BAB II

LANDASAN TEORI

2.1 Optical Character Recognition

Optical Character Recognition (OCR) adalah sebuah teknologi yang membuat mesin dapat mengenali karakter secara otomatis melalui mekanisme optik. Menurut Sameeksha Barve (2012), "mekanisme optik" merupakan mekanisme yang digunakan manusia dalam mengenali objek melalui mata. Secara psikologi manusia melihat sebuah karakter berdasarkan bentuk keseluruhan dan feature yang ada, seperti goresan, lengkungan, tonjolan, serta pembatasnya.

Berdasarkan penelitian Shrivastava dan Sharma (2012), langkah-langkah yang dilakukan OCR dibagi menjadi:

a. Pre-processing

Setiap citra butuh tahap *pre-processing* sebelum dilakukan pengenalan oleh sistem. Langkah pertama adalah mengkonversi citra menjadi sebuah citra biner (hanya memiliki nilai *pixel* berupa "0" dan "1"). Hasil konversi tersebut membuat citra menjadi citra dengan teks berwarna hitam di atas latar berwarna putih.

Langkah kedua adalah segmentasi yang terjadi dalam dua fase. Fase pertama adalah memisahkan setiap baris dalam citra. Fase kedua adalah memisahkan setiap karakter yang ada pada tiap barisnya.

Lalu setiap karakter yang sudah dipisahkan akan dilakukan normalisasi sehingga menyerupai *template* yang digunakan untuk training ANN.

b. *Feature Extraction*

Feature Extraction memiliki dua tujuan, yang pertama adalah untuk mengekstrak properti yang dapat mengidentifikasi sebuah karakter secara unik, dan yang kedua adalah untuk mengekstrak properti yang dapat membedakan karakter yang mirip.

Sebuah karakter dapat ditulis dengan berbagai cara, akan tetapi manusia masih dapat mengenalinya dengan mudah. Oleh sebab itu muncullah prinsip atau logika yang melampaui berbagai macam perbedaan, sehingga *feature* yang digunakan sistem bekerja sebagai properti yang menyerupai sifat dari karakter.

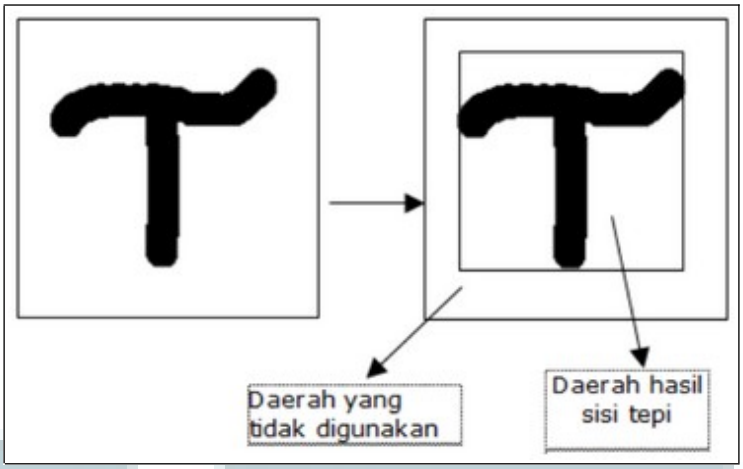
c. ANN *Training and Classification*

Sebelum pengenalan karakter dapat dilakukan, ANN harus di-*training* terlebih dahulu dengan menggunakan metode *backpropagation*, agar ANN memiliki kemampuan untuk memetakan berbagai input menjadi output yang diinginkan secara efektif.

Sedangkan menurut Hendri, langkah-langkah yang dilakukan OCR sederhana adalah sebagai berikut.

a. Mencari Sisi Tepi

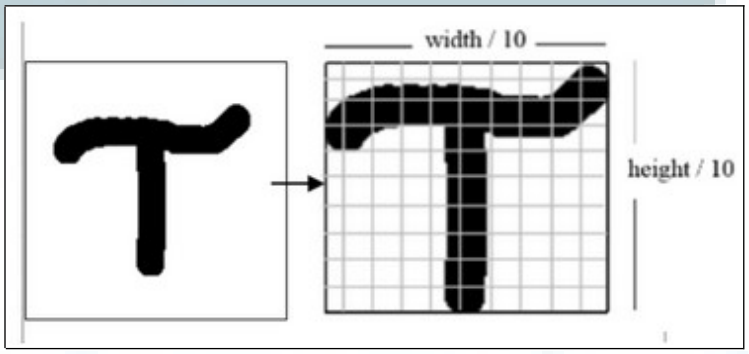
Pada tahap ini bertujuan untuk mengecilkan daerah kerja. Dengan melakukan proses pencarian sisi tepi, akan didapatkan hasil seperti yang terlihat pada Gambar 2.1. Dimana tampak sebuah kotak mengelilingi karakter ‘T’ tersebut.



