

ĐẠI HỌC BÁCH KHOA HÀ NỘI
Viện công nghệ thông tin và truyền thông



Báo cáo môn học
XỬ LÝ ẢNH SỐ

Đề tài: Nhận dạng chữ số

Giảng viên hướng dẫn :

Lã Thế Vinh

Người thực hiện:

Nguyễn Đức Tuấn – 2010 2430

Hà Nội, 2012

Mục lục

Chương I: Bài toán nhận dạng chữ viết.....	5
1 Bài toán.....	5
1.1 Bài toán nhận dạng chữ viết.....	5
1.2 Ứng dụng của nhận dạng chữ viết.....	5
2 Quy trình nhận dạng.....	6
2.1 Tiền xử lý.....	6
2.2 Phân đoạn chữ viết.....	7
2.2.1 Phân đoạn chữ viết dựa trên phân tích hình chiếu.....	7
2.2.2 Phân đoạn chữ viết dựa trên phân tích thành phần liên kết.....	8
2.3 Trích xuất đặc trưng.....	8
2.3.1 Các đặc trưng thống kê.....	9
2.3.2 Các đặc trưng cấu trúc.....	10
2.3.3 Các đặc trưng toàn cục và biến đổi chuỗi.....	10
2.4 Phân loại.....	10
2.5 Hậu xử lý.....	10
Chương II: Một số kỹ thuật tiền xử lý	11
1 Khử bỏ nhiễu.....	11
2 Nhị phân hóa ảnh.....	11
2.1 Tách ngưỡng sử dụng ngưỡng cục bộ.....	13
2.2 Tách ngưỡng sử dụng các đặc trưng cục bộ.....	14
3 Loại bỏ độ nghiêng chữ.....	14
Chương III: Mạng neuron.....	17
1 Mạng neuron nhân tạo.....	17
1.1 Mạng neuron sinh học.....	17
1.1.1 Neuron nhân tạo.....	18
1.1.2 Mạng neuron nhân tạo.....	19
1.1.3 Học trong mạng neuron nhân tạo.....	19
1.2 Mạng Kohonen.....	20
1.2.1 Mạng Kohonen.....	20
1.2.2 Cấu trúc mạng Kohonen.....	21
1.2.3 Giải thuật học.....	21
Chương IV: Chương trình thử nghiệm.....	23
1 Chương trình googleDocsDemo.....	23
1.1 Một số hình ảnh từ chương trình.....	24

Lời nói đầu

Trong môn học xử lý ảnh chúng em được tiếp cận với nhiều kiến thức khác nhau. Đây là những kiến thức thực sự bổ ích nhưng hết sức thú vị và có tính ứng dụng cao. Một trong những ứng dụng từ những kiến thức này là nhận dạng chữ viết. Trên cơ sở những kiến thức học tập được cũng như kiến thức tự nghiên cứu, em đã hoàn thành báo cáo này.

Do thời gian có hạn, báo cáo không thể không tránh khỏi những sai sót. Em rất mong nhận được những phản hồi từ thầy cô.

Em xin chân thành cảm ơn thầy !

Chương I: Bài toán nhận dạng chữ viết

1 Bài toán

1.1 Bài toán nhận dạng chữ viết

Bài toán nhận dạng chữ viết (Optical Character Recognition (OCR)) là một bài toán có ý nghĩa thực tiễn cao. Từ một ảnh chụp quét một đoạn văn bản, bài toán yêu cầu xác định nội dung đoạn chữ trong ảnh.

Bài toán được đề xuất do một phát minh của M. Sheppard được gọi là GISMO, một robot biết đọc viết. Năm 1954, máy nhận dạng chữ đầu tiên đã được phát triển bởi J. Rainbow dùng để đọc chữ in hoa nhưng rất chậm. Năm 1967, Công ty IBM đã thương mại hóa hệ thống nhận dạng chữ.

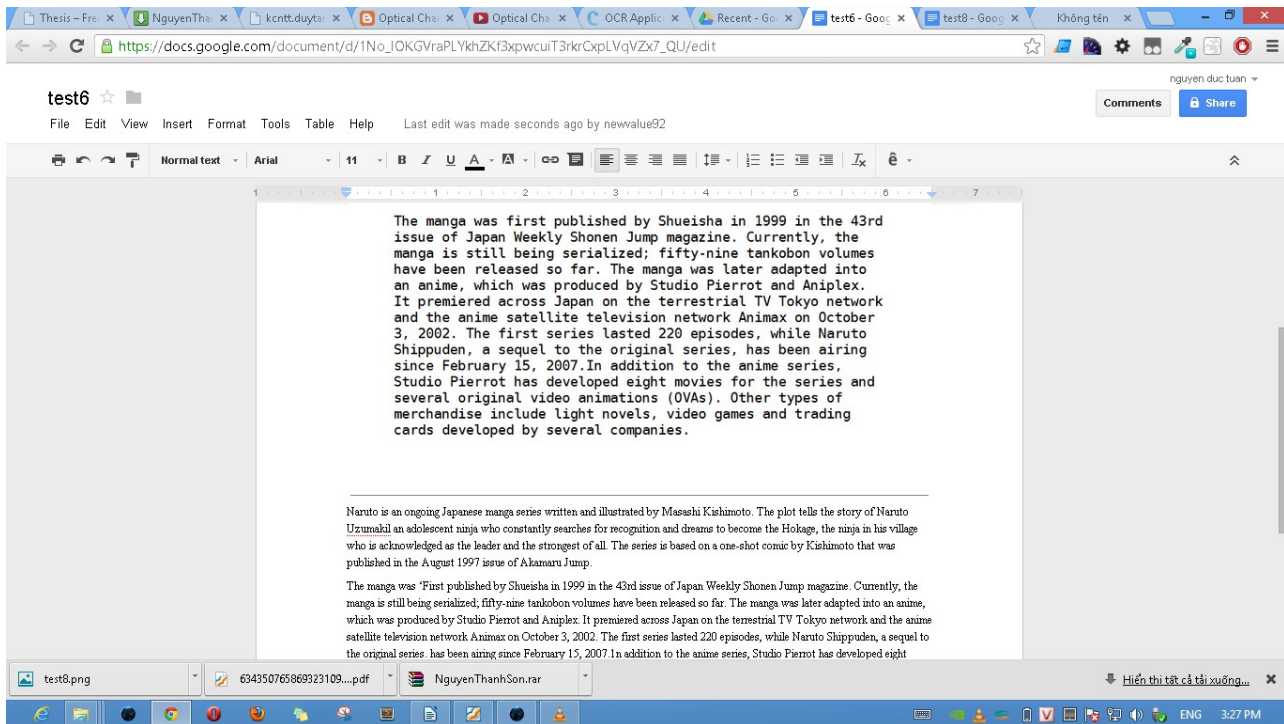
Giai đoạn 1980 – 1990, với sự phát triển của các thiết bị phần cứng máy tính và các thiết bị thu nhận dữ liệu, các phương pháp luận nhận dạng đã được phát triển trong giai đoạn trước đã có được môi trường lý tưởng để triển khai các ứng dụng nhận dạng chữ. Trong giai đoạn này, các hướng nghiên cứu chỉ tập trung vào các kỹ thuật nhận dạng hình dáng chữ chưa áp dụng cho thông tin ngữ nghĩa. Điều này dẫn đến sự hạn chế về hiệu suất nhận dạng, không hiệu quả trong nhiều ứng dụng thực tế.

