MỤC LỤC

DANH SÁCH CÁC HÌNH

LỜI CẢM ƠN

Sẽ không có khả năng cho em để hoàn thành đồ án đợt này nếu không có những sự giúp đỡ, ủng hộ của các thầy cô giáo, bạn bè và người thân. Em muốn dành một lời cảm ơn chân thành nhất tới tất cả và đặc biệt có bốn lời cảm ơn sau:

Đầu tiên, em xin gửi lời cảm ơn tới Thạc sĩ Ngô Trường Giang, người thầy đã hướng dẫn em trong đợt làm đồ án này. Sự đóng góp về tài liệu và đặc biệt là sự ân cần giúp đỡ, chỉ bảo của thầy đối với chúng em chính là nhân tố quyết định đôi với sự hoàn thành đồ án kịp thời của em.

Thứ hai, em xin gửi lời cảm ơn tới các thầy cô giáo trong bộ môn công nghệ thông tin. Sự dạy dỗ dìu dắt tận tình của các thầy, các cô trong suốt bốn năm học đã giúp em tiếp thu được kiến thức để có thể thực hiện đồ án này.

Thứ ba, em xin gửi lời cảm ơn tới giáo sư Trần Hữu Nghị - hiệu trưởng của trường đại học Dân Lập Hải Phòng. Nếu không có thầy thì sẽ không có chúng em - những sinh viên đã và sẽ tốt nghiệp của trường đại học Dân Lập Hải Phòng. Thầy đã đem lại cho chúng em một môi trường rèn luyện hiệu quả không chỉ về kiến thức mà còn về con người. Em tự hào là một phần tử trong môi trường đó.

Cuối cùng, em muốn nói lời cảm ơn tới bố mẹ em. Tuy đây là lần đầu tiên được nói ra, nhưng nó đã luôn ở trong lòng em từ lâu. Chính sự chăm sóc, ủng hộ vô điều kiện của bố mẹ đối với em đã là một động lực to lớn giúp em vượt qua những khó khăn để luôn quyết tâm thực hiện tốt những công việc của mình.

Hải Phòng, *ngày 5 tháng 7 năm 2010*

Sinh viên

Nguyễn Thành Công

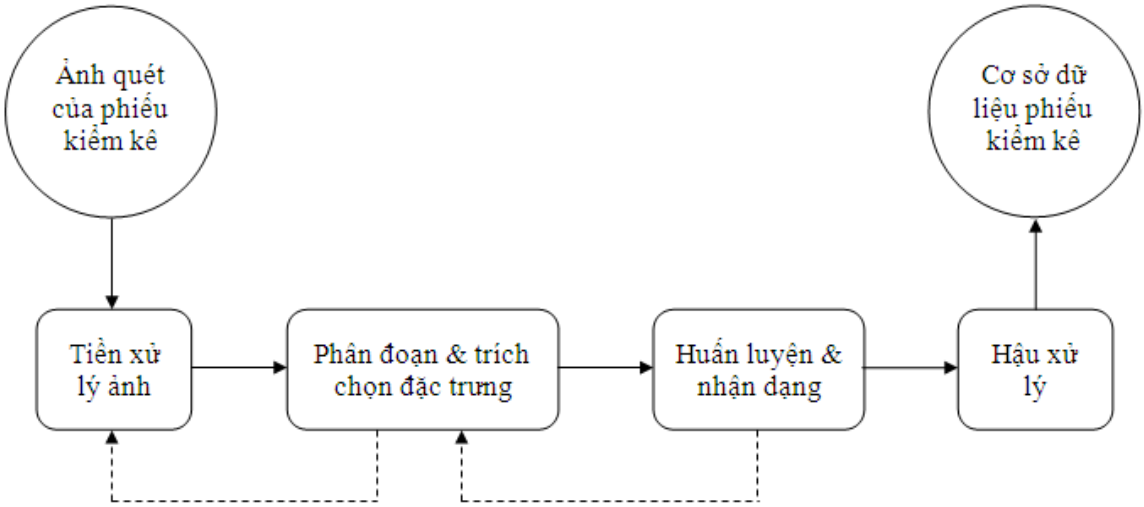
CHƯƠNG 1: GIỚI THIỆU

1.1 Phát biểu bài toán

Công ty TNHH UNIDEN thuộc khu công nghiệp Thiên Trường, thành phố Hải Dương chuyên sản xuất các linh kiện điện tử cho máy điện thoại. Hàng quý, nhân viên kho phải kiểm kê các sản phẩm sau đó tổng hợp để báo cáo cho công ty điều hành sản xuất. Mỗi quý nhân viên quản lý phải xử lý hàng ngàn phiếu bao gồm nhập dữ liệu, tổng hợp tính toán nhưng hiện tại công việc này đang được thực hiện thủ công nên mất rất nhiều thời gian. Đề tài này tập trung giải quyết khâu hỗ trợ nhập liệu tự động cho công việc trên tại công ty.