Gambar 2.1 Karakter 'T' setelah didapatkan sisi tepinya
(Sumber: Hendri, 2014)

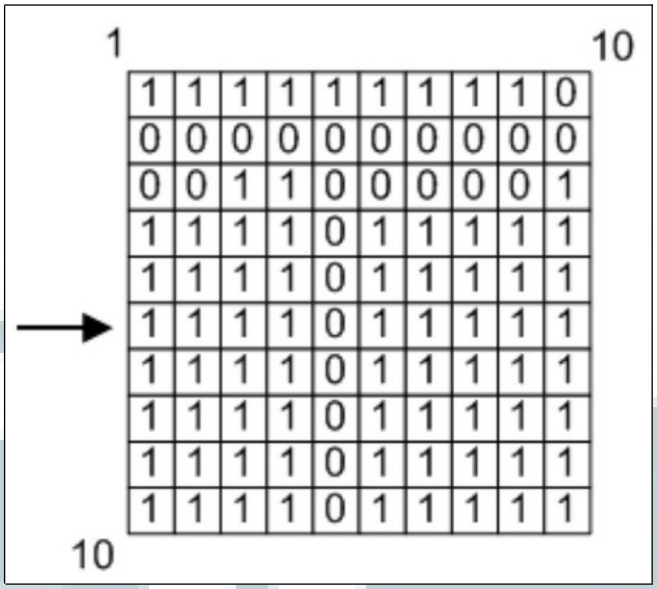
b. Reduksi Matriks 10 x 10

Pada tahap ini bertujuan untuk memasukkan karakter tersebut menjadi sebuah matriks berdimensi 10 x 10. Dengan menggunakan algoritma *reduction*, tahapan - tahapan yang dilakukan adalah sebagai berikut.



Gambar 2.2 Pembagian batas sisi tepi
(Sumber: Hendri, 2014)

Pertama daerah hasil pencarian sisi tepi dilakukan pembagian dengan 10. Masing-masing lebar dan tinggi dari daerah tersebut dibagi dengan 10, seperti yang terlihat pada Gambar 2.2.



Gambar 2.3 Hasil reduksi
(Sumber: Hendri, 2014)

Setelah dilakukan pembagian dengan 10, dilakukan pemeriksaan terhadap daerah-daerah titik temu berdasarkan sumbu x dan sumbu y antara garis-garis hasil pembagian tersebut seperti yang terlihat pada Gambar 2.3. Jika daerah titik temu berwarna hitam maka pada nilai yang dimasukkan ke dalam matriks adalah bernilai 0 (nol). Demikian sebaliknya, jika daerah titik temu berwarna putih maka nilainya adalah 1 (satu). Hasil akhir dari proses reduksi ini adalah pola dari karakter yang digambarkan yang berupa matriks berdimensi 10 x 10.

c. Pencocokan Pola

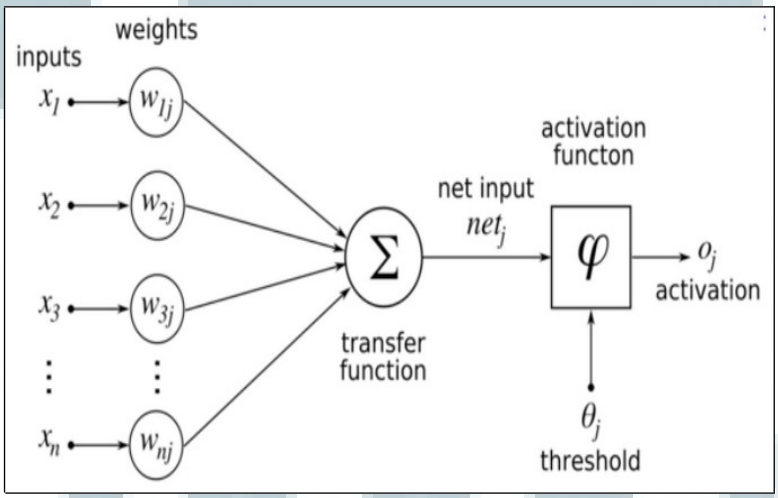
Pada tahap ini, setelah mendapatkan pola matriks berdimensi 10 x 10, maka pola ini akan dicocokan dengan pola yang ada pada *knowledge base*. Di mana persentase perhitungan serta pencocokan menggunakan algoritma *recognition*.

2.2 Artificial Neural Network

Artificial Neural Network (ANN) adalah sebuah sistem tiruan dari jaringan syaraf manusia untuk mensimulasikan proses pembelajaran dan melakukan tugas-tugas yang dapat dilakukan oleh jaringan syaraf manusia. Karena sistemnya yang menyerupai jaringan syaraf manusia, ANN dapat diterapkan dalam *feature learning* untuk mempelajari *feature* dari suatu karakter.