Từ những năm 1990 trở lại đây, các kỹ thuật nhận dạng kết hợp với các phương pháp luận trong lĩnh vực học máy (Machine Learning) được áp dụng rất hiệu quả. Nhiều công cụ học máy như mạng neuron, SVM,... được áp dụng.

1.2 Ứng dụng của nhận dạng chữ viết

Những thành tựu của nhận dạng chữ viết được ứng dụng khá rộng rãi. Dưới đây là một số ứng dụng tiêu biểu :

- Google docs : Đây là một công cụ giúp ta trực tiếp soạn thảo văn bản trên web. Năm 2009, google docs đã tích hợp OCR API cho phép có thể chuyển đổi nội dung trong hình ảnh thành chữ viết.
- Các chương trình chuyển đổi từ các ảnh chụp scan, sách scan sang văn bản như ABBY FineReader, tesseract-ocr giúp giảm đáng kể thời gian đánh lại các văn bản. Ở Việt Nam, một số phần mềm nổi tiếng loại này là VnDOCR, VietOCR.
- Các ứng dụng bàn phím ảo trên các thiết bị cảm ứng: Đó là các ứng dụng cho phép người dùng ghi chú, đánh văn bản thông qua việc viết tay. Ứng dụng sẽ chuyển nó sang văn bản text tương ứng.



Hình 1: Ví dụ nhận dạng chữ với Google Docs

2 Quy trình nhận dạng

Một hệ thống nhận dạng thông thường sẽ lần lượt thực hiện một số bước sau:

- Tiền xử lý ảnh(preprocessing)
- Phân đoạn chữ viết (Segmentation)
- Trích chọn đặc trưng(Feature extraction)
- Phân loại (Classify)
- Hậu xử lý (post processing)

Chúng ta lần lượt tìm hiểu qua các bước này.

2.1 Tiền xử lý

Đây là bước đầu trong xử lý ảnh. Các ảnh thô đầu vào cho quá trình nhận dạng thường có chất lượng không tốt do điều kiện chụp ảnh : bị nhiễu, độ tương phản thấp, chữ bị nghiêng... Do vậy để đạt được

kết quả nhận dạng tốt hơn, ta phải tiến hành một số bước tiền xử lý. Các kỹ thuật tiền xử lý được thực hiện tùy theo đặc điểm của ảnh đầu vào. Dưới đây là một số kỹ thuật thông dụng thường được sử dụng:

- Lọc bỏ nhiễu : loại bỏ nhiễu của ảnh đầu vào
- Nhị phân hóa ảnh : Chuyển đổi ảnh đầu vào thành ảnh chỉ biểu diễn bằng 2 giá trị 1, 0 tương ứng với việc xác định pixel thuộc chữ hay thuộc nền.
- Loại bỏ độ nghiêng chữ : Loại bỏ hiện tượng nghiêng văn bản trong hình ảnh do việc chụp

Trong phạm vi báo cáo này, em sẽ trình bày tập trung vào một số kỹ thuật tiền xử lý. Một số kỹ thuật tiền xử lý ảnh sẽ được trình bày trong chương II của báo cáo này.

2.2 Phân đoạn chữ viết

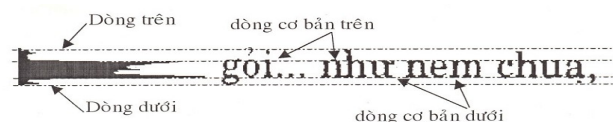
Từ một bức ảnh nhị phân, bước tiếp theo là ta sẽ phải tách được các ký tự ra khỏi văn bản. Hai kỹ thuật cơ bản được sử dụng để phân đoạn chữ viết là sử dụng phân tích hình chiếu (Projection analysis – PA) và phân tích dựa trên thành phần liên kết (Connected component analysis – CCA). Phân tích PA thích hợp với trường hợp mà các chữ cái có thể tách biệt bởi dấu cách (ví dụ chữ latin đánh máy). CCA thích hợp cho trường hợp mà mỗi chữ viết gồm nhiều thành phần liên kết (connected component) kết hợp với nhau hoặc khó xác định bằng phương pháp PA (ví dụ chữ hán). Ta sẽ tìm hiểu qua 2 phương pháp này

2.2.1 Phân đoạn chữ viết dựa trên phân tích hình chiếu

Phương pháp này được áp dụng cho trường hợp mà các chữ dễ dàng tách được bởi các đường thẳng ngang dọc. Phương pháp này dựa trên phép chiếu. Dựng trên ảnh nhị phân đầu ra của bước tiền xử lý, ta sẽ tính toán histogram theo từng dòng, cột ứng với các pixel thuộc chữ (có màu đen). Ta tạm gọi số lượng pixel trên một dòng, số lượng pixel trên một cột là năng lượng cột. Ví dụ ta có kết quả như hình dưới :



Hình 2: Năng lượng dòng và năng lượng cột



Hình 3: Khảo sát năng lượng dòng

Từ hình vẽ ta có nhận xét sau:

- Năng lượng dòng cao ở phần chứa chữ, thấp nhất ở vùng trống giữa các dòng, năng lượng dòng thấp nhất ở vùng trống ngăn cách dòng
- Năng lượng cột tập chung cao ở phần chữ cái và thấp ở vùng trống giữa các ký tự

Do vậy dựa trên nhận xét này ta có thể tách được dòng và chữ. Tuy nhiên phương pháp này cũng có hạn chế. Trong trường hợp chữ nghiêng, các chữ không tách biệt nhau bởi dấu cách, tính chất này không còn đúng, phương pháp này đem lại kết quả không chính xác.

2.2.2 Phân đoạn chữ viết dựa trên phân tích thành phần liên kết

Phương pháp này dựa trên nhận xét: các pixel thuộc chữ viết sẽ luôn kề các pixel thuộc chữ viết đó (pixel láng giềng). Do vậy, ta có thể tách các chữ viết thông qua việc tách các thành phần kề nhau. Với các chữ viết có nhiều nét, ta phải thực hiện phép ghép các thành phần này(overlap).



