TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

BÀI TẬP LỚN MÔN TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: NHẬN DẠNG KÝ TỰ VIẾT TAY TIẾNG VIỆT

Giáo viên hướng dẫn: **Nguyễn Nhật Quang**

Nhóm sinh viên thực hiện:

1. LÊ NGỌC MINH **20071946**

2. ĐỖ BÍCH NGỌC **20072097**

Lớp: **Khoa học máy tính – K52**

HÀ NỘI 11/2010

MỤC LỤC

1. GIỚI THIỆU BÀI TOÁN

Nhận dạng kí tự quang học (Optical Character Recognition – OCR) là lĩnh vực nghiên cứu cách chuyển đổi ảnh số được chụp hay quét từ tài liệu viết tay, đánh máy hay in thành dạng văn bản máy tính có thể hiểu được.

Trên thế giới, công nghệ OCR đã có những tác động sâu sắc đến nhiều lĩnh vực trong sản xuất và đời sống. Việc chuyển các văn bản in trên giấy thành dạng điện tử nhỏ gọn và dễ tìm kiếm giúp hàng triệu trang sách báo đến được với bạn đọc khắp nơi trên thế giới. Bằng cách kết hợp với phần mềm text-to-speech lượng tài liệu này có thể được đọc thành tiếng cho những người khiếm thị. Nhiều bưu điện đã áp dụng hệ thống phân loại thư tự động dựa trên máy đọc bì thư có cài phần mềm OCR. Các ngân hàng đọc nội dung của séc để chống rửa tiền, gian lận và cả phát hiện khủng bố. OCR còn đi vào đời sống hàng ngày qua những thiết bị thông tin cá nhân (PDA) giúp người sử dụng nhập dữ liệu bằng cách viết lên màn hình cảm ứng thay vì đem theo bộ bàn phím cồng kềnh.

Ở Việt Nam, công nghệ OCR mới chỉ phát triển ở giai đoạn đầu với một vài bộ phần mềm nhận dạng kí tự in như VnDOCR, VietOCR, ABBYY trong khi đó lĩnh vực nhận dạng chữ viết tay vẫn còn bỏ ngỏ.

Với số lượng lớn tài liệu viết tay cần được xử lí cũng như sự phát triển của công nghệ di động và PDA đây là một hướng nghiên cứu đầy triển vọng.

Quá trình OCR gồm nhiều bước như phân tích cấu trúc văn bản, tách dạng, tách kí tự, kiểm tra ngữ nghĩa để tăng độ chính xác… nhưng bước cơ sở mà bất kỳ chương trình OCR nào cũng phải thực hiện là nhận dạng kí tự (đơn lẻ). Trong thời gian hạn hẹp của đồ án môn học chúng em chọn thực hiện bước này.

2. MÔ TẢ BÀI TOÁN

Giả thiết rằng ở bước xử lí trước kí tự đã được phân lập, kết quả là các ảnh nhị phân kích thước 60x80 mỗi ảnh chứa một kí tự tiếng Việt (có dấu) trong đó các kí tự có độ nghiêng không quá lớn và kích thước hợp chuẩn với sai số chấp nhận được, cần chuyển kí tự thành dạng mã hoá Unicode.

Chuẩn kích thước của chữ cái:

• ascender height: trùng với cạnh trên của ảnh.

• cap height: trùng với cạnh trên của ảnh.

• median: 1/3 chiều cao ảnh.

• baseline: 4/5 chiều cao ảnh.

• descender height: trùng với cạnh dưới của ảnh.

• chiều rộng: xấp xỉ chiều rộng ảnh.

Như vậy có nghĩa là chương trình sẽ không xử lí những chữ cái có kích thước quá nhỏ, quá nghiêng lệch hay biến dạng quá mức. Các chữ cái như thế giả thiết đã được đưa về dạng chuẩn (với sai số chấp nhận được) ở bước xử lí trước.

3. PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN

Chương trình "Nhận dạng ký tự viết tay tiếng Việt" sử dụng mô hình mạng neuron và thuật toán lan truyền ngược. Sau đây là chi tiết về phương pháp này.

3.1. Cơ sở lý thuyết

3.1.1. Mạng neuron

➢ Mạng neuron nhân tạo (artificial neural network) là một mô hình toán học hay mô hình tính toán lấy cảm hứng dựa trên cấu trúc của mạng thần kinh. Một mạng neuron bao gồm các nhóm neuron được nối với nhau, trên cơ sở đó thông tin được xử lý.

➢ Mô hình mạng neuron nhân tạo thường được áp dụng với các bài toán nhận dạng, đặc biệt với các bài toán có nhiều biểu diễn hình ảnh.