Pada penelitiannya, Sameeksha Barve mengungkapkan bahwa ANN menyerupai jaringan syaraf manusia dalam dua hal, diantaranya sebagai berikut.

1. ANN mendapatkan pengetahuan melalui proses pembelajaran.
2. Pengetahuan ANN disimpan di dalam *inter-neuron connection strengths* yang dikenal sebagai *synaptic weight*.



Gambar 2.4 Struktur ANN
(Sumber: Barve, 2012)

Fausett memaparkan bahwa satu *hidden layer* cukup untuk *backpropagation* memperkirakan pemetaan berkelanjutan dari pola-pola *input* menjadi pola-pola

output. Akan tetapi, dalam beberapa situasi dua *hidden layer* mungkin dapat membuat *training* menjadi lebih mudah.

Berdasarkan hasil penelitian yang dilakukan Shrivastava dan Sharma (2012), Arial merupakan font yang paling tinggi dikenali (35/36) kecuali karakter 'I'. Kesalahan pengenalan yang paling tinggi kemungkinannya terdapat pada karakter 'I', 'O',' Q', '1', dan '3'.

Tabel 2.1 *Recognition rate* untuk jenis *font* yang tidak di-*training* oleh ANN (Sumber: Shrivastava dan Sharma, 2012)

S.NO.	NAME OF THE FONT	RECOGNITION RATE
1	ARIAL	97.20
2	MICROSOFT SANS SERIF	91.67
3	CENTURY GOTHIC	88.89
4	TREBUCHET MS	86.11
5	CALIBRI	83.33
6	COPPERPLATE GOTHIC	83.33
7	LUCIDA CONSOLE	83.33
8	LUCIDA SANS TYPEWRITER	83.33
9	VERDANA	83.33
10	LUCIDA SANS	77.78
11	SEGOE UI	75.00
12	FRANKLIN GOTHIC BOOK	72.22
13	TEMPUS SANS ITC	69.40
14	TIMES NEW ROMAN	69.40
15	CANDARA	66.67
16	TAHOMA	66.67

Tabel 2.1 *Recognition rate* untuk *font* yang tidak di-*training* oleh *ANN* (lanjutan)

S.NO.	NAME OF THE FONT	RECOGNITION RATE
17	BOOKMAN OLD STYLE	63.89
18	OCR A	63.89
19	COMIC SANS MS	58.33
20	GEORGIA	33.33

2.3 Backpropagation

Backpropagation belajar menggunakan *input* dan *output* yang sudah disediakan sebelumnya dengan menggunakan dua fase siklus propagasi yang disesuaikan. Setelah pola *input* diaplikasikan sebagai stimulus pada *layer* pertama, kemudian pola tersebut dipropagasikan melalui setiap *upper layer* sampai menghasilkan pola *output*. Lalu pola *output* tersebut dibandingkan dengan *output* yang diinginkan serta perhitungan nilai error untuk setiap unit *output* **(**Otair dan Salameh, 2006).

Menurut Sameeksha Barve (2012) ada empat langkah dalam metode *backpropagation*, diantaranya sebagai berikut:

1. Menghitung seberapa cepat *error* berubah sejalan dengan berubahnya unit *output*. Ini merupakan error derivative (EA) yang merupakan perbedaan antara aktifitas sebenarnya dengan yang diinginkan.

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j \qquad ...(2.1)$$

2. Menghitung seberapa cepat error berubah sejalan dengan total input yang diterima oleh perubahan unit output. Kuantitas ini (EI) adalah hasil dari langkah pertama yang diperbanyak berdasarkan laju perubahan output terhadap perubahan total input.

$$EI_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \times \frac{\partial y_j}{\partial x_j} = EA_j y_j (1 - y_j) \quad ...(2.2)$$

3. Menghitung seberapa cepat error berubah berdasarkan bobot terhadap relasinya dengan perubahan output. Kuantitas ini (EW) adalah hasil dari langkah kedua yang diperbanyak berdasarkan tingkat aktifitas dari unit yang berelasi.

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial W_{ij}} = EI_j y_i \quad ...(2.3)$$

4. Menghitung seberapa cepat error berubah berdasarkan perubahan unit pada layer sebelumnya. Ini adalah hasil dari langkah kedua yang diperbanyak berdasarkan bobot pada relasinya terhadap unit output.