Hình 4: Phân đoạn chữ viết sử dụng CCA

Phương pháp này ưu điểm hơn phương pháp dựa trên phép chiếu bởi có thể tách được các thành phần mà chữ khá liền nhau như chữ viết tay. Ví dụ như hình ở trên. Rõ ràng việc tách các nét chữ không thể dùng phép chiếu bởi các chữ này chồng nên nhau. Do vậy nếu dựa trên đồ thị năng lượng ta khó thể xác định được đường biên của từng ký tự. Tuy nhiên, nếu sử dụng phương pháp này, ta có thể tách được các thành phần khá tốt như có thể thấy trên hình vẽ.

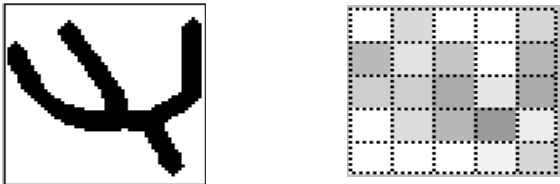
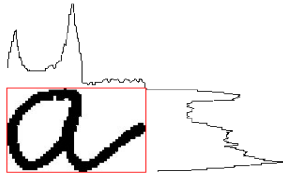

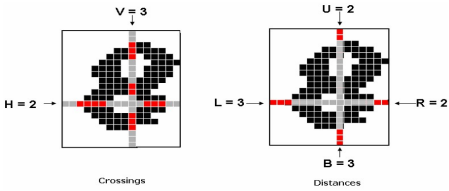
2.3 Trích xuất đặc trưng

Từ mỗi vùng xác định các chữ cái, ta sẽ trích xuất ra các đặc trưng (vector đặc trưng) để phục vụ cho quá trình phân loại. Hiện nay, cho đến nay, đã có nhiều đặc trưng khác nhau đã được nghiên cứu và áp dụng cho việc nhận dạng chữ viết. Tuy nhiên, nếu xem xét tổng thể, ta có thể phân loại chúng thành những nhóm đặc trưng chính sau:

- Đặc trưng thống kê
- Các đặc trưng cấu trúc
- Các biến đổi toàn cục và khai triển chuỗi

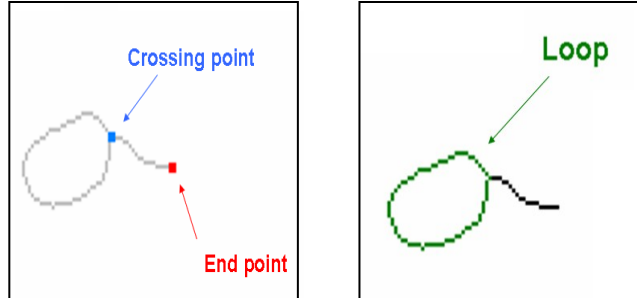
2.3.1 Các đặc trưng thống kê

Đây là các đặc trưng dựa trên phân phối thống kê của các điểm trên chữ cái. Một số đặc trưng thống kê cơ bản được thể hiện qua bảng sau:

Đặc trưng	Đặc điểm
<p>Đặc trưng vùng</p> 	<ul style="list-style-type: none"> Phân chia chữ cái thành các vùng và tính toán đặc trưng trên từng vùng (cường độ sáng trung bình, hướng,..) Giúp thu được các đặc trưng tính cục bộ
<p>Đặc trưng Histogram</p> 	<ul style="list-style-type: none"> Tính toán histogram trên từng dòng, cột của chữ Hữu dụng trong nhiều trường hợp như phân biệt chữ 'm' và 'n'
<p>Profiles</p> 	<ul style="list-style-type: none"> Đếm số pixel nằm giữa hình bao chữ và đường biên của chữ Giúp biểu diễn tốt hình dạng bên ngoài của chữ, ví dụ phân biệt chữ 'q' và 'p'
<p>Giao điểm và khoảng cách</p> 	<ul style="list-style-type: none"> Giao điểm: đếm số dải pixel liên tục chuyển tiếp từ vùng chữ sang vùng nền Khoảng cách: Khoảng cách giữa 2 biên chữ

2.3.2 Các đặc trưng cấu trúc

Đây là các đặc trưng dựa trên hình dạng, hình thái của chữ, ví dụ như tỉ lệ, số điểm điểm vượt qua, điểm rẽ nhánh, đường cong,...



Hình 5: Một số đặc trưng cấu trúc

2.3.3 Các đặc trưng toàn cục và biến đổi chuỗi

Để có thể thu nhận thêm các đặc trưng mới, người ta thực hiện các phép biến đổi ảnh từ đó trích xuất ra các vector đặc trưng. Ví dụ: biến đổi fourier, vector trung tâm và vector zenrike.

2.4 Phân loại

Sau giai đoạn trích xuất đặc trưng, ta thu được một vector đặc trưng cho mỗi chữ cái. Ta sẽ sử dụng các bộ phân loại để phân loại vector này với các chữ cái tương ứng với vector đó. Có nhiều bộ phân loại khác nhau hiện nay như:

- k-nearest neighbor (kNN)
- Bayes classifier
- Mạng neuron
- Hidden Markov Models (HMM)
- Support Vector Machines (SVM)

Trong báo cáo này, em sử dụng mạng neuron để huấn luyện và phân loại chữ viết. Chi tiết về mạng neuron sẽ được trình bày trong chương III của báo cáo này.

2.5 Hậu xử lý

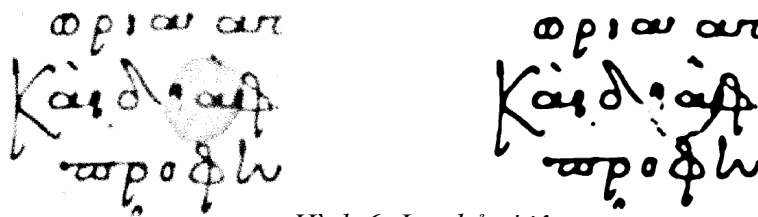
Đây là bước cuối cùng trong xử lý ảnh. Mục đích của bước này nhằm chỉnh lại một số kết quả của nhận dạng. Ví dụ sau khi nhận dạng một ký tự, ta có vài khả năng sảy ra. Nhưng dựa trên mối quan hệ ngữ nghĩa với các ký tự khác, ta lựa chọn ra ký tự tốt nhất (ví dụ, ta thường có cặp phụ âm nh, ngh, tr, th..)

Chương II: Một số kỹ thuật tiền xử lý

Trong chương này, em sẽ trình bày một số kỹ thuật tiền xử lý cơ bản em đã cài đặt và thực hiện để thực hiện tiền xử lý ảnh

1 Khử bỏ nhiễu

Do điều kiện chụp, scan, ảnh đầu vào thường bị nhiễu. Do vậy, để tránh ảnh hưởng của nhiễu, ta phải thực hiện một phép khử bỏ nhiễu. Việc này có thể thực hiện dễ dàng thông qua một phép lọc trung bình như phép lọc Gauss.