1.2 Hướng giải quyết

Từ phát biểu bài toán trên, có thể tóm tắt các công việc chính cần làm là: xây dựng chương trình nhận đầu vào là các ảnh phiếu kiểm kê sản phẩm, xác định vùng chứa dữ liệu, nhận dạng dữ liệu đó rồi cập nhật vào cơ sở dữ liệu. Dữ liệu cần nhận dạng ở đây bao gồm 10 chữ số in hoặc viết tay và 25 chữ cái in hoa (loại trừ O) viết tay. Để nhận dạng được các kí tự viết tay thì trước tiên ảnh đầu vào phải qua giai đoạn tiền xử lý là phân ngưỡng, có thể kết hợp với lọc nhiễu, chuẩn hóa kích cỡ. Tiếp theo chúng ta cần thực hiện bước phân đoạn để tìm ra các đối tượng trong ảnh, cụ thể ở đây là các kí tự. Vì ảnh phiếu phân vùng dữ liệu vào các hàng và các cột nên phân đoạn là việc tách ảnh phiếu vào các hàng, từ các hàng ta tách ra các cột, rồi từ các cột tách ra vùng ảnh chứa kí tự cần nhận dạng. Từ đó chúng ta trích ra vùng ảnh chỉ chứa kí tự cần nhận dạng, trích chọn đặc trưng của nó, rồi đưa vector đặc trưng vào mạng neuron đã qua huấn luyện cho nhận dạng. Gai đoạn cuối cùng sẽ là tổng hợp các kí tự được nhận dạng riêng lẻ thành dữ liệu để cập nhật vào cơ sở dữ liệu. Các bước cụ thể được thể hiện qua sơ đồ sau:



Hình 1.1: Sơ đồ hệ thống cập nhật phiếu.

Đầu vào của hệ thống là ảnh quét của phiếu kiểm kê sản phẩm. Trước tiên, một vài điều kiện cần phải được áp đặt lên việc ghi phiếu để quá trình phân đoạn cũng như nhận dạng diễn ra thuận lợi đó là:

- Viết chữ rõ ràng, không đứt đoạn, không chồng chéo.
- Chữ được viết đúng vị trí, đúng ô, không đè lên các đường bao quanh ô.
- Không dập xóa lên phiếu.

Sau khi đã có ảnh được quét đúng cách, hệ thống thực hiện lần lượt các bước như sau:

- **Tiền xử lý:** Ảnh sau khi quét thường có nhiễu, một phần nhiệm vụ của bước này sẽ là lọc nhiễu. Sau đó ảnh phải được biến đổi về ảnh nhị phân để tạo điều kiện cho phân đoạn ở bước tiếp theo.
- **Phân đoạn và trích chọn đặc trưng:** Tách ảnh đã qua tiền xử lý thành các hàng, mỗi hàng bao gồm các cột. Sau đó từ các cột sẽ tách ra kí tự cần nhận dạng, rồi trích chọn đặc trưng của nó.
- **Huấn luyện và nhận dạng:** Tiếp nhận vector đặc trưng của kí tự từ bước trước để đưa vào mạng neuron cho nhận dạng. Trước khi nhận dạng, quá trình huấn luyện được thực hiện trước dựa trên một tập mẫu có sẵn.
- **Hậu xử lý:** Tổng hợp các kí tự được nhận dạng riêng lẻ thành dữ liệu để cập nhật vào cơ sở dữ liệu.

1.3 Cấu trúc báo cáo

Báo cáo được tổ chức theo hướng từ lý thuyết đến thực tế, tức là nêu lý thuyết trước, rồi áp dụng lý thuyết để giải quyết bài toán sau. Báo cáo bao gồm 5 Chương, cụ thể là:

Chương 1: Mô tả bài toán đặt ra cũng như phương hướng giải quyết nó. Đưa ra sơ đồ thực hiện các bước cũng như trình bày tóm tắt các công việc cần làm trong mỗi bước.

Chương 2: Nêu các cơ sở lý thuyết được áp dụng để giải quyết bài toán. Lý thuyết gồm 2 phần: tiền xử lý và mạng neuron nhân tạo. Tiền xử lý sẽ nêu một số kĩ thuật xử lý ảnh liên quan để giải quyết khâu phân tích ảnh. Để nhận dạng được các kí tự, chúng ta cần một bộ phân lớp và mạng neuron nhân tạo đã được chọn. Chương sẽ giới thiệu tổng quan về mạng neuron cũng như mô hình mạng được áp dụng trong nhận dạng kí tự.