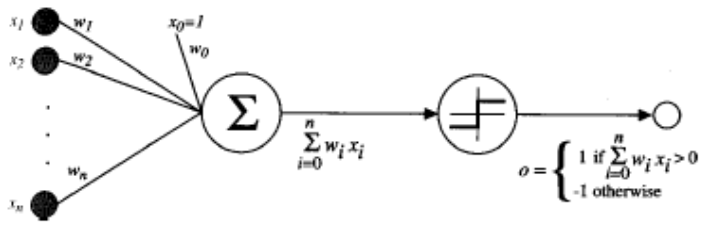
➢ Cùng với giải thuật lan truyền ngược, mạng neuron thích hợp với các bài toán mang các đặc điểm sau:

- ✓ Một thể hiện được biểu diễn bởi nhiều cặp giá trị.
- ✓ Hàm mục tiêu đầu ra có thể có giá trị rời rạc, giá trị thực hoặc một vector giá trị rời rạc hoặc giá trị thực.
- ✓ Các ví dụ học có thể có lỗi.
- ✓ Thời gian huấn luyện dài là chấp nhận được.
- ✓ Có thể yêu cầu sự tiến hóa nhanh của hàm mục tiêu cần học.
- ✓ Khả năng con người hiểu hàm mục tiêu cần học là không quan trọng.

3.1.2. Perceptron

Cơ bản của mạng neuron nhân tạo dựa trên khái niệm perceptron.

a. Biểu diễn perceptron:



Một perceptron nhận giá trị đầu vào là một vector thực, tính toán tổ hợp tuyến tính của đầu vào đó và đưa ra đầu ra bằng 1 nếu kết quả lớn hơn một ngưỡng nào đó, và bằng -1 nếu ngược lại:

$$o(x_1,...,x_n) = \begin{cases} 1 & if\ w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n > 0 \\ -1 & otherwise \end{cases}$$

vỡi mỗi w_i là một hằng giá trị thực, hay trọng số, quyết định sự đóng góp của đầu vào x_i vào đầu ra của perceptron. Giá trị w_0 là một ngưỡng để tổ hợp giữa trọng số và đầu vào $w_1x_1 + ... + w_nx_n$ phải vượt qua để perceptron cho ra giá trị 1.

Có thể viết:

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

Huấn luyện một perceptron học bao gồm việc chọn các giá trị trọng số $w_0, ..., w_n$ cho thích hợp.

b. Nguyên tắc huấn luyện perceptron

Một cách để học một vector trong số chấp nhận được là bắt đầu với một trọng số ngẫu nhiên, sau đó áp dụng từng ví dụ học cho perceptron, thay đổi giá trị trọng số nếu nó phân loại nhầm ví dụ. Quá trình này được lặp lại qua nhiều lần đến khi perceptron phân loại các ví dụ học chính xác. Trọng số thay đổi sau mỗi bước theo *nguyên tắc huấn luyện perceptron* như sau:

$$w_i \leftarrow w_i + \Delta w_i$$

với

$$\Delta w_i = \eta(t - o)x_i$$

Ở đây t là đầu ra mục tiêu cho ví dụ học hiện tại, o là đầu ra sinh bởi perceptron và η là giá trị hằng dương gọi là *tốc độ học (learning rate)*. Vai trò của tốc độ học là kiểm soát mức độ trọng số thay đổi sau mỗi bước. Nó thường được gán giá trị nhỏ (VD 0.1).

c. Sai số huấn luyện (training error)

Sai số thường được tính bằng:

$$E = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2$$

với D là tập ví dụ học, t_d là đầu ra mục tiêu của ví dụ học d, và o_d là đầu ra tính toán của ví dụ học d.

3.1.3. Mạng neuron nhiều lớp và giải thuật lan truyền ngược

a. Hàm ngưỡng

Một perceptron chỉ cho đầu ra là một hàm tuyến tính. Hàm ngưỡng được sử dụng để đưa các giá trị đầu ra là một hàm không tuyến tính của các giá trị đầu vào.

Các hàm ngưỡng hay sử dụng là:

- Hàm sigmoid:

$$o(\vec{x}) = \sigma(\vec{w} \cdot \vec{x})$$

$$\sigma(y) = \frac{1}{1 + e^{-y}}$$

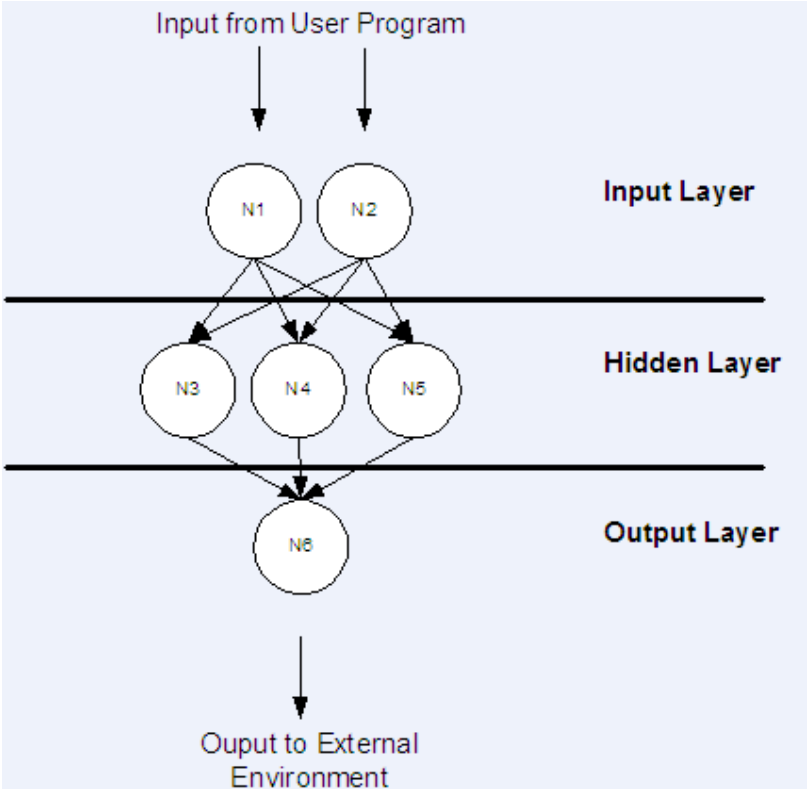
- Hàm tanh:

$$o(\vec{x}) = \tanh(\vec{w} \cdot \vec{x})$$

$$\tanh(y) = \frac{e^{y} - e^{-y}}{e^{y} + e^{-y}}$$

b. Mạng neuron feedforward

- Mạng neuron feedforward là mạng neuron mà các neuron ở lớp trước nối một chiều với lớp sau của nó.



- Thông thường mạng neuron nhiều lớp chia thành 3 loại lớp:
  - ✓ Lớp input: Là giao diện của mạng neuron với môi trường ngoài, chỉ có nhiệm vụ lấy đầu vào.
  - ✓ Lớp hidden: Là các lớp ẩn ở giữa, có nhiệm vụ tính toán.
  - ✓ Lớp output: Là đầu ra của bài toán.

Số neuron ở lớp input và output thường xác định với các bài toán, tuy nhiên số lớp hidden và số neuron ở mỗi lớp hidden cần xác định bằng thực nghiệm.

c. Giải thuật backpropagation

BACKPROPAGATION(training_example, η, n_{in}, n_{out}, n_{hidden})
Mỗi ví dụ học là một gặp có dạng $(\vec{x},\vec{t})$ *với* $\vec{x}$ *là vector đầu vào và* $\vec{t}$ *là vector mục tiêu.*
η là tốc độ học. n_{in}, n_{out}, n_{hidden} *lần lượt là số neuron ở lớp input, ouput và hidden. Đầu vào từ neuron i đến neuron j ký hiệu là* x_{ji}, *và trọng số từ neuron i đến neuron j ký hiệu là* w_{ji}.

- Tạo một mạng feedforward với n_{in} input neuron, n_{out} ouput neuron, n_{hidden} hidden neuron.
- Khởi tạo trọng số là các giá trị ngẫu nhiên nhỏ (VD giữa -.05 và .05)

- Cho đến khi thỏa mãn điều kiện kết thúc:

  Với mỗi $(\vec{x},\vec{t})$ thuộc ví dụ học:

  1. Cho giá trị đầu vào $\vec{x}$ và tính toán giá trị đầu ra *o* của mỗi neuron.

  Lan truyền sai số ngược lại mạng:

  2. Với mỗi neuron k ở lớp output, tính sai số δ_k

  $$\delta_k = o_k(1-o_k)(t_k - o_k)$$

  3. Với mỗi neuron h ở lớp hidden, tính sai số δ_h

  $$\delta_k = o_h(1-o_h)\sum_{k\in outputs} w_{kh}\delta_k$$

  4. Cập nhật lại mỗi trọng số w_{ji}

  $$w_{ji} = w_{ji} + \Delta w_{ji}$$

  với

  $$\Delta w_{ji} = \eta\delta_j x_{ji}$$

d. Momentum

Một cách phổ biến để thay đổi nguyên tắc cập nhật trọng số trong thuật toán là làm cho cập nhật trọng số trong vòng lặp thứ n phụ thuộc một phần vào lần cập nhật thứ (n - 1) như sau:

$$\Delta w_{ji}(n) = \eta\delta_j x_{ji} + \alpha\Delta w_{ji}(n-1)$$

$0 \le \alpha < 1$ là một hằng số gọi là momentum.

3.2. Giải quyết bài toán

3.2.1. Chương trình

- Input: Là một ảnh nhị phân của chữ cần nhận dạng.
- Ouput: Ký tự đã được nhận dạng cùng mã unicode của nó.
- Chương trình gồm chức năng vẽ và lấy ảnh từ bên ngoài để nhận dạng.

3.2.2. Khởi tạo mạng neuron

- Lớp input: Gồm 60 đầu vào.

60 đầu vào được lấy bằng cách tách biên ảnh thành chuỗi Fourier, lấy nhiều nhất 6 thành phần liên thông và 10 giá trị của chuỗi Fourier với mỗi thành phần.

- Lớp output: Gồm 16 đầu ra.

Các đầu ra chính là mã nhị phân của unicode của ký tự cần nhận dạng.

- Lựa chọn hàm ngưỡng: Do các đầu ra là các bit nên hàm sigmoid được chọn để làm hàm ngưỡng.