Hình 6: Lọc bỏ nhiễu

2 Nhị phân hóa ảnh

Một ảnh thông thường là ảnh màu hoặc ảnh đa mức xám. Ảnh này không giúp nhiều cho việc nhận dạng bởi mục đích của chúng ta là tách ra thành 2 lớp đối tượng (chữ - foreground và nền -background). Do vậy, công việc của tiền xử lý là biến đổi từ ảnh đa mức xám thành thành ảnh nhị phân.

Với ảnh màu để chuyển thành ảnh đa mức xám, ta thực hiện dựa trên biểu thức:

$$\text{gray} = 0.2 * R + 0.72 * G + 0.08 * B$$

Trong đó:

- gray : mức xám tương ứng trong ảnh đa mức xám
- R, G, B là giá trị màu Red, Green, Blue trong ảnh màu

Từ ảnh đa mức xám, ta có thể chuyển đổi thành ảnh nhị phân thông qua một số cách sau:

- Tách ngưỡng toàn cục (global gray level threshold)
- Tách ngưỡng cục bộ (local gray level threshold)
- Tách ngưỡng dựa trên đặc trưng cục bộ (Local feature threshold)

2.1. Tách ngưỡng toàn cục

Trong phương pháp này, ta đơn giản xác định một ngưỡng toàn cục để thực hiện tách ngưỡng. Với 1 ảnh có phạm vi độ xám: $G=\{0,1, ..., L-1\}$, khi đó các pixel thuộc chữ viết có thể được coi là $C_0 = \{0,1,..., t\}$ và pixel thuộc nền có độ xám trong tập $C_1=\{t+1, t+2,... L-1\}$. Do vậy ta chỉ cần xác định một ngưỡng t . Ngưỡng này có thể được người dùng tự chọn hoặc theo giải thuật Otsu. Ta sẽ tìm hiểu giải thuật này. Ý tưởng của giải thuật Otsu là tìm một ngưỡng mà phương sai giữa các lớp là nhỏ nhất. Phương sai này được biểu diễn bởi biểu thức bởi công thức:

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

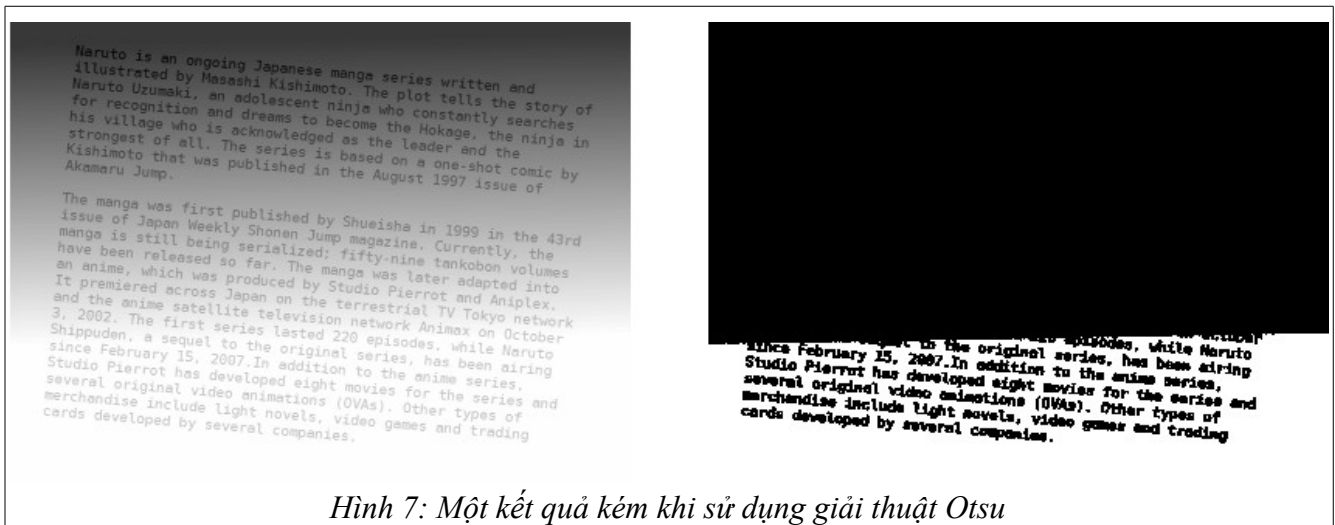
trong đó:

ω_i là tỉ lệ mà pixel thuộc nhãn đó, σ_i^2 là phương sai trên từng nhãn. Phương sai trên từng nhãn được tính theo công thức:

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_1(t)\omega_2(t) [\mu_1(t) - \mu_2(t)]^2$$

Trong đó, μ_i là giá trị trung bình của độ xám trên từng nhãn.

Trong phương pháp tách ngưỡng sử dụng ngưỡng toàn cục, ta chỉ có một ngưỡng duy nhất. Do vậy, giải thuật này không đem lại kết quả tốt với những trường hợp mà ảnh đầu vào biến đổi liên tục như gradient. Ví dụ như trong hình 7 dưới đây:



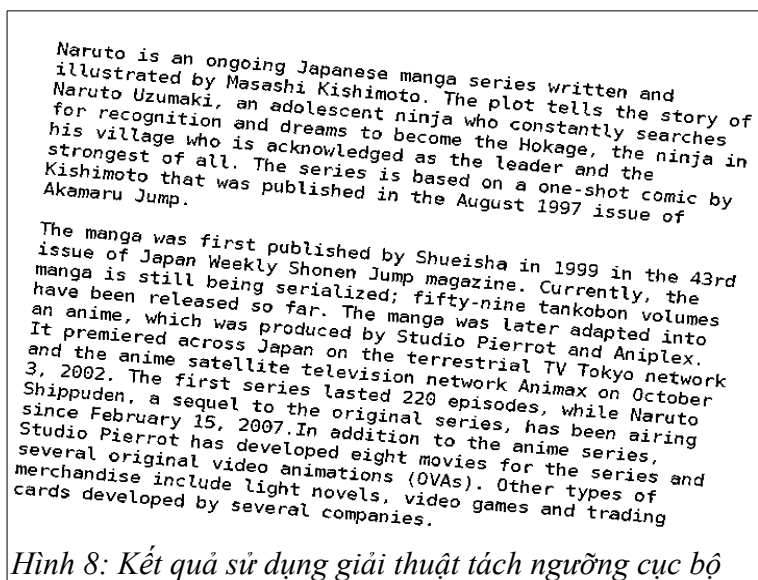
Hình 7: Một kết quả kém khi sử dụng giải thuật Otsu

2.1 Tách ngưỡng sử dụng ngưỡng cục bộ

Để khắc phục hạn chế trên của tách ngưỡng toàn cục, trong phương pháp này ta còn dựa trên các thông tin trên các vùng cục bộ để tách ngưỡng. Ta cùng đi xem xét một phương pháp như thế là Bernsen. Ý tưởng giải thuật bernsen như sau:

