IMPLEMENTASI ALGORITMA TWO-STAGE FOREGROUND SUB-SAMPLING UNTUK PENGENALAN ANGKA TULISAN TANGAN

Oleh
I Ketut Purnamawan
Jurusan Manajemen Informatika, FTK, UNDIKSHA

ABSTRAK

Paper ini berisi laporan pengembangan aplikasi pengenalan angka tulisan tangan dengan menggunakan algoritma two-stage foreground sub-sampling. Aplikasi dibuat dengan menggunakan MATLAB 7.0 dan Microsoft Visual Basic 6.0. Selain penerapan algoritma, pada aplikasi juga dibuat fasilitas untuk melakukan proses *one-stage classification* sebagai pembanding. *Classifier* dilatih dengan menggunakan *MNIST handwritten digit database*. Akurasi *classifier* yang dihasilkan mencapai 94% pada data training berjumlah 6000 buah dan level ektraksi data 3.

Kata-kata kunci: *OCR; handwritten-character; forground-subsampling; SVM*

ABSTRACT

This paper contains a report of development of handwritten digit recognition application using Two-stage Foreground Sub-sampling algorithm. Application was built using MATLAB 7.0 and Microsoft Visual Basic 6.0. In addition to implementing the algorithm, application also provide a facility to perform one-stage classification process for comparison. Classifier was trained using MNIST handwritten digit database. On 6000 data training and data extraction level is 3, the resulting classifier accuracy reaches 94%.

Keywords: OCR; handwritten-character; forground-subsampling; SVM

1. PENDAHULUAN

Sekarang telah banyak berkembang algoritma-algoritma pengenalan huruf atau dikenal dengan *Optical Character Recognition* (OCR). OCR mencakup pengenalan huruf yang dihasilkan oleh mesin maupun huruf hasil tulisan tangan. Pada saat ekarang pengenalan huruf hasil tulisan mesin secara luas sudah dianggap sebagai masalah yang sudah terpecahkan [1]. Hal tersebut khususnya untuk huruf

latin. Sehingga arah penelitian saat ini bergeser ke pengenalan huruf hasil mesin selain huruf latin seperti huruf Kanji dan huruf Arab dan ke pengenalan huruf hasil tulisan tangan. Sangat berfariasinya tulisan tangan manusia menyemabkan sangat sulit untuk mendapatkan algoritma yang benar-benar baik untuk semua tulisan tangan manusia. Hal ini menjadi tantangan yang bagi para peneliti. Sehingga sekarang ini banyak algortima yang telah dikembangkan untuk pengenalan huruf tulisan tangan dengan tingkat akurasi yang semakin meningkat. Salah satu algoritma terbaru adalah algoritma *two-stage foreground sub-sampling* yang dikembangkan oleh G. Vamvakas, B. Gatos, dan S. J. Perantonis [1]. Dari percobaan yang dilakukan terlihat bahwa hasil algoritma ini cukup menjanjikan, dengan mencapai tingkat akurasi yang tinggi pada berbagai dataset, bahkan tertinggi pada dataset CIL dan CEDAR. Disini algoritma *two-stage foreground sub-sampling* akan diimplementasikan untuk mengembangkan suatu aplikasi OCR untuk mengenali angka tulisan tangan.

Proses pengembangan aplikasi dan uji cobanya akan dijelaskan pada bagian-bagian selanjutnya. Pada bagian II akan dijelaskan secara garis besar mengenai algoritma *two-stage foreground sub-sampling*. Pada bagian III akan dijelaskan mengenai Supports Vector Machine (SVM) yang digunakan sebagai clasifier pada algoritma ini. Bagian IV berisi penjelasan mengenai proses implementasi algoritma. Di bagian V ditunjukkan hasil percobaan dari aplikasi yang dihasilkan. Bagian VI berisi pembahasan terhadap hasil uji coba. Dan terakhir bagian VII berisi kesimpulan dari hasil pembahasan.

2. METODE

2,1. Two-Stage Foreground Sub-Sampling

Ada dua hal utama yang dilakukan algoritma *two-stage foreground sub-sampling*. Yang pertama terletak pada proses ekstraksi fitur. Dan yang kedua teletak pada proses optimasi pengklasifikasian.