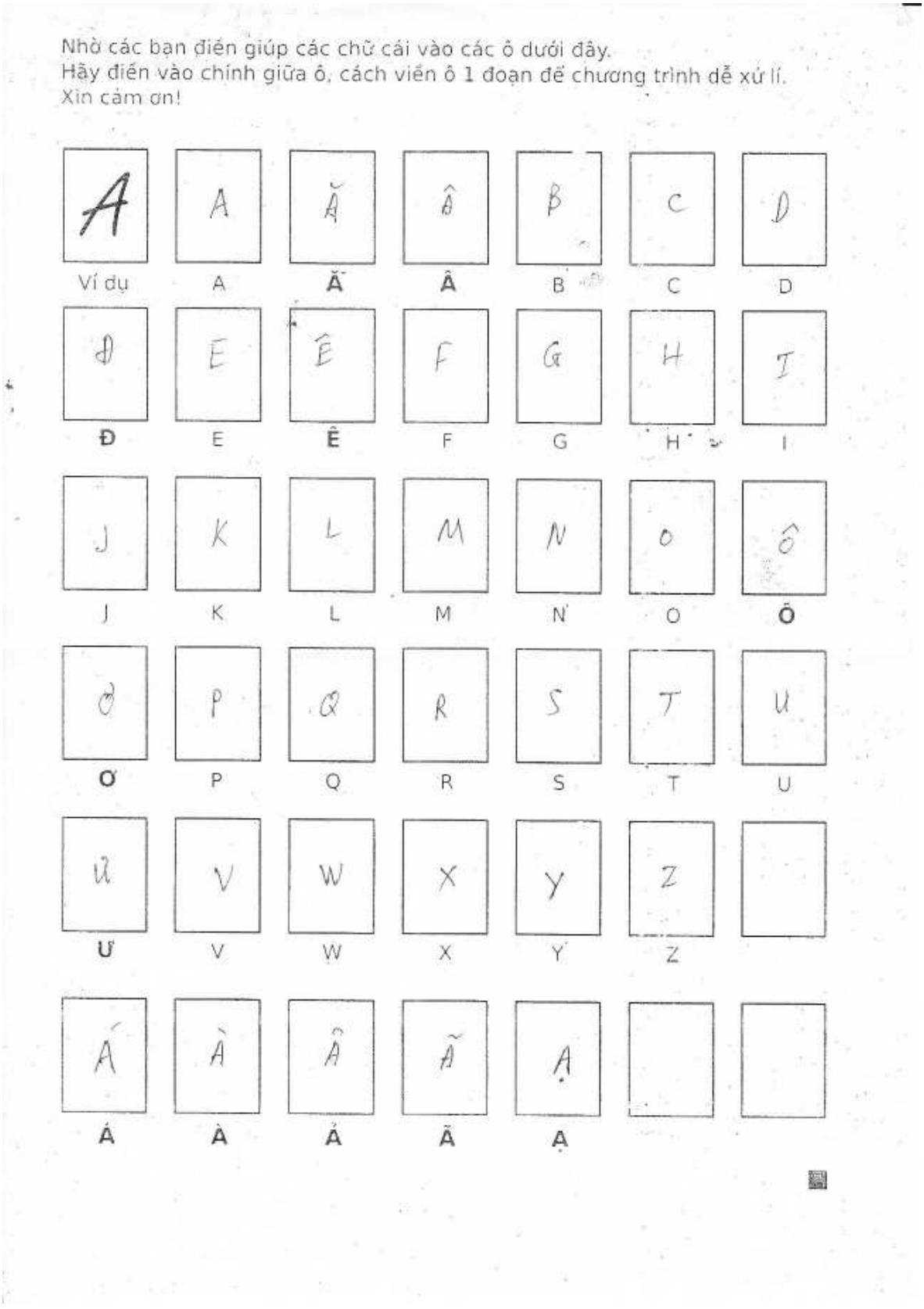
- Tập ví dụ học và kiểm tra

  ✓ Các ví dụ học và kiểm tra được lấy từ các mẫu viết tay thực tế của các sinh viên trong giảng đường.

  ✓ Các mẫu sau khi scan được xử lý thành ảnh nhị phân và tách sẵn thành các thành phần đầu ra và đầu vào tương ứng để huấn luyện và kiểm tra.

3.2.3. Chuẩn bị dữ liệu

Để thu thập mẫu chữ viết tay chúng em đã nhờ các bạn trong giảng đường điền các chữ cái vào phiếu. Số lượng phiếu phát ra là khoảng 200 phiếu. Các phiếu này khi thu về sẽ được quét vào trong máy và dùng một chương trình tách thành từng kí tự.

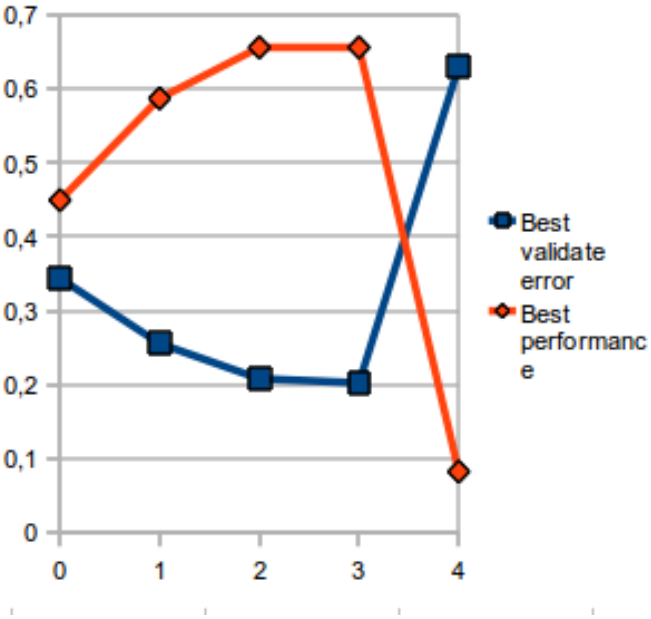


Hình 1 Phiếu thu thập mẫu chữ viết tay

3.2.4. Các kết quả thực nghiệm

a. Số lớp hidden

Layer size	Learning rate	Momentum	Layer count	Best validate error	Best performance	Training time
100	0.5	0.6	0	0.3438487922	0.4495412844	0.1009166667
100	0.5	0.6	1	0.2557802063	0.5871559633	0.4101333333
100	0.5	0.6	2	0.2082746529	0.6559633028	1.0681
100	0.5	0.6	3	0.2025941125	0.6559633028	2.2036166667
100	0.5	0.6	4	0.6304081646	0.0825688073	1.5053166667