- Lần lượt xét trên các vùng cục bộ kích thước cố định (ví dụ 5x5).
- Xác định một ngưỡng độ xám $global_threshold = 128$, và ngưỡng sai khác $constrast$
- Trên vùng này ta tính 2 giá trị:
 - mid_gray : Giá trị độ xám trên toàn vùng
 - $local_contrast = max_gray - min_gray$: độ lệch lớn nhất về độ xám trên vùng cục bộ
- Nếu $local_constrast < constrast$: ta có thể coi các pixel trong vùng là đồng nhất, khi đó tất cả các pixel trên vùng này sẽ được xếp vào chữ viết hay nền dựa vào việc so sánh mid_gray và ngưỡng toàn cục $threshold$
 - $pixel = (mid_gray > threshold) ? background : object$
- Nếu $local_constrast > constrast$: Khi đó việc xác định pixel thuộc chữ hay nền sẽ được dựa trên đánh giá trên từng pixel
 - $pixel = (pixel > mid_gray) ? background : object$

Kết quả hình trên khi thực hiện với giải thuật Bernsen được thể hiện qua hình:



2.2 Tách ngưỡng sử dụng các đặc trưng cục bộ

Đây cũng là một cách dùng để nhậ phân khá tốt. Ta sẽ sử dụng các đặc trưng cục bộ để tách ngưỡng. Ví dụ là đặc trưng cạnh. Ta sẽ thu được ảnh nhậ phân bằng cách tìm đường biên. Khi đó các chữ sẽ được bao bởi các đường biên. Hình dưới đây minh họa kết quả hình trên sử dụng phương pháp xác định đường biên Canny:

Naruto is an ongoing Japanese manga series written and illustrated by Masashi Kishimoto. The plot tells the story of Naruto Uzumaki, an adolescent ninja who constantly searches for recognition and dreams to become the Hokage, the ninja in his village who is acknowledged as the leader and the strongest of all. The series is based on a one-shot comic by Kishimoto that was published in the August 1997 issue of Akamaru Jump.

The manga was first published by Shueisha in 1999 in the 43rd issue of Japan Weekly Shonen Jump magazine. Currently, the manga is still being serialized; fifty-nine tankōbon volumes have been released so far. The manga was later adapted into an anime, which was produced by Studio Pierrot and Aniplex. It premiered across Japan on the terrestrial TV Tokyo network and the anime satellite television network Animax on October 3, 2002. The first series lasted 220 episodes, while Naruto Shippuden, a sequel to the original series, has been airing since February 15, 2007. In addition to the anime series, Studio Pierrot has developed eight movies for the series and several original video animations (OVAs). Other types of merchandise include light novels, video games and trading cards developed by several companies.

Hình 9: Tách ngưỡng sử dụng giải thuật xác định đường biên Canny

3 Loại bỏ độ nghiêng chữ

Trong thực tế, ta thường gặp trường hợp mà chữ bị nghiêng đi hơn là được in thẳng hàng. Trong trường hợp đó ta phải loại bỏ độ nghiêng này. Quá trình này thường diễn qua 2 bước:

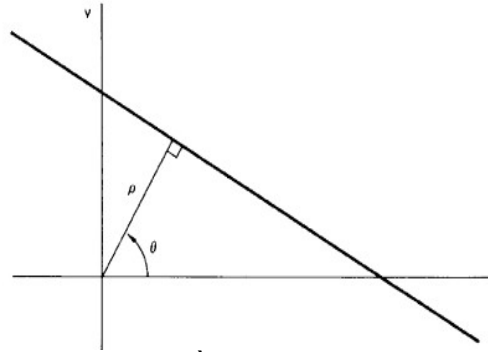
- Xác định độ nghiêng
- Loại bỏ độ nghiêng

Ta cũng có nhiều kỹ thuật khác nhau để xác định độ nghiêng của ảnh. Ta sẽ nghiên cứu một phương pháp đơn giản để xác định độ nghiêng của chữ là phương pháp Hough (Hough transform). Trong

phương pháp này, ta sẽ dựa trên xấp xỉ đường thẳng. Một đường thẳng Hough là đường thẳng có dạng:

$$x \cos(\theta) + y \sin(\theta) = r$$

hay được đặc trưng bởi tham số (θ, r) như trong hình dưới đây.



Hình 10: Đường thẳng Hough

Như ta biết, các văn bản gồm nhiều dòng song song với nhau. Do vậy ta có thể xấp xỉ các dòng này bởi các đường thẳng Hough và đi tìm tham số (θ, r) thích hợp và chứa nhiều đường thẳng Hough nhất. Khi đó tham số θ chính là giá trị góc nghiêng của dòng chữ.

Giả sử ảnh kích thước $M \times N$, q thuộc đoạn $[0, 45]$. Ta sẽ chỉ xét các giá trị q cách nhau 0.5. Ta minh họa một code đơn giản để tính giá trị này như sau:

```
change = Math.PI / 180;
max = 0;
angle = 0;
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        for (int a = 0; a < 45; a += 1) {
            temp = a * change;
            p = Math.floor(i * Math.sin(temp) + j * Math.cos(temp));
            H[p][a + 45] += 1;
            if (H[p][a] > max) {
                max = H[p][a];
                angle = a;
            }
        }
    }
}
return angle;
```

Từ đó ta sẽ tiến hành khử độ nghiêng. Để làm việc đó ta chỉ cần thực hiện một phép quay ảnh để chỉnh lại, trong đó giá trị góc quay là $-\theta$ trong đó θ là giá trị góc nghiêng được tính toán ra ở trên. Phép quay ma trận một góc θ được thể hiện qua công thức sau:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Ta có kết quả minh họa chạy giải thuật Hough:

```
//canvas
this._super = ocrObj;
this.canvas = ocrObj.canvas;
this.context = ocrObj.context;
this.privCanvas = ocrObj.privC
this.privContext = ocrObj.priv

var index;
```

```
//canvas
this._super = ocrObj;
this.canvas = ocrObj.canvas;
this.context = ocrObj.context;
this.privCanvas = ocrObj.priv
this.privContext = ocrObj.priv

var index;
```

Hình 11: Loại bỏ độ nghiêng sử dụng Hough

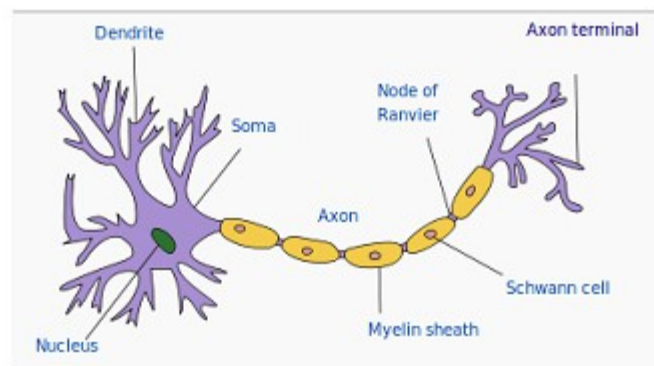
Chương III: Mạng neuron

1 Mạng neuron nhân tạo

1.1 Mạng neuron sinh học

Neuron (tế bào thần kinh) là một loại tế bào dễ bị kích thích điện, có thể xử lý và truyền thông tin qua các tín hiệu điện / hoá học. Một neuron gồm những thành phần sau [2]:

- soma: thành phần trung tâm của neuron, có chứa nhân tế bào,
- dendrites: các sợi nhánh, nơi nhận hầu hết các tín hiệu đầu vào của neuron,
- axon: trục chính, mang các tín hiệu ra từ soma. Một neuron chỉ có một axon nhưng axon có thể phân nhánh, do đó cho phép neuron tương tác với nhiều đích khác nhau.
- Điểm kết thúc của axon, chứa *synapses*. Đây là nơi giải phóng các hoá chất truyền dẫn thần kinh để giao tiếp giữa các neuron.