Chương 3: Trình bày việc áp dụng các cơ sở lý thuyết được giới thiệu trong Chương 2 để giải quyết bài toán đặt ra. Việc phân đoạn bao gồm tìm các hàng, tìm các cột để xác định kí tự trong ảnh được nêu ra chi tiết. Các bước thực hiện để giải quyết bài toán được trình bày lần lượt như sau: tiền xử lý ảnh đầu vào, xác định các kí tự cần nhận dạng trong ảnh, trích chọn đặc trưng kí tự, tiến hành nhận dạng.

Chương 4: Mô tả các chương trình được xây dựng và quá trình thực nghiệm.

Chương 5: Đánh giá những gì đã đạt được và nêu lên những hướng nghiên cứu tiếp theo.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Một số kĩ thuật xử lý ảnh liên quan

2.1.1 Lọc nhiễu

Ảnh được thu nhận qua máy quét thường có nhiễu. Đó là những chấm đen thường xuất hiện một mình trong ảnh, gây khó khăn cho mục đích sử dụng ảnh. Có nhiều bộ lọc sử dụng các toán tử không gian thực hiện loại bỏ nhiễu như lọc trung bình, trung vị, low-pass. Trong đó, lọc low-pass tỏ ra là hiệu quả hơn cả.

Lọc low-pass làm mượt các chuyển tiếp sắc trong mức xám và loại bỏ nhiễu. Các bộ lọc low-pass bỏ qua các tần số thấp và dừng các tần số cao. Trong khi đó các bộ lọc high-pass bỏ qua các tần số cao và dừng các tần số thấp. Hay nói cách khác, lọc low-pass làm giảm các thay đổi mức xám thường xuyên, còn lọc high-pass phóng đại các thay đổi mức xám thường xuyên. Vì thế khi kết hợp lọc high-pass ngay sau lọc low-pass sẽ khiến ảnh vừa không còn nhiễu, vừa sắc cạnh, cải thiện chất lượng ảnh.

Thực thi lọc low-pass và high-pass là tương tự nhau, dùng một mặt nạ 3 x 3 nhân chập với vùng 3 x 3 của ảnh, ngoại trừ một điểm khác là lọc low-pass sẽ chia cho một số nguyên là tổng các phần tử của mặt nạ sau khi nhân chập.

$$\frac{1}{6} \cdot \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\frac{1}{9} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{10} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Hình 2.1: Các mặt nạ nhân chập bộ lọc low-pass.

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

$$\begin{matrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{matrix}$$

Hình 2.2: Các mặt nạ nhân chập bộ lọc high-pass.

2.1.2 Phân ngưỡng

Phân ngưỡng là một kĩ thuật dùng để biến đổi ảnh về ảnh nhị phân bao gồm chỉ hai giá trị là 0 hoặc 1.

Để thực hiện phân ngưỡng thì có 2 phương pháp: thủ công và tự động. Với phương pháp thủ công, một ngưỡng cố định được chọn trước còn trong phương pháp tự động ngưỡng sẽ được chọn tự động. Trước khi phân ngưỡng ảnh phải được biến đổi về ảnh xám. Nếu coi ảnh là một ma trận 2 chiều, thì:

$$g(x,y) = \begin{cases} 0 & if\ g(x,y) < \mu \\ 1 & else \end{cases}$$

Trong đó, *g(x, y)* là giá trị mức xám tại tọa độ *(x, y)*, μ là ngưỡng.

2.1.3 Dò và chỉnh nghiêng

Do nhiều yếu tố khác nhau mà ảnh không tránh khỏi bị nghiêng trong suốt quá trình quét ảnh. Tùy theo mức độ mà góc nghiêng có thể rất cao đến nỗi mà không thể áp dụng được các thuật toán phân tích ảnh. Do vậy cần phải phát hiện và chỉnh nghiêng cho ảnh trước khi tiến hành xử lý ở những bước sau.

Tư tưởng cơ bản để loại bỏ nghiêng là như sau:

- Tìm ra các dòng tham chiếu trong ảnh.
- Tính góc θ của các dòng.

- Tính góc nghiêng *skew* là trung bình của các góc θ.
- Xoay ảnh bởi một góc *-skew*.