2.1.1 Ektraksi Fitur

Langkah pertama yang dilakukan dalam ekstraksi fitur adalah citra dibagi dengan satu garis vertikal dan dan satu garis horisontal, sehingga citra terbagi menjadi empat bagian. Pembagian dilakukan sehingga sedapat mungkin masing-masing bagian mempunyai jumlah foreground yang sama. Masing-masing bagian kemudian dibagi lagi dengan cara yang sama secara rekursif. Proses rekursif ini dilakukan sebanyak L kali dimana L merupakan *level of granularity* yang dipakai. Titik-titik potong antara garis vertikal dan horisontal pada level L digunakan sebagai fitur citra. Jika pembagian dilakukan sebanyak L + 1 kali, dimana L = 0 menunjukkan proses pembagian pertama, maka akan didapat sebanyak 4L buah titik potong. Karena setiap titik potong berdimensi 2, maka pada level L akan didapatkan vector fitur berdimensi 2*4L.

Yang pertama dilakukan pada proses pembagian adalah membagi citra secara vertikal. Pertama citra diproyeksikan secara vertikal, yaitu suatu array satu dimensi yang panjangnya sama dengan lebar citra dihasilkan yang isinya adalah jumlah pixel foreground pada setiap kolom. Untuk meningkatkan kepresisian, pada array ini kemudian disisipkan nila 0 didepan setiap isinya, sehingga ukuran array menjadi dua kalinya. Kemudian dicari suatu titik xq yang membagi array menjadi dua bagian sehingga jumlah isi masing-masing potongan array menjadi seseimbang mungkin. Kemudian koordinat horisontal dari titik potong x0 ditentukan. Jika xq mod 2 = 0, maka citra dibagi sebagai berikut. Citra bagian sebelah kiri adalah: {(1,1), (x0,ymax), dan citra bagian sebelah kanan adalah: {(x0,ymax), (xmax,ymax)} dimana x0 = xq div 2. Jadi kolom x0 dimiliki oleh kedua bagian. Jika xq mod 2 = 1, maka citra dibagi sebagai berikut. Citra bagian sebelah kiri adalah: {(1,1), (x0,ymax), dan citra bagian sebelah kanan adalah: {(x0 + 1,ymax), (xmax,ymax)}. Koordinat vertikal titik potong y0 dicari dengan algoritma yang sama.

2.1.2 Klasifikasi

Skema pengklasifikasian terdiri dari dua bagian, yaitu tahap training dan tahap recognition.

Tahap training: Tahap training dibagi menjadi 3 tahap. Tahap pertama adalah menentukan nilai L yang optimal untuk klasifikasi awal. Tahap kedua adalah menggabungkan class-class yang mengalami missclasification yang besar, dengan kata lain, karakternya mirip, menjadi satu kelompok. Tahap ketiga adalah menemukan nilai L baru untuk setiap kelompok, yang dapat mebedakan setiap class dalam kelompok tersebut dengan lebih baik, dan men-training classifier baru untuk setiap kelompok pada level tersebut.

Tahap recognition: Tahap pengenalan dilakukan secara bertahap. Pada clasifikasi awal, jika satu data dikenali masuk ke dalam class yang tidak termasuk dalam kelompok class-class yang mempunyai kemiripan, maka proses dihentikan. Tetapi jika data tersebut dikenali masuk kedalam salah satu kelompok yang mempunyai kemiripan, maka proses clasifikasi akan dilakukan lagi dengan menggunakan classifier kelompoknya. Classifier yang digunakan adalah SVM.

2.2. SUPPORTS VECTOR MACHINE

Supports Vector Machine (SVM) adalah salah satu *classsifier* yang sekarang banyak digunakan untuk melakukan melakukan berbagai keperluan klasifikasi. Selain untuk klasifikasi, SVM juga digunakan untuk regresi. SVM merupakan binary classifier yang membagi data menjadi dua class dengan sebuah hyperplane. Hyperplane ini tepat berada di tengah-tengah kedua class dengan jarak d ke titik data terdekat untuk masing-masing class. d disebut margin. Sedangkan titik-titik data yang berada tepat pada jarak d dari hyperplane disebut support vector. Hyperplane SVM dinyatakan dengan persamaan sebagai berikut.

$$w \bullet x + b = 0$$

dimana w merupakan normal dari hyperplane, dan $b/\|w\|$ adalah jarak hyperplane ke titik origin. Titik-titik data yang masuk ke class 1 adalah titik-titik data yang memenuhi persamaan

$$w \bullet x + b \leq -1$$

dan titik-titik data yang masuk ke class 1 adalah titik-titik data yang memenuhi persamaan