Hình 12: Tế bào neuron

Mạng neuron là một dãy các neuron liên kết với nhau. Trong đó axon của neuron này liên kết với dendrite của neuron khác [3].

Nếu tín hiệu đầu vào của một neuron đạt giá trị xác định, nó sẽ gửi các tín hiệu tương ứng đến các neuron kết nối với nó thông qua axon.

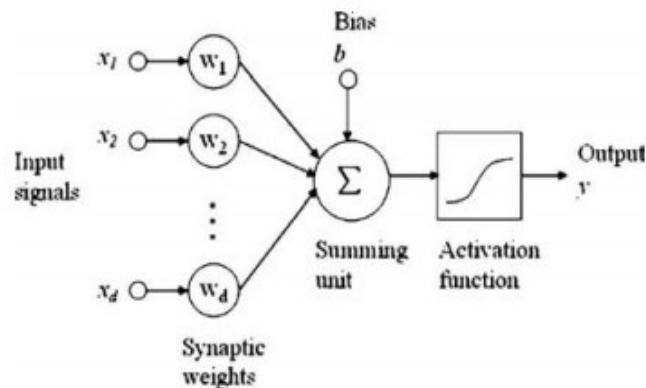
Dựa vào cấu trúc của mạng neuron sinh học, người ta đã xây dựng nên mô hình mạng neuron

nhân tạo để có thể đạt được sức mạnh tính toán tương tự.

1.1.1 Neuron nhân tạo

Neuron: Trong ANN, neuron (còn được gọi là một unit hay node) là sự mô hình hoá của neuron sinh học. Neuron bao gồm các thành phần sau:

- **Vector đầu vào** (input vector): tương ứng với dendrites, chứa các giá trị đầu vào để neuron xử lý. Các đầu vào này được liên kết từ các neuron khác, mỗi liên kết có một thuộc tính gọi là trọng số (weight) chỉ mức độ mạnh yếu của liên kết. Thông thường giá trị của trọng số nằm trong khoảng $[0,1]$.
- **Hàm tổng** (summation function): hàm tổng hợp các giá trị đầu vào của neuron thành một giá trị đơn.
- **Hàm truyền**: hàm giới hạn đầu ra của mỗi neuron. Thường được giới hạn trong đoạn $[0,1]$ hoặc $[-1,1]$,
- **Đầu ra**: tín hiệu đầu ra của mỗi neuron. Một neuron có tối đa một đầu ra.



Hình 13: Mô hình một neuron nhân tạo

Với neuron ở hình 4.2, tổng các đầu vào được tính như sau:

$$v = \sum_{i=1}^d w_i x_i + b$$

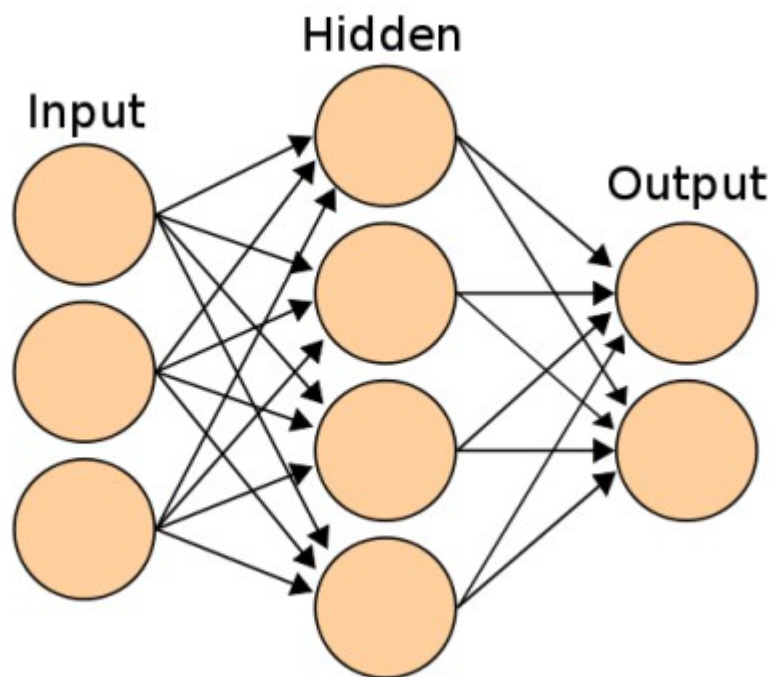
Giá trị đầu ra của neuron:

$$y = f(v) \text{ (} f() \text{ là hàm truyền).}$$

1.1.2 Mạng neuron nhân tạo

Tương tự như mạng neuron sinh học, mạng neuron nhân tạo là sự kết nối giữa các neuron nhân tạo với nhau. Việc kết nối các neuron là tùy ý. Mạng neuron được tổ chức thành các lớp, mỗi lớp bao gồm nhiều neuron có cùng chức năng với nhau.

Hình 4.3 là một mạng neuron 3 lớp (Input, Hidden, Output) với 9 neuron. Các neuron thuộc lớp Input gọi là các neuron đầu vào, các neuron thuộc lớp Output là các neuron đầu ra. Các tín hiệu sẽ được đưa vào các neuron đầu vào, sau đó đi đến các neuron của lớp tiếp theo (lớp Hidden), tiếp đó lại đưa đến các neuron của lớp Output và đi ra ngoài môi trường. Mạng neuron như thế này gọi là mạng MLP (Multilayer Perception- Mạng truyền thẳng nhiều lớp).



Hình 14: Một mạng neuron gồm 3 lớp

Ngoài mạng MLP, dựa vào cấu trúc mạng có thể chia ra các loại mạng khác như mạng một lớp, mạng neuron phản hồi, mạng neuron hồi quy, . . .

1.1.3 Học trong mạng neuron nhân tạo

Học là quá trình thay đổi hành vi của các vật theo một cách nào đó làm cho chúng có thể thực hiện tốt hơn trong tương lai.

