][3] = r.getFloat("bbt14");
ww[2][4] = r.getFloat("bbt15");
ww[3][0] = r.getFloat("bbt16");
ww[3][1] = r.getFloat("bbt17");
ww[3][2] = r.getFloat("bbt18");
ww[3][3] = r.getFloat("bbt19");
ww[3][4] = r.getFloat("bbt20");
ww[4][0] = r.getFloat("bbt21");
ww[4][1] = r.getFloat("bbt22");
ww[4][2] = r.getFloat("bbt23");
ww[4][3] = r.getFloat("bbt24");
ww[4][4] = r.getFloat("bbt25");

bias = r.getFloat("bias");

for(int i=0; i<5; i++){
   for(int j=0; j<5; j++){
      ss = ss + ppop[i][j]*ww[i][j];
   }
}
ss = ss +bias;
```

Gambar 10. *Proses pengenalan karakter (2)*

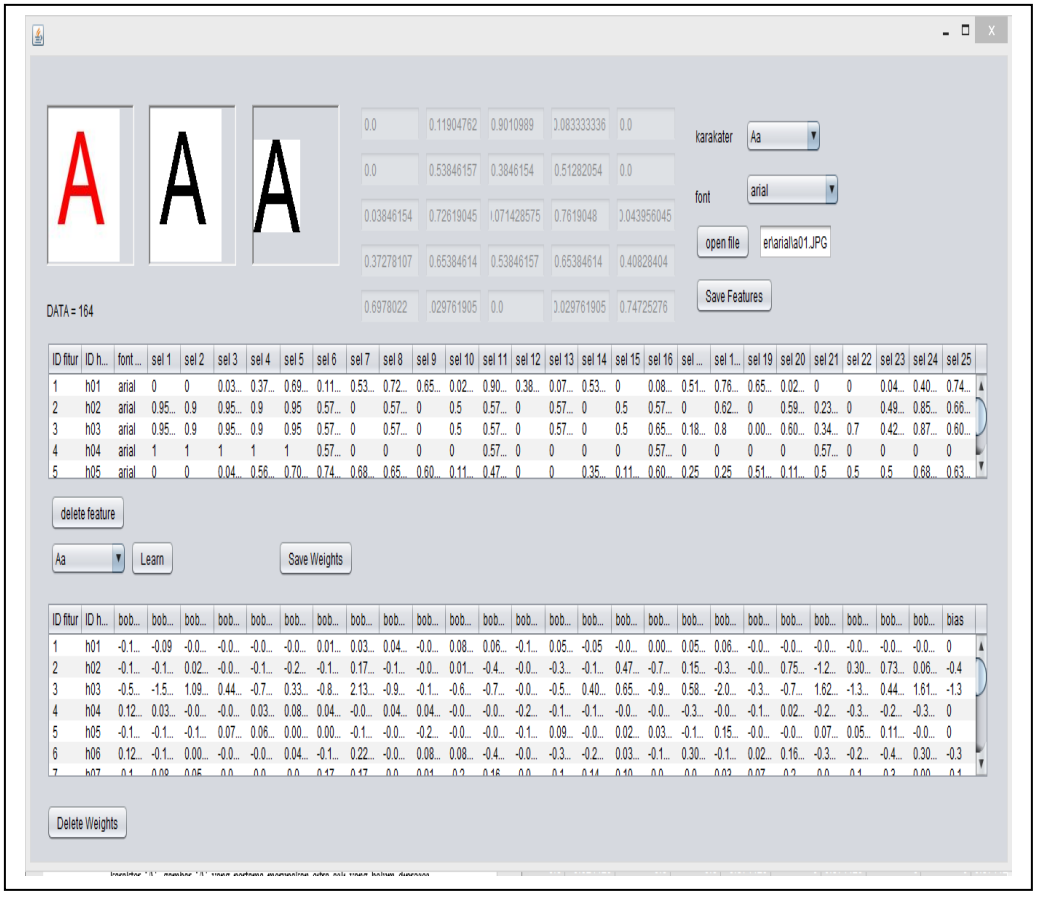
Nilai *input* berupa nilai fitur dari citra uji akan dimasukkan ke dalam rumus 1 untuk menghasilkan sebuah *output*. Bila *output* yang dihasilkan memenuhi batas ambang yang ada, maka *output* yang didapat adalah karakter tersebut. Misalkan, nilai input dikalikan dengan bobot huruf ‘Б’ dan *output* yang dihasilkan lebih besar dari 0. Maka, input tersebut dianggap huruf 'Б'. Bila *output* dari suatu huruf lebih kecil dari 0, maka nilai *input* akan dikalikan dengan bobot huruf selanjutnya

e. Implementasi Antarmuka

Pada aplikasi pengenal huruf Rusia terdapat dua buah *form. Form* pertama merupakan *form training* untuk pembelajaran/pelatihan huruf Rusia dan *form* kedua bernama *recognition* yang digunakan untuk mengenali huruf Rusia pada file citra.