Các dòng được dò tìm với thuật toán Hough. Mỗi điểm trong ảnh có thể nằm trên vô số các dòng. Để tìm ra các dòng tham chiếu, chúng ta thực hiện thủ tục bỏ phiếu, tức là đối với mỗi dòng mà đi qua một điểm thì chúng ta bỏ phiếu điểm đó cho dòng. Các dòng với số điểm cao nhất sẽ là các dòng tham chiếu.

Trước tiên chúng ta cần tham số hóa một dòng. Một dòng có thể được tham số hóa như sau:

$$y = m*x + d \quad (1)$$

Với *m* là độ dốc và *d* là độ lệch. Chúng ta không quan tâm đến độ dốc mà chỉ quan tâm đến góc. Góc θ của dòng phải thỏa mãn:

$$m = \tan(\theta) = \frac{\sin(\theta)}{\cos(\theta)} \quad (2)$$

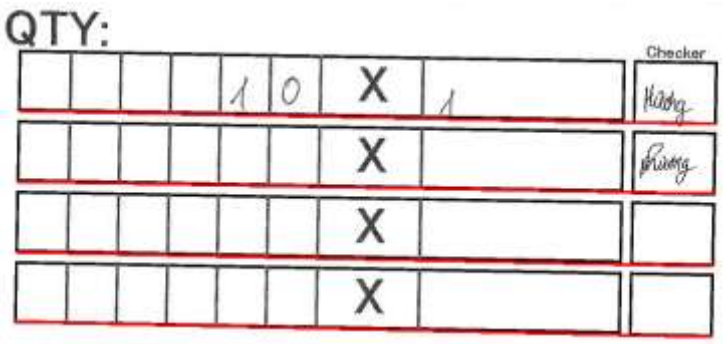
Từ (1) và (2) ta được:

$$y = \frac{\sin(\theta)}{\cos(\theta)} * x + d \quad \Leftrightarrow \quad y*\cos(\theta) - x*\sin(\theta) = d$$

Vì chúng ta không thể tìm kiếm trong một không gian tham số vô hạn nên chúng ta phải định nghĩa một không gian rời rạc với θ được lấy trong khoảng [-20, 20] với bước nhảy là 0.2. Thủ tục bỏ phiếu diễn ra như sau:

- Duyệt từ vị trí y = 0 cho tới vị trí height – 1.
- Ứng với mỗi vị trí y, chúng ta duyệt từ vị trí x = 0 cho tới vị trí width – 1.
- Nếu điểm (x, y) là đen, thì với θ trong khoảng [-20, 20], với mỗi bước nhảy, chúng ta tính:
- d = Round(y*cos(θ) – x*sin(θ)).
- Hough(Round(θ*5), d) += 1.

Để tiết kiệm thời gian tính toán, số các điểm bỏ phiếu được giảm đi. Để loại bỏ nghiêng, chỉ dòng bottom là quan trọng như trong hình sau:



Hình 2.3: Các dòng bottom là các dòng tham chiếu cần tìm.

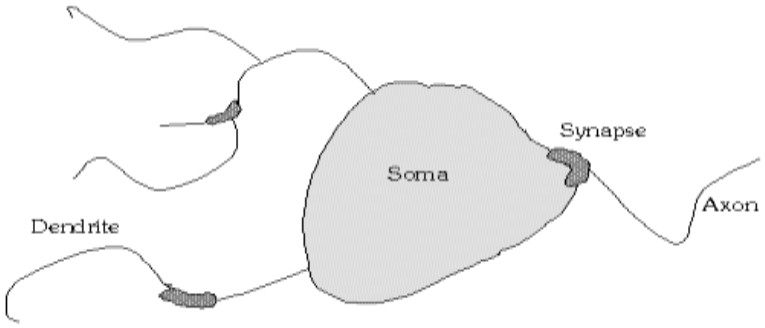
Các điểm trên dòng bottom có một điểm chung là lân cận ngay phía dưới của nó là một điểm trắng. Vì thế, chúng ta chỉ bỏ phiếu cho các điểm (x, y) nếu nó thỏa mãn:

- Điểm (x, y) là đen.
- Điểm lân cận dưới (x, y + 1) là trắng.

2.2 Tổng quan mạng neuron

2.2.1 Neuron sinh học

Một neuron sinh học bao gồm những thành phần chính sau: Dendrite, Soma, Synapse, Axon như hình 2.4.



Hình 2.4: Mô hình neuron sinh học.

Soma là thân của neuron. Các dendrite là các dây mảnh, dài, gắn liền với soma, chúng truyền dữ liệu (dưới dạng xung điện thế) đến cho soma xử lý. Bên trong soma các dữ liệu đó được tổng hợp lại, có thể xem gần đúng sự tổng hợp ấy như là một phép lấy tổng tất cả các dữ liệu mà neuron nhận được.