Với mạng neuron, cho một tập các vector đầu vào X , tập các vector đầu ra tương ứng là Y . Học là việc tìm ra một ánh xạ $f: X \rightarrow Y$ dựa vào các vector đầu vào/đầu ra đó.

Tập X được gọi là tập dữ liệu học hay tập huấn luyện (training set), mỗi phần tử x thuộc X được gọi là một mẫu huấn luyện (training sample).

Ảnh xạ f được xác định bởi các trọng số của các liên kết trong mạng, chính vì vậy bản chất của việc học chính là điều chỉnh giá trị các trọng số. Trong quá trình học, các trọng số này sẽ dần hội tụ đến giá trị để mạng đạt trạng thái sao cho với mọi $x \in X$ thì $f(x) = y$ (y là đầu ra mong muốn của x).

Có 3 phương pháp học:

- **Học có giám sát:** mẫu huấn luyện sẽ có dạng (x,y) với x là đầu vào, y là đầu ra tương ứng. Mục tiêu là tìm ảnh xạ f khớp với các ví dụ.
- **Học không giám sát:** mẫu huấn luyện chỉ vai gồm đầu vào x . Nhiệm vụ của học không giám sát là phân chia tập huấn luyện thành các nhóm con, mỗi nhóm chứa các vector đầu vào có đặc trưng giống nhau.
- **Học tăng cường:** còn gọi là học thưởng phạt. Với học tăng cường, dữ liệu học thường không được cho trước mà được tạo ra trong quá trình hoạt động. Kết quả của các dữ liệu này cũng không được cung cấp mà chỉ có một số thông tin để đánh giá kết quả do ANN tạo ra (VD: "tốt", "xấu"). Dựa vào tín hiệu để đánh giá kết quả mà mạng điều chỉnh các trọng số cho phù hợp. Nếu tín hiệu đánh giá là "tốt" thì điều chỉnh các trọng số sao cho gần với kết quả hơn, ngược lại nếu đánh giá là "xấu" thì điều chỉnh trọng số về xa kết quả hơn.

1.2 Mạng Kohonen

1.2.1 Mạng Kohonen

Ảnh xạ tự tổ chức (SOM) là một loại ANN sử dụng phương pháp học không giám sát. *SOM* khác với các ANN khác ở chỗ nó sử dụng một hàm lân cận (neighborhood function) để bảo toàn trật tự sắp xếp của không gian dữ liệu đầu vào [4].

SOM được giới thiệu lần đầu tiên bởi Teuvo Kohonen, vì vậy đôi khi được gọi là mạng Kohonen. Với các loại ANN khác, người ta chỉ quan tâm đến giá trị của thông tin đầu vào mà không quan tâm đến mối liên hệ giữa các dữ liệu mẫu hay vùng lân cận, nhưng SOM đã quan tâm đến các yếu tố này.

Tự tổ chức (self-organization): Đầu ra "đúng" không được cung cấp do đó quá trình cập nhật các trọng số là tự động.

Với mạng Kohonen, một vector đầu vào được cung cấp mỗi bước. Tất cả các vector đầu vào này tạo nên "môi trường" cho mạng neuron. Mỗi một lần nhận đầu vào là một lần thay đổi mạng để thích nghi với dữ liệu mới. Đến một lúc, mạng neuron có thể trở thành đại diện cho môi trường (tập các vector đầu vào).

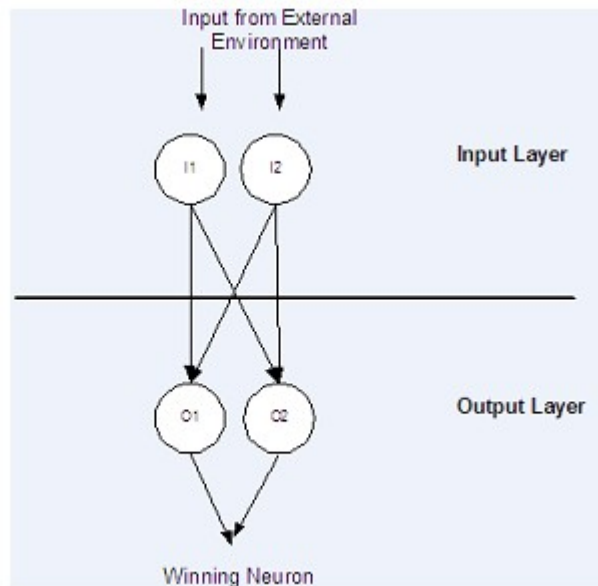
Bảo toàn trật tự sắp xếp (topology-preserving): Các mẫu dữ liệu thuộc không gian nhiều chiều khi ánh xạ thành các mảng neuron với số chiều nhỏ hơn (thường là 2 chiều) được bảo toàn thứ tự sắp xếp

không gian của chúng.

Cụ thể, với mạng Kohonen, quá trình học là quá trình ánh xạ một neuron tương ứng với một lớp các đầu vào cùng loại. Các neuron lân cận biểu diễn cho những đầu vào với các giá trị "gần nhau".

1.2.2 Cấu trúc mạng Kohonen

Mạng Kohonen bao gồm 2 lớp: một lớp đầu vào (với số neuron bằng số chiều vector đầu vào) và lớp đầu ra (với số neuron bằng số loại mẫu). Không có lớp ẩn.



Hình 15: Mạng SOM

Chỉ có duy nhất một neuron sinh ra kết quả trong mạng neuron. Neuron này gọi là winning neuron, là neuron tương ứng với tập giá trị đầu vào mà có chứa đầu vào hiện tại. Giá trị kết quả thường là chỉ số của winning neuron.

1.2.3 Giải thuật học

Giả sử có mạng Kohonen với n neuron lớp đầu vào và m neuron lớp đầu ra. $X = (x_1, x_2, \dots, x_n)$ là vector dữ liệu đầu vào, w_{ij} là trọng số của liên kết giữa neuron đầu vào thứ j và neuron đầu ra thứ i .

Mục đích của quá trình là mỗi neuron (của lớp ra) học cách nhận diện một lớp các tập vector đầu vào. Khi mạng nhận một vector đầu vào, nó sẽ chọn ra neuron nào thích hợp nhất với dữ liệu này.

Tiêu chí để chọn neuron (còn gọi là winning neuron) có thể là khoảng cách Euclide giữa vector đầu vào và vector trọng số của các liên kết đi vào neuron đó nhỏ nhất.

$$d_j = \sum_{i=1}^m (x_i - w_{ji})^2$$

Khi đã chọn được winning neuron, ta cập nhật lại vector trọng số của nó cho phù hợp với vector đầu vào. Ở đây, ta sử dụng đại lượng η - tốc độ học, có giá trị trong khoảng $[0,1]$. η càng lớn thì trọng số sau khi cập nhật càng gần với vector đầu vào. Giá trị thích hợp của η là 0.4 - 0.5.