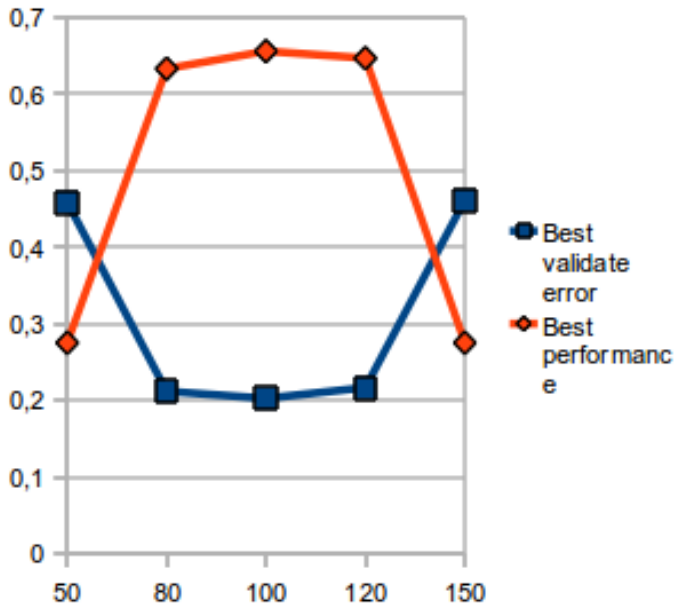


Mạng neuron với 3 lớp ẩn có xác suất nhận dạng đúng tương đương với mạng có 2 lớp ẩn nhưng error nhỏ hơn một chút. Thời gian huấn luyện của mạng tăng khá nhanh khi số lớp ẩn tăng từ 0 đến 3.

Kết quả của mạng neuron 4 lớp ẩn thấp hơn hẳn các mạng còn lại do số lớp ẩn lớn yêu cầu số lần lặp lớn hơn để tinh chỉnh kết quả. Trong khi đó điều kiện dừng của thuật toán được chọn cố định là sau 100 lần lặp mà không giảm được validate error.

b. Số neuron ở lớp hidden

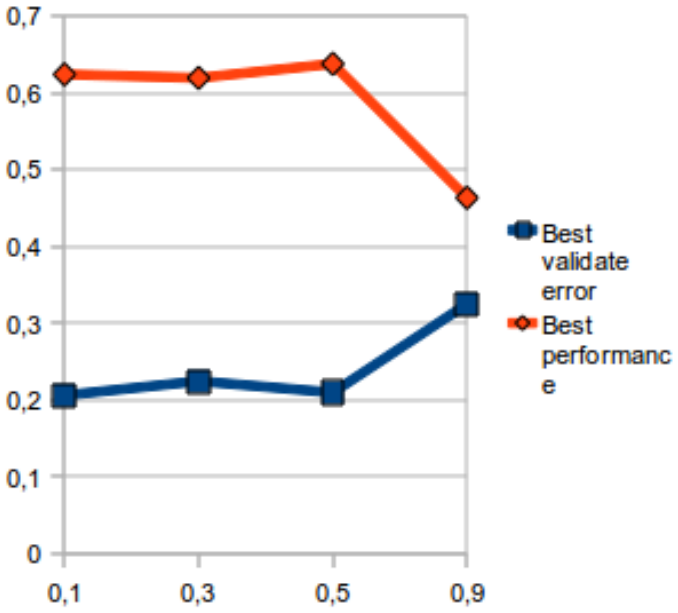
Layer count	Learning rate	Momentum	Layer size	Best validate error	Best performance	Training time
3	0.5	0.6	50	0.4587816385	0.2752293578	0.3109166667
3	0.5	0.6	80	0.2118265925	0.6330275229	1.7182166667
3	0.5	0.6	100	0.2025941125	0.6559633028	2.2036166667
3	0.5	0.6	120	0.2166624127	0.6467889908	2.7471166667
3	0.5	0.6	150	0.4611618863	0.2752293578	1.4589666667



Khi số neuron lớp ẩn vượt qua 80 thì tăng số neuron không tác động nhiều đến kết quả. Với mạng 150 neuron mỗi lớp ẩn, kết quả thấp hơn hẳn do không thoả mãn điều kiện dừng giống như phần (a).

