Pengembangan Aplikasi Pengenalan Karaketer Alfanumerik Dengan Menggunakan Algoritma *Neural Network Three-Layer Backpropagation*

Andi Wahju Rahardjo Emanuel, Arie Hartono

Jurusan S1 Teknik Informatika
Fakultas Teknologi Informasi, Universitas Kristen Maranatha
Jl. Prof. Drg. Suria Sumantri No. 65 Bandung 40164
Email: andi.wre@eng.maranatha.edu

Abstract

This paper demonstrates the use of neural network's three-layer backpropagation algorithms written in C# programming language for recognizing alphanumeric characters which is written by hand in canvas or inputted by user in pictorial format. Better accuracy in recognizing alphanumeric characters is shown compared to single layer implementation with trade-off with the training times which will take much longer time to complete. The result shows some promising potential of the use of this algorithms combined with other algorithms to increase the accuracy of the detection in wider ranges of applications.

Keywords : Neural Network, Back Propagation, Forward Propagation, Alphanumeric, Digital Image Processing, Alphanumeric Character Recognition

Pendahuluan

Pengolahan citra digital dewasa ini tidak hanya berkisar antara pengeditan citra digital dengan menggunakan filter-filter efek yang ada, namun juga meliputi teknik pengenalan karakter seperti karakter alfanumerik, karakter tulisan tangan, karakter huruf arab, karakter huruf kanji, dan lain-lain. Teknik pengenalan karakter ini sering disebut secara umum sebagai teknologi OCR (*Optical Character Recognition*). Teknologi ini bukanlah hal baru dalam ruang lingkup teknologi informasi. Teknologi OCR ini banyak ditawarkan dalam produk-produk *scanner* pada masa terkini. Selain pada produk *scanner*, teknologi OCR juga terdapat pada *Handphone*, *Smart Phone*, dan PDA (*Personal Digital Assistant*) yang sudah mengimplementasikan teknologi layar sentuh dan memiliki fitur *handwriting recognition.*

Salah satu teknik pengenalan citra adalah dengan menerapkan algoritma runut balik (*back propagation)* yang terdapat dalam jaringan syaraf tiruan (*neural network*). Pada tulisan ini dibahas teknik pengenalan karakter alfanumerik dengan menggunakan suatu algoritma tersebut. Penekanan algoritma pada pengenalan pola karakter dari suatu objek di dalam citra. Algoritma yang digunakan dapat mengenali karakter tulisan alfanumerik pada suatu citra digital / gambar dan juga mengenali pola tulisan tangan alfanumerik. Beberapa fitur dari aplikasi ini yang akan dibahas adalah :

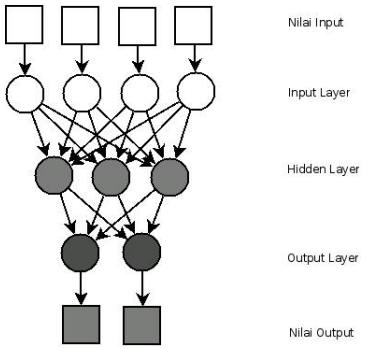
1. Mengenali karakter tulisan tangan alfanumerik.

2. Mengenali karakter alfanumerik dari suatu gambar
3. Membandingkan karakter alfanumerik inputan dengan karakter alfanumerik yang sudah ditentukan untuk mencari tingkat presentase kecocokan.

2. Dasar Teori

Jaringan Syaraf Tiruan

Aplikasi pengenalan karakter alfanumerik ini merupakan suatu implementasi dari algoritma *Neural Network. Neural Network* atau jaringan saraf tiruan adalah jaringan dari sekelompok unit pemroses kecil yang dimodelkan berdasarkan jaringan saraf manusia. Untuk *Neural Network Three Layer* secara garis besar dapat dilihat dari gambar di bawah ini.



Gambar 1. Rumus *Neural Network Three Layer*

Nilai – nilai input akan diberikan suatu bobot yang akan dimasukkan melewati beberapa lapisan *neuron* (*neuron layers*) yaitu lapisan *input layer*, *hidden layer*, dan *output layer*. Masing – masing lingkaran di setiap lapisan adalah sebuah *neuron* yang aktivasinya dikontrol oleh sebuah "fungsi aktivasi" yang menentukan apakah nilai input beserta bobotnya akan diteruskan atau tidak ke lapisan berikutnya. Fungsi aktifasi yang dipilih dan yang paling sesuai untuk algoritma runut balik (*back propagation*) adalah fungsi *sigmoid* sebagai berikut:

$$y=g(x)=\frac{1}{1+e^{-x}} \quad \text{.....(1)}$$

Dimana :

$$x=bias_0+\sum_{i=1}^{n} w^i x^i \quad \text{.....(2)}$$

$bias_0$ = bias awal (biasanya nol)
w_i = bobot untuk neuron i
x_i = nilai input untuk neuron i
n = jumlah neuron pada layer sebelumnya