Ngoài winning vector, các neuron lân cận cũng cần phải được cập nhật trọng số, với cường độ cập nhật có thể không cần lớn bằng. Ta sử dụng hàm lân cận $\theta(k; j)$ để chỉ mức độ lân cận của hai neuron j và k . Một cách chọn hàm $\theta(k; j)$ là $\theta(k; j)$ bằng 1 với i là lân cận của k và bằng 0 với j còn lại. Hoặc cũng có thể sử dụng hàm lân cận giảm dần từ 1 về 0 khi j ngày càng xa k .

Giải thuật:

- Bắt đầu: Khởi tạo giá trị ngẫu nhiên cho m vector trọng số liên kết n -chiều của m neuron lớp đầu ra.
- Bước 1: Chọn vector đầu vào X trong tập dữ liệu mẫu,
- Bước 2: Chọn winning neuron, giả sử là k ,
- Bước 3: Cập nhật vector trọng số của k và các neuron lân cận theo công thức: $w_i \leftarrow w_i + \eta \theta(i,k)(X - w_i)$
- Bước 4: Nếu đã lặp đủ số lần cho trước, hoặc sai số đã đủ nhỏ thì kết thúc. Nếu không thì trở về bước 1.

Chú ý: Sau mỗi lần lặp, mạng neuron lại càng gần đến trạng thái ổn định. Lúc đó "khoảng cách" giữa các neuron "xa" hơn (do sự phân biệt các lớp dữ liệu đầu vào ngày càng rõ ràng). Do đó cần giảm dần các giá trị tốc độ học η và hàm lân cận θ ().

Nhận xét: Trong quá trình học, chúng ta không cần dùng đến kết quả của các dữ liệu học. Như vậy hoàn toàn có thể học được bất cứ khi nào có dữ liệu đầu vào mà không cần biết kết quả. Rõ ràng giải thuật học của mạng Kohonen là học không có giám sát.

Chương IV: Chương trình thử nghiệm

1 Chương trình googleDocsDemo

GoogleDocsDemo là một chương trình mô phỏng hoạt động của bộ công cụ nổi tiếng của Google là Google Docs. Từ một đầu vào là một ảnh. Chương trình sẽ cho phép bôi đen các đoạn văn bản ngay trực tiếp trên hình ảnh như trên text bình thường. Khi đã bôi đen xong, đoạn văn bản trong vùng lựa chọn sẽ được tự động nhận dạng.

Hoạt động của chương trình như sau:

- Load ảnh và thực hiện tiền xử lý (nhị phân hóa ảnh + loại bỏ độ nghiêng) để được ảnh nhị phân tốt
- Sử dụng một bộ phân đoạn để tách dòng (highlight). Từ tập các dòng này, ta tách được các ký tự.
- Từ mỗi các ký tự này, ta trích xuất đặc trưng:
 - vùng: mỗi chữ được chia thành vùng 8×8 , tại mỗi vùng tính giá trị độ xám trung bình
 - Chiều: tính histogram trên 2 nửa trái và phải của mỗi chữ
 - profile: tính thuộc tính profile trên các chữ đã được chuẩn hóa thành 8×8 trên cả 2 phương

→ Tổng cộng có $8 * 8 + 2 * 8 + 4 * 8 = 112$ đặc trưng cho mỗi chữ.
- Trên cơ sở là tập các đặc trưng, sử dụng mạng SOM để nhận dạng các chữ cái

Chương trình được viết bằng Javascript/ HTML5 nên có thể chạy trên bất kỳ trình duyệt nào hỗ trợ HTML5.

1.1 Một số hình ảnh từ chương trình

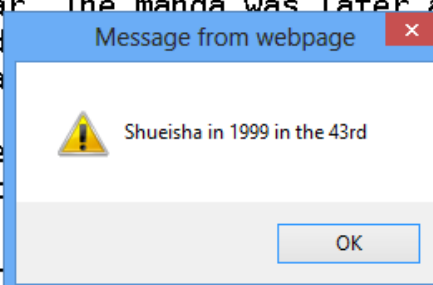
Naruto is an ongoing Japanese manga series written and illustrated by Masashi Kishimoto. The plot tells the story of Naruto Uzumaki, an adolescent ninja who constantly searches for recognition and dreams to become the Hokage, the ninja in his village who is acknowledged as the leader and the strongest of all. The series is based on a one-shot comic by Kishimoto that was published in the August 1997 issue of Akamaru Jump.

The manga was first published by Shueisha in 1999 in the 43rd issue of Japan Weekly Shonen Jump magazine. Currently, the manga is still being serialized; fifty-nine tankobon volumes have been released so far. The manga was later adapted into an anime, which was produced by Studio Pierrot and Aniplex. It premiered across Japan on the terrestrial TV Tokyo network and the anime satellite television network Animax on October 3, 2002. The first series lasted 220 episodes, while Naruto Shippuden, a sequel to the original series, has been airing since February 15, 2007. In addition to the anime series, Studio Pierrot has developed eight movies for the series and several original video animations (OVAs). Other types of merchandise include light novels, video games and trading cards developed by several companies.

Hình 16: Công cụ highlight

Naruto is an ongoing Japanese manga series written and illustrated by Masashi Kishimoto. The plot tells the story of Naruto Uzumaki, an adolescent ninja who constantly searches for recognition and dreams to become the Hokage, the ninja in his village who is acknowledged as the leader and the strongest of all. The series is based on a one-shot comic by Kishimoto that was published in the August 1997 issue of Akamaru Jump.

The manga was first published by Shueisha in 1999 in the 43rd issue of Japan Weekly Shonen Jump magazine. Currently, the manga is still being serialized; fifty-nine tankobon volumes have been released so far. The manga was later adapted into an anime, which was produced by Studio Pierrot and Aniplex. It premiered across Japan on October 3, 2002. The first series, Naruto, aired on the Tokyo network and the anime satellite TV channel. The second series, Naruto Shippuden, a sequel to the first, premiered on October 3, 2007. Studio Pierrot has developed several original video animations (OVAs). Other types of merchandise include light novels, video games and trading cards developed by several companies.



Hình 17: Nhận dạng đoạn văn lựa chọn

Tài liệu tham khảo

- [1] MOHAMED CHERIET, NAWWAF KHARMA, CHENG-LIN LIU, and CHING Y. SUEN. CHARACTER RECOGNITION SYSTEMS: A Guide for Students and Practioners. Wiley, 2007.
- [2] Giorgos Vamvakas: Optical Character Recognition for Handwritten Characters
- [3] Wikipedia. Neural network, 2012. Available at http://en.wikipedia.org/wiki/Neural_network.
- [5] Wikipedia. Self-organizing map, 2012. Available at http://en.wikipedia.org/wiki/Self-organizing_map.
- [5] Raul Rojas. Neural Networks: A Systematic Introduction. Springer, 2010.
- [6] Jeff T Heaton. Introduction to Neural Networks with Java. Heaton Research, 2005.