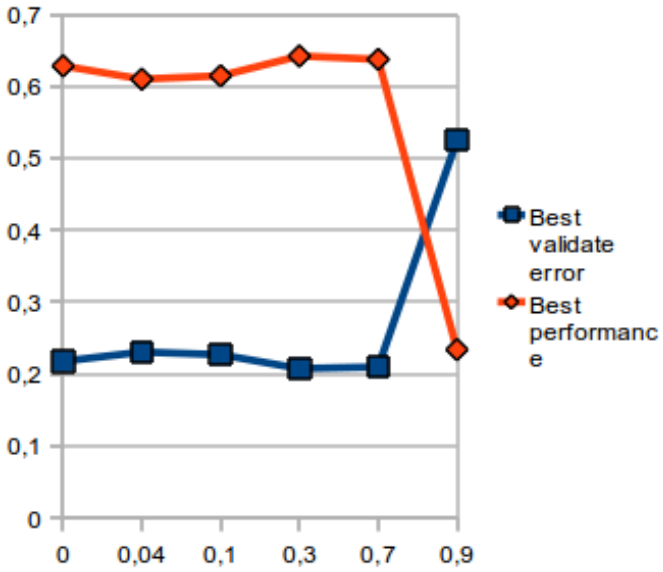
c. Tốc độ học

Layer count	Layer size	Momentum	Learning rate	Best validate error	Best performance	Training time
3	100	0.7	0.1	0.2061831533	0.623853211	3.9724166667
3	100	0.7	0.3	0.224606125	0.619266055	2.6471166667
3	100	0.7	0.5	0.2097530639	0.6376146789	2.1955333333
3	100	0.7	0.9	0.3250993898	0.4633027523	0.8474333333



d. Momentum

Layer count	Layer size	Learning rate	Momentum	Best validate error	Best performance	Training time
2	100	0.5	0	0.2177829205	0.628440367	1.4793666667
2	100	0.5	0.04	0.2305489015	0.6100917431	1.0493333333
2	100	0.5	0.1	0.2269169588	0.6146788991	1.9783166667
2	100	0.5	0.3	0.2076263935	0.6422018349	1.71595
3	100	0.5	0.7	0.2097530639	0.6376146789	2.1955333333
2	100	0.5	0.9	0.5251302293	0.2339449541	0.4365



Đồ thị cho thấy lựa chọn giá trị momentum quá cao khiến cho thuật toán bỏ sót một số giá trị tối ưu. Giá trị momentum tối ưu là 0,7.

e. Lựa chọn tham số

- Số neuron ở lớp ẩn: 100
- Số lớp ẩn: 3
- Tốc độ học: 0,5
- Momentum: 0,6

f. Kết quả huấn luyện

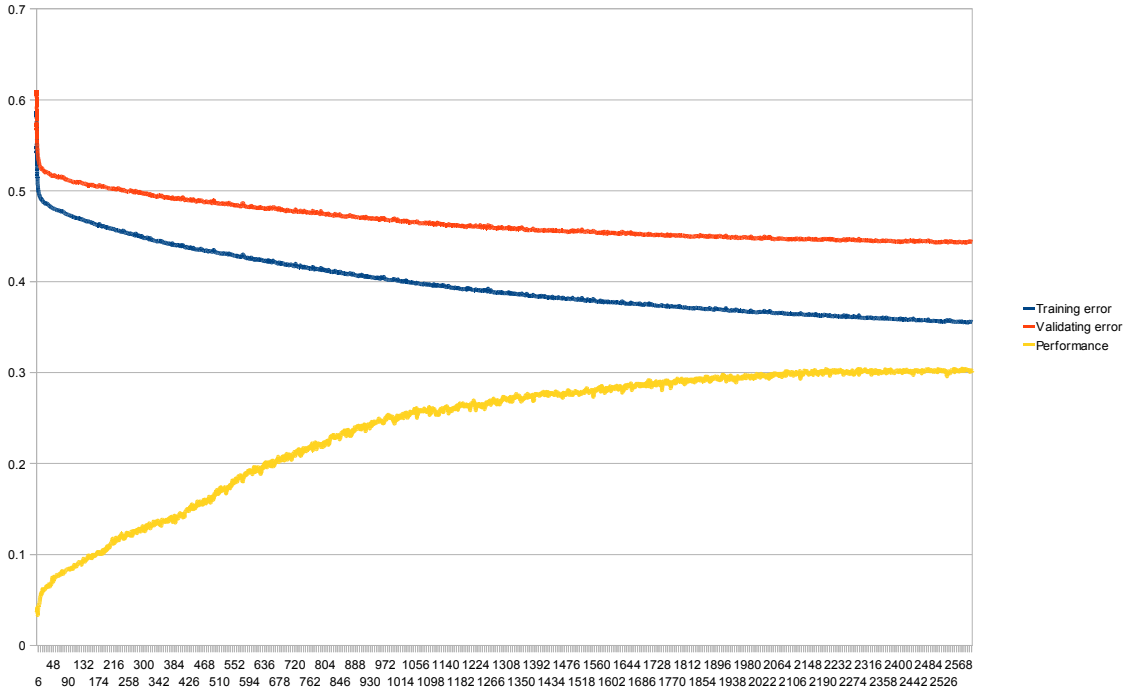
Thử nghiệm với bộ dữ liệu gồm một số kí tự có dấu (Ă, Ấ, Á, À, Ả v.v…) cho kết quả tạm ổn.