- **Pembelajaran Back Propagation**

Proses penentuan nilai – nilai bobot yang berpasangan dengan masing – masing *neuron* di setiap lapisan (*layers*) dinamakan proses pembelajaran atau *training*. Adapun proses yang dilakukan adalah sebagai berikut:

1. Melakukan inisialisasi bobot dengan suatu nilai awal yang cukup kecil atau bahkan bisa nol.
2. Runut maju (*forward propagation*) – seperti yang dijelaskan di bagian 2.1
3. Runut mundur (*backward propagation*)

Proses inisialisasi adalah proses untuk memberikan nilai – nilai awal pada bobot yang bisa berupa nilai 0 atau 1.

Sedangkan untuk proses runut mundur (*back propagation*) dilakukan mulai dari *output layer*, kemudian merambat ke *hidden layer* dan akhirnya ke *input layer*. Caranya dengan menghitung informasi error dan kemudian informasi tersebut dipergunakan untuk memperbaiki bobotnya dengan rumus:

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

Dimana:

$$\Delta w_{jk} = \alpha \delta_k z_j \quad \text{.....(4)}$$

$$\delta_k = (t_k - y_k) f'(y_{ink}) \quad \text{.....(5)}$$

α = kecepatan pembelajaran (biasanya 1 atau 2)

$f'(y_{ink})$ = fungsi turunan dari sigmoid

t_k = target pola

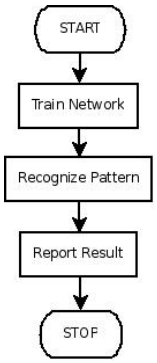
y_k = nilai output hasil forward propagation

z_k = nilai dari layer sebelumnya

Proses diatas diulangi ke tingkat *hidden layer* dan akhirnya ke *input layer* dan nilai bobot sampai mencapai tingkat pengulangan tertentu atau mencapai tingkat kesalahan (*error*) yang dikehendaki.

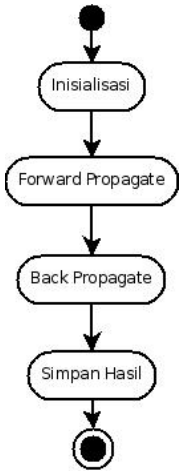
1. Desain Perangkat Lunak

Diagram alur dari aplikasi pengenalan alfanumerik ini ditunjukkan pada gambar berikut:



Gambar 4. Diagram Alur Aplikasi

Diantara ketiga bagian alur diatas yang merupakan bagian yang paling penting adalah bagian "*Train Network*" yang merupakan inti dari algoritma *Neural Network*. Diagram aktifitas yang menunjukkan detil aktifitas "*Train Network*" khususnya untuk algoritma runut balik (*backward propagation*) ditunjukkan dalam gambar berikut:



Gambar 5. Diagram Aktifitas Train Network

Kode sumber untuk inisialisasi adalah sebagai berikut:

```
private void InitializeNetwork()   {
        int i, j;
        Random rand = new Random();
        for (i = 0; i < PreInputNum; i++)   {
           PreInputLayer[i].Weights = new double[InputNum];
           for (j = 0; j < InputNum; j++)
              PreInputLayer[i].Weights[j] = 0.01 + ((double)rand.Next(0, 8) / 100);
         }
        for (i = 0; i < InputNum; i++)  {
           InputLayer[i].Weights = new double[HiddenNum];
           for (j = 0; j < HiddenNum; j++)
              InputLayer[i].Weights[j] = 0.01 + ((double)rand.Next(0, 8) / 100);
        }
```

```
        for (i = 0; i < HiddenNum; i++)   {
            HiddenLayer[i].Weights = new double[OutputNum];
            for (j = 0; j < OutputNum; j++)
                HiddenLayer[i].Weights[j] = 0.01 + ((double)rand.Next(0, 8) / 100);
        }
}
```