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial y_i} = \sum_j EI_j W_{ij} \quad ...(2.4)$$

Sedangkan menurut Emmanuel dan Hartono fungsi aktivasi yang paling sesuai untuk algoritma *backpropagation* adalah fungsi aktivasi sigmoid sebagai berikut.

$$y = g(x) = \frac{1}{1 + e^{-x}} \quad ...(2.5)$$

Dimana:

$$x = bias_0 + \sum_{i=1}^{n} w^i x^i \quad ...(2.6)$$

$bias_0$ = bias awal (biasanya nol)

w_i = bobot *neuron* i

x_i = nilai input untuk *neuron* i

n = jumlah *neuron* pada *layer* sebelumnnya

Untuk menghitung akurasi pengenalan karakter, Viet Dung Do dan Dong-Min Woo (2015) menyarankan untuk menggunakan MSE (*Mean Squared Error*) untuk menganalisa hasil sebagai pengukuran akurasi.

$$MSE = \frac{\sum_{i=0}^{n} (desired\ value_i - actual\ value_i)^2}{n} \quad ...(2.7)$$

Dimana:

n = jumlah entri dalam set

2.4 OpenCV

OpenCV (*Open Source Computer Vision Library*) adalah sebuah *library* perangkat lunak yang *open source* untuk *computer vision* dan *machine learning*. OpenCV dibuat untuk memberi infrastruktur umum untuk aplikasi *computer vision* dan untuk mempercepat penggunaan persepsi mesin dalam produk komersial (OpenCV, 2016). OpenCV pada mulanya dikembangkan oleh pusat

penelitian Intel di Nizhny Novgorod, Rusia, yang kemudian didukung oleh Willow Garage dan sekarang dikelola oleh Itseez (Itseez, 2016).

Menurut OpenCV, selain perusahaan-perusahaan ternama seperti Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, dan Toyota yang menggunakan OpenCV, ada banyak juga *startup* seperti Applied Minds, VideoSurf, dan Zeitera yang menggunakan OpenCV.

2.5 JavaScript

JavaScript atau yang biasa disingkat JS adalah sebuah bahasa pemrograman yang ringan dan *interpreted*. Setiap JavaScript engine mengimplementasikan JavaScript yang agak berbeda, meskipun semuanya berdasarkan pada spesifikasi dari ECMAScript, ada beberapa engine yang tidak mendukung spesifikasi ECMA secara penuh. Bahkan adapula yang mendukung fitur yang bahkan melebihi ECMA. Menurut Mozilla, ECMAScript merupakan standar untuk JavaScript dan sejak tahun 2012, semua peramban modern sudah mendukung ECMAScript 5.1 secara penuh.

Pada awalnya JavaScript dikenal sebagai Mocha pada masa pengembangannya yang kemudian rilis secara resmi dengan nama LiveScript pada September 1995 dan diganti menjadi JavaScript pada Desember 1995. Penggantian nama tersebut sempat membuat kebingungan karena membuat JavaScript terkesan sebagai sempalan dari bahasa pemrograman Java.

2.6 Precision dan Recall

Precision dan *recall* sering digunakan dalam *pattern recognition*, *information retrieval,* dan *binary classification*. *Precision* itu sendiri adalah fraksi dari instansi yang relevan diantara instansi-instansi yang terambil atau biasa juga disebut *positive predictive value*. Sedangkan *recall* adalah fraksi dari instansi relevan yang sudah diambil dari jumlah total keseluruhan instansi-instansi yang relevan.

Menurut Andreas Klintberg (2017), ada empat istilah dalam *precision* dan *recall*, diantaranya sebagai berikut.

a. *True Negative* (tn)

Melambangkan data sampel yang terklasifikasi bukan merupakan bagian dari golongan secara tepat. Contohnya, dalam kasus "hot dog" vs "bukan hot dog", *classifier* berhasil mengenali sebuah mobil bukan merupakan sebuah "hot dog".

b. *False Negative* (fn)

Melambangkan data sampel yang terklasifikasi bukan merupakan bagian dari golongan secara tidak tepat. Contohnya, dalam kasus "hot dog" vs "bukan hot dog", *classifier* salah dalam mengenali citra "hot dog" yang kurang bagus sehingga dikenali bukan sebuah "hot dog".

c. *True Positive* (tp)

Melambangkan data sampel yang terklasifikasi sebagai bagian dari golongan secara tepat. Contohnya, dalam kasus "hot dog" vs "bukan hot dog", *classifier* berhasil mengenali bahwa sebuah "hot dog" merupakan sebuah "hot dog".

d. *False Positive* (fp)

Melambangkan data sampel yang terklasifikasi sebagai bagian dari golongan secara tidak tepat. Contohnya, dalam kasus "hot dog" vs "bukan hot dog", *classifier* salah mengenali sebuah *hamburger* sebagai sebuah "hot dog".

Olson dan Delen mendefinisikan *precision* dan *recall* dalam bentuk berikut.

$$Precision = \frac{tp}{tp + fp} \quad ...(2.5)$$

$$Recall = \frac{tp}{tp + fn} \quad ...(2.6)$$

$$True\ negative\ rate = \frac{tn}{tn + fp} \quad ...(2.7)$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad ...(2.8)$$