- Tỉ lệ nhận dạng đúng: 49%
- Tổng lỗi khi validate: ~0,33

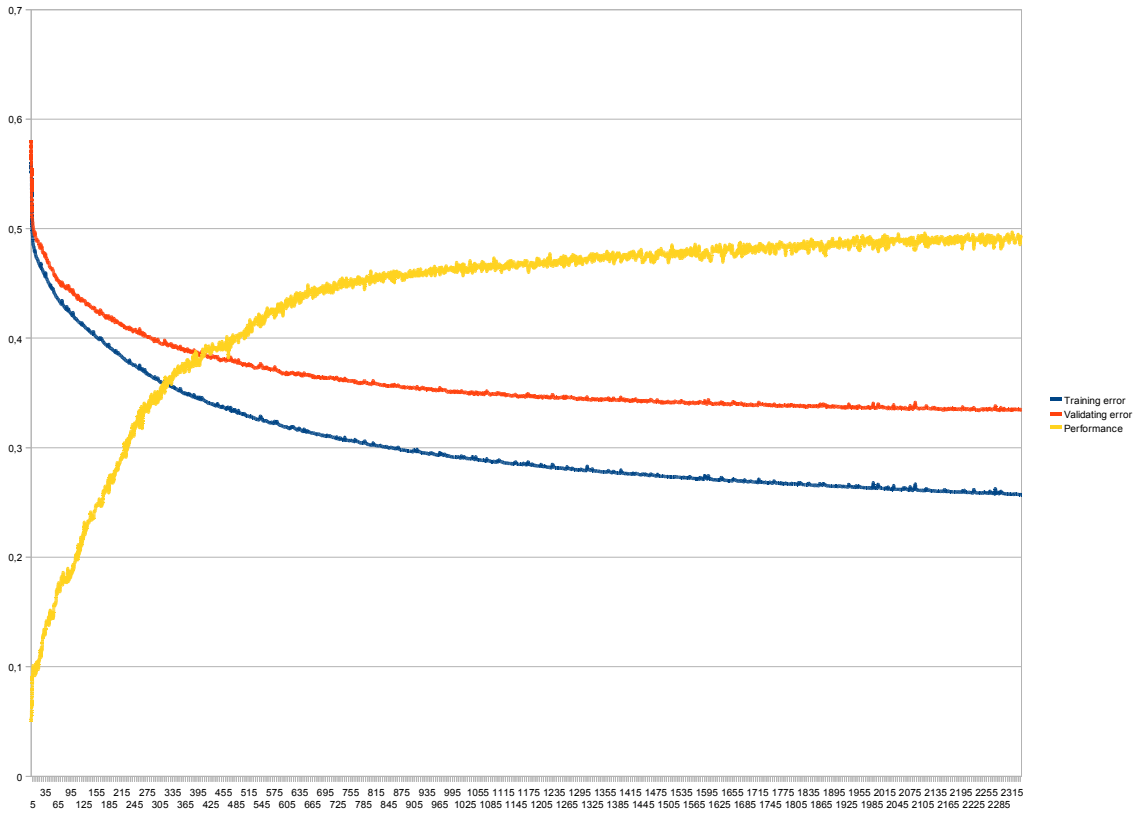
- Số bước lặp: 2338

Tuy nhiên với bộ dữ liệu đầy đủ chữ cái tiếng việt không dấu và có dấu (gần 200 kí tự), kết quả thu được rất thấp:

- Tỉ lệ nhận dạng đúng: ~30%
- Tổng lỗi trên bộ dữ liệu kiểm thử: ~0,45
- Số bước lặp: 2608



Hình 2Thử nghiệm với bộ dữ liệu đầy đủ



Hình 3 Thử nghiệm với bộ dữ liệu chỉ chứa một vài kí tự có dấu

4. GIỚI THIỆU VỀ PHẦN MỀM sapphireOCR

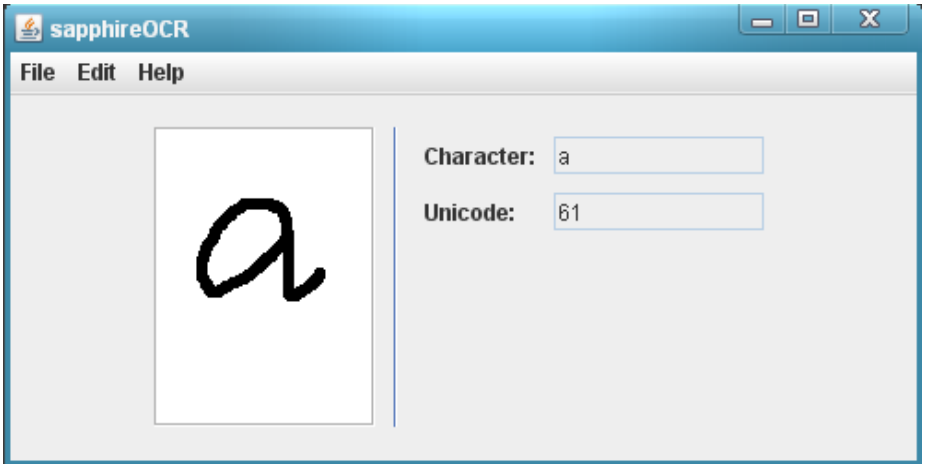
4.1. Hướng dẫn cài đặt

➢ Phần mềm sapphireOCR chạy trên nền Java Runtime Environment (JRE). Để chạy chương trình trước hết máy phải có JRE, download và cách cài đặt xem tại http://www.java.com/en/download/index.jsp.