Kode sumber untuk proses training adalah sebagai berikut:

```
public void Train(ArrayList TrainingInputs, ArrayList TrainingOutputs, double MaxError)  {
        foreach (string s in TrainingOutputs)    {
            if (!OutputSet.Contains(s))
                OutputSet.Add(s);
        }
        stopTraining = false;
        do  {
            for (int i = 0; i < TrainingInputs.Count; i++)     {
                ForwardPropagate((double[])TrainingInputs[i], (string)TrainingOutputs[i]);
                BackPropagate();
            }
            currentError = GetTotalError(TrainingInputs, TrainingOutputs);
        } while (currentError > MaxError && !stopTraining);
      }
```

Dalam realisasinya di kode sumber C# metode *ForwardPropagate* untuk tiga layer dituliskan sebagai berikut:

```
private void ForwardPropagate(double[] pattern, string output)   {
        int i, j;
        double total;
        //Masukan Input ke Network
        for (i = 0; i < PreInputNum; i++)   {
            PreInputLayer[i].Value = pattern[i];
        }
        //Kalkulasi 1st untuk (Input)Layer yang terdiri dari Inputs dan Outputs
        for (i = 0; i < InputNum; i++)    {
            total = 0.0;
            for (j = 0; j < PreInputNum; j++)
                total += PreInputLayer[j].Value * PreInputLayer[j].Weights[i];
            InputLayer[i].InputSum = total;
            InputLayer[i].Output = F(total);
        }
```

```
        //Kalkulasi 2nd untuk (Hidden)Layer yang terdiri dari Inputs dan Outputs
        for (i = 0; i < HiddenNum; i++)    {
           total = 0.0;
           for (j = 0; j < InputNum; j++)
              total += InputLayer[j].Output * InputLayer[j].Weights[i];
           HiddenLayer[i].InputSum = total;
           HiddenLayer[i].Output = F(total);
        }
        //Kalkulasi 3rd untuk (Output)Layer yang terdiri dari Inputs, Outputs, Targets dan Errors
        for (i = 0; i < OutputNum; i++)    {
           total = 0.0;
           for (j = 0; j < HiddenNum; j++)
              total += HiddenLayer[j].Output * HiddenLayer[j].Weights[i];
           OutputLayer[i].InputSum = total;
           OutputLayer[i].output = F(total);
           OutputLayer[i].Target = (((string)OutputSet[i]) == output ? 1.0 : 0.0);
           OutputLayer[i].Error = (OutputLayer[i].Target - OutputLayer[i].output) *
                                  (OutputLayer[i].output) * (1 - OutputLayer[i].output);
        }
}
```

Sedangkan untuk metode *BackPropagate* untuk tiga layer direalisasikan menjadi kode sumber C# sebagai berikut:

```
private void BackPropagate()   {
        int i, j;
        double total;
        //Fix Error untuk Layer Hidden
        for (i = 0; i < HiddenNum; i++)    {
           total = 0.0;
           for (j = 0; j < OutputNum; j++)
               total += HiddenLayer[i].Weights[j] * OutputLayer[j].Error;
           HiddenLayer[i].Error = total;
        }
        //Fix Error untuk Layer Input
        for (i = 0; i < InputNum; i++)    {
           total = 0.0;
           for (j = 0; j < HiddenNum; j++)
               total += InputLayer[i].Weights[j] * HiddenLayer[j].Error;
           InputLayer[i].Error = total;
        }
        //Update Weights Layer pertama
        for (i = 0; i < InputNum; i++)    {
           for (j = 0; j < PreInputNum; j++)    {
```

```
                PreInputLayer[j].Weights[i] += LearningRate * InputLayer[i].Error *
                                                PreInputLayer[j].Value;
        }
        //Update Weights Layer kedua
        for (i = 0; i < HiddenNum; i++)    {
           for (j = 0; j < InputNum; j++)
              InputLayer[j].Weights[i] += LearningRate * HiddenLayer[i].Error *
                                            InputLayer[j].Output;
        }
        //Update Weights Layer ketiga
        for (i = 0; i < OutputNum; i++)    {
           for (j = 0; j < HiddenNum; j++)
               HiddenLayer[j].Weights[i] += LearningRate * OutputLayer[i].Error *
                                             HiddenLayer[j].Output;
        }
}
```