Một loại dây dẫn tín hiệu khác cũng gắn với soma là các axon. Khác với dendrite, axons có khả năng phát các xung điện thế, chúng là các dây dẫn tín hiệu từ

neuron đi các nơi khác. Chỉ khi nào điện thế trong soma vượt quá một giá trị ngưỡng nào đó thì axon mới phát một xung điện thế, còn nếu không thì nó ở trạng thái nghỉ.

Axon nối với các dendrite của các neuron khác thông qua những mối nối đặc biệt gọi là synapse. Khi điện thế của synapse tăng lên do các xung phát ra từ axon thì synapse sẽ nhả ra một số chất hoá học (neurotransmitters). Các chất này "mở cửa" trên dendrite để cho các ions truyền qua. Chính dòng ions này làm thay đổi điện thế trên dendrite, tạo ra các xung dữ liệu lan truyền tới các neuron khác.

Có thể tóm tắt hoạt động của một neuron như sau: neuron lấy tổng tất cả các điện thế vào mà nó nhận được, và phát ra một xung điện thế nếu tổng ấy lớn hơn một ngưỡng nào đó. Các neuron nối với nhau ở các synapse. Synapse được gọi là mạnh khi nó cho phép truyền dẫn dễ dàng tín hiệu qua các neuron khác. Ngược lại, một synapse yếu sẽ truyền dẫn tín hiệu rất khó khăn.

Các synapse đóng vai trò rất quan trọng trong sự học tập. Khi chúng ta học tập thì hoạt động của các synapse được tăng cường, tạo nên nhiều liên kết mạnh giữa các neuron. Có thể nói rằng người nào học càng giỏi thì càng có nhiều synapse và các synapse ấy càng mạnh mẽ, hay nói cách khác, thì liên kết giữa các neuron càng nhiều, càng nhạy bén.

2.2.2 Mạng neuron nhân tạo

Khái niệm

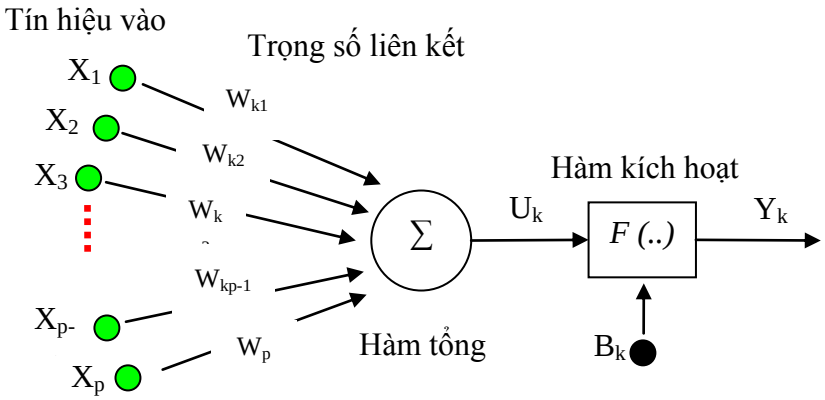
Mạng neuron nhân tạo (artificial neural network) là mạng các phần tử (các neuron) kết nối với nhau thông qua các liên kết (các trọng số) để thực hiện một công việc cụ thể nào đó. Khả năng xử lý của mạng neuron được hình thành thông qua quá trình hiệu chỉnh trọng số liên kết giữa các neuron, nói cách khác là học từ tập hợp các mẫu huấn luyện. Mạng được mô phỏng dựa trên cấu tạo hệ thần kinh của con người.

Cấu trúc một neuron

Với bản chất là một mô hình mô phỏng đơn giản neuron sinh học, neuron nhân tạo cũng thực hiện nhiệm vụ của mình thông qua các thao tác: nhận đầu vào từ các neuron trước nó, xử lý đầu vào bằng cách nhân mỗi đầu vào này với trọng số liên kết tương ứng và tính tổng các tích thu được rồi đưa qua một hàm kích hoạt,

sau đó gửi kết quả cuối cùng cho các neuron tiếp theo hoặc đưa ra đầu ra. Cứ như vậy các neuron này hoạt động phối hợp với nhau tạo thành hoạt động chính của mạng neuron.

Chúng ta có thể hình dung rõ ràng hơn quy trình xử lý thông tin của một neuron qua cấu trúc của nó được thể hiện trong hình 2.5.



Hình 2.5: Cấu trúc một neuron.