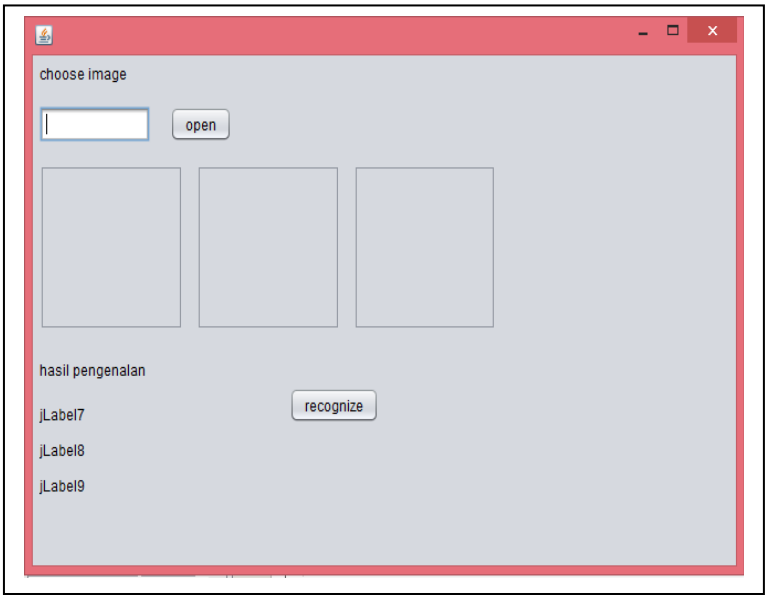


Gambar 11. *Antarmuka form training* (1)

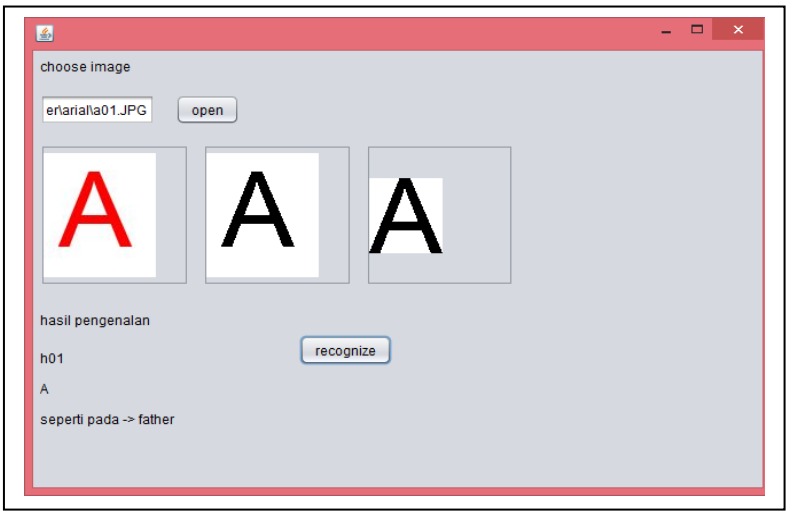


Gambar 11. *Antarmuka form training* (2)

Tampilan antarmuka proses *training* terdiri atas dua bagian, yaitu bagian untuk ekstraksi fitur dan bagian *training*. Bagian ekstraksi fitur terletak di sebelah atas *form*, dan bagian pembelajaran untuk menghasilkan bobot pengenalan terletak di bagian bawah *form*. Sedangkan tiga buah *image* di bagian kiri atas digunakan untuk menampilkan citra yang diproses.



Gambar 12. *Antar muka form recognition* (1)

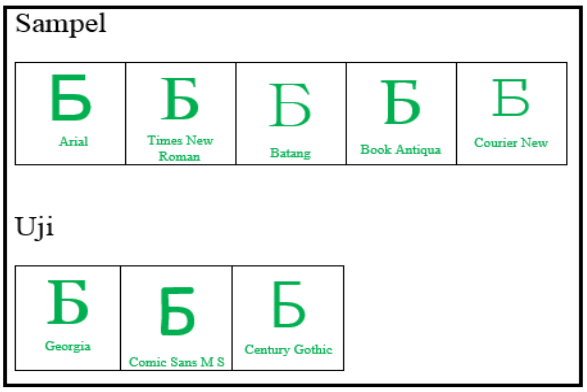


Gambar 13. *Antarmuka form recognition* (2)

Tampilan antarmuka dari *form recognition* memiliki tampilan yang lebih sederhana dibandingkan *form training*, karena form ini hanya digunakan untuk mengenali huruf dalam citra yang di-*input*-kan

4.2 Hasil Pengujian

Untuk memastikan bahwa sistem yang dibuat mampu berjalan dengan baik dan bekerja sesuai dengan fungsinya, pengujian perlu dilakukan. Pengujian yang dilakukan ada dua, yaitu pengujian keberhasilan belajar terhadap huruf yang dipelajari dan pengujian akurasi terhadap huruf lain yang mirip dengan huruf yang dipelajari. Pengujian sistem pengenal huruf Rusia terhadap *font* huruf lain dilakukan dengan menggunakan data uji berupa citra berisi karakter kapital dari huruf Rusia yang menggunakan *font Georgia, Comic Sans M S* dan *Century Gothic*. Data sampel yang terdapat dalam database sendiri berjumlah 165 data yang merupakan 33 karakter kapital huruf Rusia menggunakan 5 *font* berbeda yaitu *font arial, Times New Roman, Batang, Book Antiqua* dan *Courier New.*



Gambar 14. *Font data sampel dan data uji*

Berdasarkan pengujian yang telah dilakukan terhadap data sampel, sistem berhasil mengenali seluruh huruf Rusia sebanyak 165 data dengan benar sehingga diperoleh persentase keberhasilan terhadap data sampel sebesar 100%. Sedangkan berdasarkan pengujian yang telah dilakukan terhadap data uji, sistem berhasil mengenali huruf Rusia sebanyak 84 dari 99 data dengan benar sehingga diperoleh persentase keberhasilan terhadap data uji sebesar 84,84%. Penyebab kegagalan sistem mengenali huruf Rusia adalah karena pengenalan huruf sangat bergantung dengan data sampel yang dimiliki. Beberapa karakter dari data uji tidak dapat dikenali dikarenakan adanya perbedaaan yang mencolok dibandingkan dengan huruf sampel dalam proses pelatihan. Di mana huruf tersebut tentunya akan dianggap berbeda dan tidak dikenali atau akan dikenali sebagai karakter lain.