Kode sumber untuk proses *Recognize* adalah sebagai berikut:

```
public void Recognize(double[] pattern, ref string MatchedHigh, ref double OutputValueHight,
                                ref string MatchedLow, ref double OutputValueLow)   {
        int i, j;
        double total = 0.0;
        double max = -1;
        //Masukan Input ke Network
        for (i = 0; i < PreInputNum; i++)
             PreInputLayer[i].Value = pattern[i];
        //Kalkulasi Layer Input untuk Inputs dan Outputs
        for (i = 0; i < InputNum; i++)    {
           total = 0.0;
           for (j = 0; j < PreInputNum; j++)    {
              total += PreInputLayer[j].Value * PreInputLayer[j].Weights[i];
           }
           InputLayer[i].InputSum = total;
           InputLayer[i].Output = F(total);
        }
        //Kalkulasi Layer Hidden untuk Inputs dan Outputs
        for (i = 0; i < HiddenNum; i++)   {
           total = 0.0;
           for (j = 0; j < InputNum; j++)    {
              total += InputLayer[j].Output * InputLayer[j].Weights[i];
           }
```

```
            HiddenLayer[i].InputSum = total;
            HiddenLayer[i].Output = F(total);
        }
        //Cari 2 Output tertinggi
        for (i = 0; i < OutputNum; i++)    {
            total = 0.0;
            for (j = 0; j < HiddenNum; j++)     {
                total += HiddenLayer[j].Output * HiddenLayer[j].Weights[i];
            }
            OutputLayer[i].InputSum = total;
            OutputLayer[i].output = F(total);
            if (OutputLayer[i].output > max)    {
                MatchedLow = MatchedHigh;
                OutputValueLow = max;
                max = OutputLayer[i].output;
                MatchedHigh = (string)OutputSet[i];
                OutputValueHight = max;
            }
        }
}
```

Fungsi aktivasi berupa fungsi sigmoid direalisasikan sebagai berikut:

```
private double F(double x) { return (1 / (1 + Math.Exp(-x))); }
```

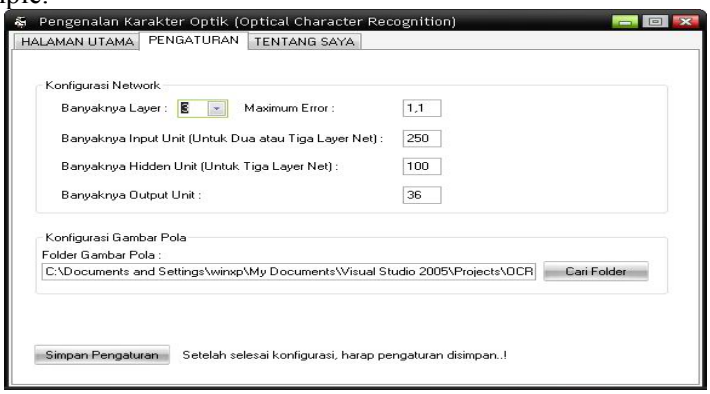
Sedangkan fungsi untuk mencari nilai error yang akan diumpanbalikan dalam proses backpropagate adalah sebagai berikut:

```
public double GetTotalError(ArrayList TrainingInputs, ArrayList TrainingOutputs)    {
        double total = 0.0;
        for (int i = 0; i < TrainingInputs.Count; i++)     {
            ForwardPropagate((double[])TrainingInputs[i], (string)TrainingOutputs[i]);
            for (int j = 0; j < OutputNum; j++)
                    total += Math.Pow((OutputLayer[j].Target - OutputLayer[j].output), 2) / 2;
        }
        return total;
}
```

2. Realisasi Aplikasi

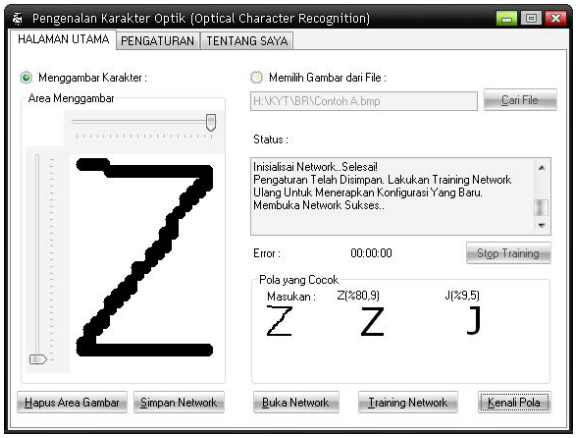
Pada realisasinya dalam aplikasi, terdapat 2 buah form utama yaitu Main Form dan Form Pengaturan. Form Pengaturan dipergunakan untuk memilih parameter – parameter yang dibutuhkan untuk aplikasi yaitu banyaknya layer, maksimum error,

banyaknya *input* unit, banyaknya *hidden* unit dan lain – lain. Pada menu pengaturan terdapat pilihan untuk melakukan pilihan banyaknya layer yang akan digunakan, nilai pada masing-masing layer dan juga untuk menentukan lokasi file gambar sample.