➢ sapphireOCR là phần mềm không cần cài đặt. Sau khi giải nén ra thư mục, chỉ cần chạy file shortcut sapphireOCR.

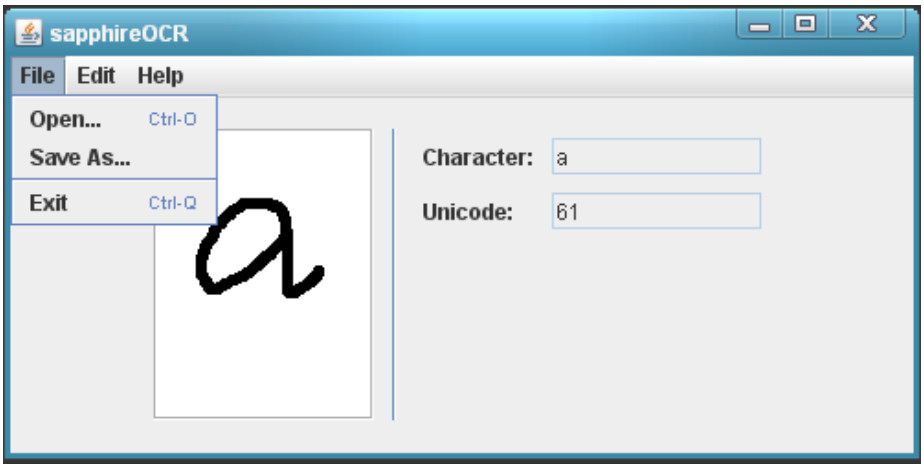
4.2. Hướng dẫn sử dụng

Sau khi làm theo hướng dẫn cài đặt, sau đây là giao diện chính của chương trình:

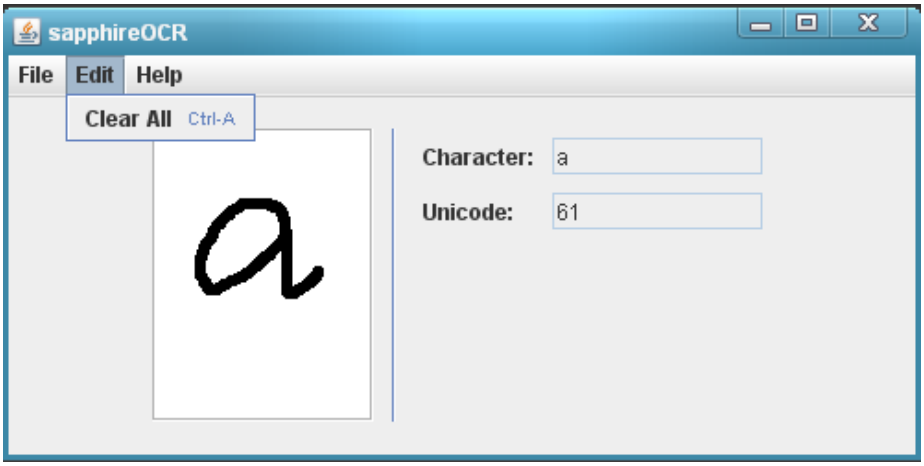


➢ Chương trình có một khung vẽ cho người sử dụng dùng chuột trái viết chữ, chữ nhận diện được sẽ được ghi ở phần Character và có mã Unicode (hexa) tương ứng.

➢ Ngoài ra, người sử dụng có thể load file ảnh bên ngoài qua menu File > Open và chỉnh sửa thêm tùy ý. Để lưu lại file ảnh vừa vẽ, chọn File > Save As.



➢ Để tẩy, dùng chuột phải thao tác thay cho chuột trái. Để xóa toàn bộ khung vẽ, chọn Edit > Clear All.



5. CÁC VẤN ĐỀ PHÁT SINH VÀ ĐỀ XUẤT

5.1. Kết quả nhận dạng thấp

Nguyên nhân có thể là do khâu tiền xử lí thực hiện chưa tốt. Để giải quyết vấn đề này cần cải thiện thuật toán tiền xử lí và khi kết hợp với chương trình nhận dạng tài liệu có thể đưa thêm một số thuật toán "đoán" dựa vào từ điển.

5.2. Tốc độ huấn luyện chậm

Khi thử nghiệm bộ dữ liệu đầy đủ, chương trình cần 112 phút để hoàn tất huấn luyện. Giải pháp: song song hoá thuật toán huấn luyện.

6. TÀI LIỆU VÀ MÃ NGUỒN SỬ DỤNG

6.1. Tài liệu tham khảo

[1] Thầy Nguyễn Nhật Quang, Slide bài giảng môn Trí tuệ nhân tạo, 2010

[2] Jeff Heaton – Introduction to Neural Networks with Java, 1st Ed

[3] Tom M. Mitchell, Machine learning, 1997

[4] Tinku Acharya, Image processing Principles and Applications

6.2. Mã nguồn

1. CannyEdgeDetector – Tom Gibara (Public domain)
2. Commons Lang – Apache Software Foundation (Apache License v2)
3. Commons IO – Apache Software Foundation (Apache License v2)
4. yamlbeans (New BSD License)