$w \bullet x + b \geq 1.$

SVM pada dasarnya adalah sebuah *binary* classifier. Namun dengan membuat suatu sekenario SVM bisa digunakan untuk *multiclass classification*. Ada dua sekenario yang sering digunakan, yaitu sekenario one againts one, dan one againts all. Pada sekenario *one* againts *one* dibuat sejumlah *classifier* yang memisahkan satu kelas dengan satu kelas lain. Setiap pasangan kelas diadu satu sama lain, kelas yang paling menang dijadikan sebagai class label akhir. Pada sekenario *one* againts *n all*, dibuat sejumlah *classifier* yang memisahkan satu kelas dengan semua kelas lainnya. Di saat data masuk ke suatu kelas dimana menghasilkan jarak yang paling jauh dari *hyperplane*, maka kelas tersebut dijadikan *class label* data tersebut. Untuk jumlah kelas yang banyak, sekenario *one* againts *one* sulit untuk dilakukan, karena akan diperlukan banyak sekali *classifier*.

2.3 DATA

Data yang dipakai dalam implementasi adalah *MNIST handwritten digit database*. Dataset ini terdiri dari 60000 data training dan 10000 data testing.

2.4 IMPLEMENTASI ALGORITMA

Algoritma diimplementasikan dengan menggunakan MATLAB 7.0. Implementasi SVM menggunakan SVM Toolbox SVM-KM [2].

Pada tahap pertama dilakukan preproccessing terhadap data citra. Citra grayscale diubah menjadi citra monocrome. Tidak dilakukan perbahan pada ukuran citra. Citra yang digunakan berukuran 28 x 28 pixel.

Tahap selanjutnya adalah proses ektraksi fitur. Proses ini dilakukan secara recursive. Proses ektraksi fitur tidak bisa dilakukan sekali secara bersamaan. Hal ini disebabkan oleh proses klasifikasi yang terdiri dari dua tahap yang masing-masing tahap memerlukan fitur dengan level berbeda.

Pada implementasi classifier dilakukan percobaan untuk menentukan sekenario multiclass yang menghasilkan akurasi yang lebih tinggi dan lebih cepat. Didapatkan bahwa sekenario one again one lebih akurat dan lebih cepat untuk data yang digunakan. Pada prosses training ditemukan satu masalah ketika jumlah data

trainingnya cukup banyak. Pada level fitur = 3, ketika data training mencapai 8000 data terjadi error pada proses yang disebabkan oleh kekurangan memori. Untuk fitur dengan level lebih besar dari 3, jumlah data training yang bisa digunakan menjadi semakin kecil. Pada level 3 vector fitur akan berdimensi 128, sedangkan pada level 4 vector fitur akan berdimensi 512.

Pada implementasi two-stage classification, pengelompokan kelas-kelas yang mirib sama dengan [1], yaitu {0}, {6}, {4, 9}, {1, 2, 7}, dan {3, 5, 8}. Pada tahap ini juga dilakukan percobaan untuk menentukan kombinasi sekenario yang terbaik untuk proses two-stage classification. Kombinasi pertama menggunakan sekenario one againts one pada stage pertama dan stage kedua. Kombinasi kedua menggunakan sekenario *one againts one* pada stage pertama dan sekenario *one againts all* pada stage kedua. Kombinasi ketiga menggunakan sekenario *one againts all* pada stage pertama dan sekenarion *one againts one* pada stage kedua. Kombinasi keempat menggunakan sekenario *one againts al*l pada kedua stage. Dari hasil percoabaan didapatkan bahwa kombinasi pertama menghasilkan akurasi paling tinggi.

Pada proses klasifikasi, selain menerapkan algoritma two-stage classification, juga disediakan fasilitas untuk melakukan one-stage classification. Setelah melakukan percobaan diketahuin bahwa sekenario *one againts one* lebih baik untuk melakukan one-stage classification.

Dengan memilih sekenario yang terbaik, fungsi-fungsi yang dibangun dengan menggunakan MATLAB 7.0 dicompilasi untuk dijadikan komponen. Komponen yang dihasilkan selanjutnya dipanggil melalui program yang dibangun menggunakan Microsoft Visual Basic 6.0.