Trong đó:

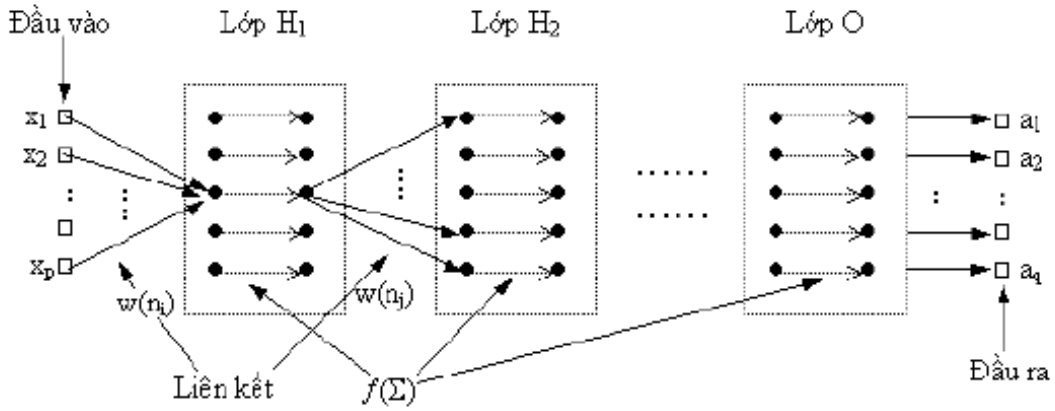
- $(X_1, X_2, \ldots, X_p)$, với $p \geq 1$: là các tín hiệu đầu vào của neuron. Các tín hiệu này có thể là đầu ra của các neuron trước nó hoặc đầu vào ban đầu của mạng và thường được đưa vào dưới dạng một vector p chiều.
- $(W_{k1}, W_{k2}, \ldots, W_{kp})$ là tập các trọng số liên kết của neuron k với p đầu vào tương ứng $(X_1, X_2, \ldots, X_p)$. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học của mạng.
- Σ là hàm tổng trên một neuron, dùng để tính tổng các giá trị kích hoạt lên neuron đó. Thông thường, đây là tổng của các tích giữa đầu vào với trọng số liên kết tương ứng của neuron.
- U_k là tổng các giá trị kích hoạt lên neuron thứ k, giá trị này chính là đầu ra của hàm tổng.
- B_k là hệ số bias (còn gọi là ngưỡng) của neuron thứ k, giá trị này được dùng như một thành phần phân ngưỡng trên hàm kích hoạt và cũng được cập nhật liên tục trong quá trình học của mạng.

- F() là hàm kích hoạt (hay hàm truyền, hàm nén). Hàm kích hoạt được dùng để giới hạn phạm vi đầu ra của mỗi neuron. Đối số của hàm là giá trị hàm tổng và ngưỡng B_k. Thông thường, phạm vi đầu ra của mỗi neuron được giới hạn trong khoảng [0, 1] hoặc [-1, 1]. Như vậy miền giá trị của các hàm kích hoạt cũng là một trong hai khoảng trên. Có rất nhiều hàm kích hoạt thường được dùng, việc lựa chọn hàm kích hoạt nào cho phù hợp tuỳ thuộc vào từng bài toán.
- Y_k là tín hiệu đầu ra của neuron thứ k, mỗi neuron thường có một đầu ra và tối đa là một đầu ra. Giá trị Y_k được tính theo công thức:

$$Y_k = F(U_k + B_k) \text{ với } U_k = \sum_{j-1}^{p} W_{kj} * X_j$$

Cấu trúc mạng neuron

Một mạng neuron có thể có nhiều lớp và ít nhất phải có một lớp đó là lớp ra (lớp vào thường không được tính). Mỗi lớp có một hoặc nhiều neuron. Cấu trúc tổng quát của mạng neuron được thể hiện trong hình 2.6 dưới đây:



Hình 2.6: Cấu trúc chung của mạng neuron.

Trong đó:

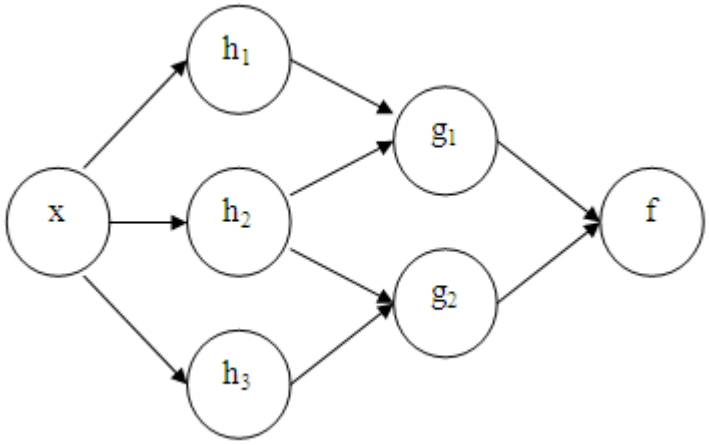
- Đầu vào của mạng là vector có kích thước p: (x_1, x_2, …, x_p) và đầu ra là vector (a_1, a_2, …, a_q) có kích thước q. Lớp ẩn đầu tiên là lớp H_1, sau đó đến lớp ẩn thứ hai H_2, tiếp tục như vậy cho đến lớp ẩn cuối cùng rồi lớp đầu ra.
- Các neuron trong các lớp có cấu trúc như trên hình 2.6, liên kết giữa các neuron giữa các lớp có thể là liên kết đầy đủ (mỗi neuron thuộc lớp sau liên

kết với tất cả neuron ở lớp trước nó) hoặc liên kết chọn lọc (mỗi neuron thuộc lớp sau chỉ liên kết với một số neuron ở lớp trước nó).

Đầu ra của lớp trước chính là đầu vào của lớp ngay sau nó.Với cấu trúc như vậy, hoạt động của mạng neuron diễn ra như sau: đầu tiên, vector đầu vào được lan truyền qua lớp H_1. Tại lớp này, mỗi neuron nhận vector đầu vào rồi xử lý (tính tổng có trọng số của các đầu vào rồi cho qua hàm kích hoạt) và cho ra kết quả tương ứng. Đầu ra của lớp H_1 chính là đầu vào của lớp H_2, do đó sau khi lớp H_1 cho kết quả ở đầu ra của mình thì lớp H_2 nhận được đầu vào và tiếp tục quá trình xử lý. Cứ như vậy cho tới khi thu được đầu ra sau lớp O, đầu ra này chính là đầu ra cuối cùng của mạng.

2.2.3 Xây dựng mạng

Về cơ bản chúng ta có thể hiểu mạng neuron là một đồ thị có hướng như hình 2.7. Trong đó các đỉnh của đồ thị là các neuron và các cạnh của đồ thị là các liên kết giữa các neuron.



Hình 2.7: Một đồ thị có hướng đơn giản.

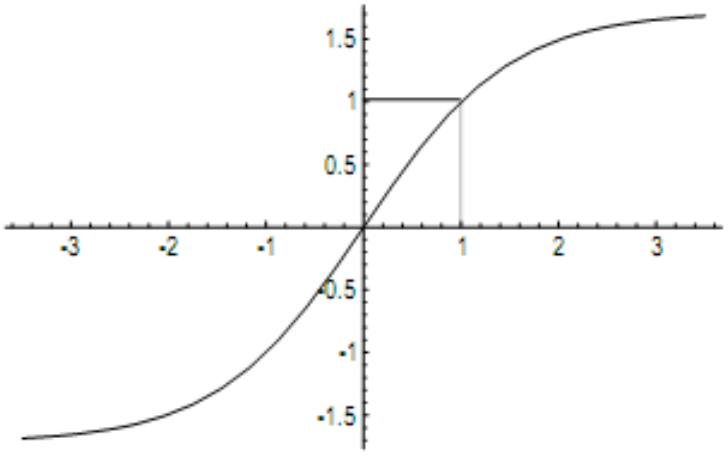
Vì vậy để xây dựng một mạng neuron chúng ta xây dựng một đồ thị có hướng: số đỉnh của đồ thị bằng số neuron trong mạng, giá trị của các cạnh chính là trọng số liên kết neuron. Về việc lựa chọn số lượng neuron cho từng lớp cũng như số lượng lớp ẩn được quyết định dựa theo đặc trưng của từng bài toán đề ra. Ngoài ra, việc lựa chọn các giá trị ban đầu cho hàm kích hoạt, các trọng số hay các giá trị đích có tác động đáng kể lên quá trình huấn luyện của mạng. Phần sau là một số thủ thuật trong việc lựa chọn các giá trị này nhằm cải thiện hiệu năng của mạng.

Hàm kích hoạt

Sự lựa chọn một hàm kích hoạt tốt là một phần quan trọng trong thiết kế mạng neuron. Hàm kích hoạt nên được chọn nếu nó cân xứng quanh gốc tọa độ, và mạng nên được huấn luyện tới một giá trị thấp hơn các giới hạn của hàm.

Với lý do trên, thì không nên chọn một hàm như là hàm logistic $F(y) = \frac{1}{1+e^{-y}}$ bởi vì nó không cân xứng: giá trị của nó tiếp cận +1 khi tăng y, nhưng chỉ tiếp cận 0 khi giảm y.