Gambar 6. Form Pengaturan Aplikasi

Sedangkan MainForm merupakan tampilan utama dimana pengguna dapat melakukan fungsi – fungsi utama seperti *Train Network*, *Recognize Pattern* dan *Report Result*. Pada mainform terdapat pilihan untuk melakukan pengenalan karakter alfanumerik dengan cara menggambar pada area gambar, ataupun juga dengan cara input gambar dari file.



Gambar 7. Main Form Aplikasi

3. Observasi dan Evaluasi

Dari hasil penelitian dan evaluasi menggunakan aplikasi pengenalan karakter alfanumerik ini dapat ditarik kesimpulan yaitu :

1. Aplikasi pengenalan karakter alfanumerik ini dapat mengenali karakter tulisan tangan yang digambar oleh pengguna pada area gambar ataupun file gambar yang diinputkan ke aplikasi.

2. Aplikasi pengenalan karakter alfanumerik ini dapat memberikan gambar hasil yang sesuai dengan karakter gambar yang dimasukan oleh pengguna beserta dengan tingkat presentase kecocokannya.
3. Pada pengaturan layer satu, waktu yang dibutuhkan sistem untuk melakukan training terbilang cepat. Tetapi memiliki kelemahan dalam tingkat keakurasian dalam mengenali pola karakter. Sedangkan pada pengaturan layer dua dan layer tiga, waktu yang dibutuhkan sistem untuk melakukan training terbilang relatif lama. Tetapi memiliki keunggulan dalam tingkat keakurasian dalam mengenali pola karakter.
4. Gambar dari hasil tulisan tangan yang buruk atau file gambar yang kurang baik akan berpengaruh pada hasil yang tidak diinginkan atau memiliki tingkat presentase kemiripan yang rendah.
5. Bila font gambar sample sama dengan font gambar masukan, maka akan menghasilkan hasil yang maksimal. Font gambar masukan yang mirip dengan font gambar sample, masih dapat memberikan hasil sesuai dengan yang diharapkan.

4. Kesimpulan

Pada tulisan ini telah didemonstrasikan penerapan algorithma *neural network back-propagation* (runut-balik) untuk mengenali karakter alfanumerik (huruf dan angka). Semakin tinggi jumlah layer yang dipakai akan semakin meningkatkan tingkat akurasi pengenalan karakter alfanumerik yang diinginkan, namun dengan pengorbanan di lamanya waktu *training* yang bertambah sangat signifikan.

Salah satu contoh kemungkinan penerapan aplikasi ini misalnya dipergunakan untuk mengenali plat nomor kendaraan secara otomatis dari hasil tangkapan kamera digital yang dipasang di lapangan parkir di pusat perbelanjaan. Pengembangan aplikasi ini lebih lanjut memungkinkan untuk mengkombinasikan dengan algorithma yang lain dan juga proses pengolahan citra yang baik sehingga tingkat akurasinya bisa lebih ditingkatkan lagi.

5. Daftar Pustaka

[Luk07] Lukman. (2007). Studi Sistem Adaptif dan Contoh Aplikasinya. Diakses 25 Oktober 2007. http://www.ilmukomputer.com/sistem-adaptif

[___07] ___.(2007). ION-528 - Image Processing Algorithms. Retrieved April 4, 2007, from http://www.ii.metu.edu.tr/~ion528/demo/lectures/6/1/index.html

[Bar06] Barber, S. (2006). AI_Neural Network for beginners, Prentice Hall.

[Mad06] Madhusudanan, A. (2006). Brainnet - A Neural Netwok Project - With Illustration, http://amazedsaint.blogspot.com

[Hai05] Haidar, A. (2005). Studi Kasus Mengenai Aplikasi Multilayer Perceptron Neural Netwok Pada Sistem Pendeteksi Gangguan (IDS) Berdasarkan Anomali Suatu Jaringan. Diakses 25 Oktober 2007. http://www.ilmukomputer.com/andry-report

[Rin04] Rinaldi, M. (2004). Pengolahan Citra Digital dengan Pendekatan Algoritmik, Informatika.

[Nug03] Nugroho, S. (2003). Pengantar Softkomputing. Diakses 25 Oktober 2007. http://asnugroho.net/pengantar-softkomputing

[Kus03] Kusumadewi, S. (2003). Artificial Intelligence (Teknik dan Aplikasinya). 1 Januari 2003. Penerbit Graha Ilmu Yogyakarta

[Mat02] Matthew, M. (2002). .Net User Interface with C#: Windows Forms and Custom Controls, Apress.

[Dei02] Deitel. (2002). C# How To Program, Prentice Hall.