3. HASIL

Setelah dilakukan beberapa percobaan, didapatkan beberapa hasil percobaan. Hasil percobaan yang diperhitungkan adalah tingkat akurasi klasifikasi dan lamanya waktu proses. Tabel 1 memperlihatkan tingkat akurasi dan lamanya waktu proses berdasarkan jumlah data dan level fitur pada proses training.

Tabel 1. Tabel Hasil Proses Training

Jumlah data traning	***Level fitur***		***Akurasi rata-rata (%)***	***Waktu rata-rata (menit)***	
	Stage I	***Stage II***		***Ektraksi fitur***	***Training***
1000	3	3, 3, 3	90	1	2
1000	3	4, 4, 4	90	2	6
1000	4	4, 4, 4	90	3	10
3000	3	3, 3, 3	92	3	15
3000	3	4, 4, 4	92	5	30
3000	4	4, 4, 4	92	8	50
6000	3	3, 3, 3	94	10	60

Tabel II. memperlihatkan lamanya waktu proses berdasarkan jumlah data dan level fitur pada proses testing.

Tabel 1. Tabel Hasil Proses Testing

Jumlah data traning	***Level fitur***		***Waktu rata-rata (menit)***	
	Stage I	***Stage II***	***Ektraksi fitur***	***Testing***
1000	3	3, 3, 3	1	1
1000	3	4, 4, 4	2	1
1000	4	4, 4, 4	3	1
3000	3	3, 3, 3	3	1
3000	3	4, 4, 4	5	1
3000	4	4, 4, 4	8	1
6000	3	3, 3, 3	10	1

Level fitur stage I menunjukkan level fitur yang digunakan pada proses klasifikasi tahap pertama. Level fitur stage II menunjukkan level fitur yang digunakan pada proses klasifikasi tahap kedua. Terdapat tiga buah nilai, masing-masing uttuk satu kelompok angka yang memunyai tingkat kemiripan yang tinggi. Terdapat tiga kelompok yaitu kelompok {4, 9}, {1, 2, 7}, dan {3, 5, 8}.

4. PEMBAHASAN

Dari hasil percobaan terlihat bahwa perbedaan level fitur 3 dan 4 tidak terlalu berpengaruh terhadap tingkat akurasi. Namun di lain pihak peningkatan level

fitur dari 3 ke 4 meningkatkan waktu ektaksi fitur dan training secara signifikan. Di sisi lain penigkatan jumlah data training cukup memberikan kontribusi terhadap peningkatan akurasi dengan resiko peningkatan waktu training. Namun pada kenyataannya waktu training tidak menjadi masalah, karena proses training bisa dilakukan sebelum suatu aplikasi digunakan. Pada proses klasifikasi, sebagian besar waktu dihabiskan untuk ektraksi fitur. Pada algoritma two-stage classification, dilakukan dua kali ektraksi fitur dengan level yang berbeda. Hal ini menghabiskan waktu yang cukup besar dibanding proses klasifikasi itu sendiri.

Dari hasil percobaan juga dapat di lihat bahwa akurasi rata-rata tertinggi yang bisa dicapai tidak bisa menyamai yang didapatkan oleh Vamvakas dkk [1]. Hal ini mungkin disebabkan oleh ketidakmampuan aplikasi ini untuk melakukan training dengan jumlah data yang besar.

5. PENUTUP

Dari hasil dan pembahasan dapat diambil beberapa kesimpulan sebagai berikut.

- Level fitur 3 sudah cukup optimal untuk mendapatkan tingkat akurasi yang paling tinggi.
- Penigkatan level fitur akan meningkatkan waktu training dengan besar.

Diperlukan jumlah data training yang besar agar didapat akurasi yang maksimal.

DAFTAR PUSTAKA

[1] G. Vamvakas, B. Gatos, dan S. J. Perantonis, “Handwritten character recognition through two-stage foreground sub-sampling,” *Pattern Recognition, doi:10.1016/j.patcog.2010.02.018.*

[2] S. Canu and Y. Grandvalet, V. Guigue dan A. Rakotomamonjy, “SVM and Kernel Methods Matlab Toolbox,” *Perception Systèmes et Information, INSA de Rouen, Rouen, France, 2005*.

[3] T. Fletcher, “Support Vector Machine explained,” *www.cs.ucl.ac.uk/staff/T. Fletcher/ (2009)*.