Tabel 2. *Hasil Pengujian*

	Georgia	Comic Sans M S	Century Gothic	Total
Jumlah	33	33	33	99
Benar	33	27	24	84
Salah	0	6	9	15
%	100%	81,81%	72,72%	84,84%

5. Kesimpulan

5.1 Simpulan

Berdasarkan penelitian yang telah dilakukan, berikut kesimpulan yang dapat ditarik dari penelitian ini:

1. Aplikasi pengenal huruf Rusia yang dibangun mampu mengenali karakter dengan *input* berupa citra digital dan mampu memberikan informasi mengenai huruf Rusia yang dimasukkan seperti keterangan pelafalan huruf beserta contohnya.
2. *Perceptron* dan ekstraksi fitur menggunakan matriks populasi piksel dapat diimplementasikan pada aplikasi pengenal huruf Rusia dan menghasilkan persentase keberhasilan yang cukup tinggi sebesar 84.84%.
3. Kegagalan sistem dalam mengenali huruf Rusia dipengaruhi oleh banyaknya data sampel yang dimiliki dalam sistem. Oleh karena itu, untuk meningkatkan keberhasilan sistem, penambahan data sampel berupa huruf dengan *font* yang beragam merupakan solusinya.

5.2 Saran

Dari penelitian ini terdapat pula beberapa saran yang sekiranya dapat berguna untuk penelitian selanjutnya :

1. Penelitian ini dikembangkan sehingga dapat mengenali huruf lain selain huruf Rusia.
2. Penelitian ini dikembangkan sehingga dapat mengenali lebih dari satu karakter dalam satu *file* citra.
3. Untuk penelitian selanjutnya dapat dicoba dengan menggunakan algoritma pembelajaran yang lain.

6. Daftar Rujukan

[1] West, D. (2017). *Complete Russian Beginner to Intermediate Course: Enhanced Edition.* Hachetter: United Kingdom.

[2] Hartanto, S, Sugiharto, A, dan Endah, S. (2015). *Character Recognition Menggunakan Algoritma Template Matching Correlation*. Jurnal Masyarakat Informatika. Vol. 9. p.1-12. ISSN: 2477-5894.

[3] Mulyana, Teady Matius (2014). *Fitur Matriks Populasi Piksel Untuk Membedakan Frame-frame Dalam Deteksi Gerakan*. Jurnal Teknologi Informasi Universitas Bunda Mulia. Volume 10 Nomor 1. p 14-18.

[4] Yuwatining, Eka Farda., Hidayat, Bambang., & Andini, Nur. (2014). *Implementasi Metode Hidden Markov Model Untuk Deteksi Tulisan Tangan*. Jurnal E-Proceeding of Engineering.Volume 1 Nomor 1. p.396-401. ISSN: 2355-9365.

[5] Septiarini, A. (2012). *Segmentasi Karakter Menggunakan Profil Proyeksi.* Jurnal Informatika Mulawarman Universitas Mulawarman. Volume 7 Nomor 2. Juli 2012. p 66-69.

[6] Hafizah, Sulindawaty & Tugiono. (2015). *Penerapan Jaringan Syaraf Tiruan Dengan Algoritma Perceptron untuk Mendeteksi Karakteristik Sidik Jari.* Jurnal Saintikom. Volume 14 Nomor 2. ISSN:1978-6603.

[7] Simbolon, Regina. (2013). *Perangkat Lunak Untuk Identifikasi dan Pengenalan Huruf Braille Dengan Algoritma Perceptron*. Jurnal Pelita Informatika Budi Darma. Volume 4 Nomor 2. ISSN: 2301-9425.