Một lựa chọn tốt cho hàm kích hoạt là hàm hyperbolic tangent, $F(y) = \tanh(y)$. Nó là lựa chọn tốt vì nó hoàn toàn cân xứng như trong đồ thị sau:



Hình 2.8: Đồ thị hàm tanh.

Lý do khác, hàm tanh là một lựa chọn tốt vì dễ dàng tính được đạo hàm của nó:

$$x = F(y) = \tanh(y) = \frac{\sinh(y)}{\cosh(y)}$$

Với x là đầu ra, y là giá trị kích hoạt của neuron thì:

$$\frac{dF}{dy} = \frac{d}{dy}\left(\frac{\sinh(y)}{\cosh(y)}\right) = \frac{\cosh^2(y) - \sinh^2(y)}{\cosh^2(y)}$$

$$\frac{dF}{dy} = 1 - \tanh^2(y)$$

Vì $x = \tanh(y)$, nên kết quả là:

$$\frac{dF}{dy} = 1 - x^2$$

Từ kết quả ta thấy rằng tính đạo hàm chỉ cần dựa vào đầu ra mà không cần biết đầu vào. Từ hai lý do trên thì hàm tanh nên được chọn là hàm kích hoạt cho mạng. Khi được chọn thì nên dùng một phiên bản chỉ tính xấp xỉ của nó bằng cách dùng một hệ số đa thức ví dụ như sau:

$$F(y) = 1.7159 \cdot \tanh\left(\frac{2}{3}y\right)$$

Lý do cho khuyến nghị này là để hạn chế tính phức tạp tính toán của hàm tanh [2].

Khởi tạo các trọng số

Các giá trị trọng số ban đầu có thể có một tác động đáng kể lên quá trình huấn luyện. Các trọng số nên được chọn ngẫu nhiên nhưng theo một cách mà sigmoid được kích hoạt chủ yếu trong miền tuyến tính của nó. Nếu các trọng số rất lớn thì sigmoid sẽ bão hòa, dẫn đến các gradient nhỏ khiến cho việc học diễn ra chậm. Nếu các trọng số rất nhỏ thì các gradient cũng sẽ rất nhỏ. Các trọng số có thuận lợi khi (1) các gradient đủ lớn để việc học có thể tiến triển và (2) mạng sẽ học phần ánh xạ tuyến tính trước khi học phần không tuyến tính khó hơn [2].

Cho rằng sự phân bố các đầu ra của mỗi nút có một độ lệch chuẩn (σ) xấp xỉ bằng 1. Để đạt được một độ lệch chuẩn gần tới 1 tại đầu ra của lớp ẩn đầu tiên chúng ta dùng hàm kích hoạt như đề cập ở trên với điều kiện đầu vào của hàm cũng có độ lệch chuẩn $\sigma_y = 1$. Giả sử các đầu vào, y_i, tới một đơn vị không bị ràng buộc với phương sai 1, thì độ lệch chuẩn của đơn vị sẽ là:

$$\sigma_y = \left(\sum w_{ij}^2\right)^{1/2}$$

Bởi vậy, để đảm bảo rằng σ_y xấp xỉ bằng 1, các trọng số nên được rút ra ngẫu nhiên từ một sự phân bố với trung bình 0 và một độ lệch chuẩn được tính bởi:

$$\sigma_w = m^{-1/2}$$

Ở đây *m* là số các đầu vào tới đơn vị.

2.2.4 Huấn luyện mạng

Để huấn luyện mạng neuron thì lan truyền ngược là thuật toán học phổ biến và hiệu quả nhất. Tư tưởng chính là tính toán giá trị đầu ra tại mỗi neuron của lớp vào cho tới lớp ra (lan truyền xuôi), sau đó tính lỗi tại mỗi neuron của lớp ra rồi lan truyền lỗi qua các lớp (lan truyền ngược). Quá trình chi tiết diễn ra như sau:

- Bước 1: Khởi tạo tất cả trọng số, bias tới các giá trị ngẫu nhiên nhỏ. Bước này xác định điểm bắt đầu trên bề mặt lỗi cho phương thức giảm gradient mà vị trí của nó có thể quyết định cho sự hội tụ của mạng.
- Bước 2: Lan truyền xuôi mẫu đầu tiên của tập huấn luyên từ lớp vào qua các lớp ẩn cho tới lớp ra, trong đó mỗi neuron cộng các đầu vào được áp trọng số lại với nhau, rồi truyền tổng đó tới các neuron trong lớp kế tiếp.
- Bước 3: Tính toán sai khác giữa đầu ra thật sự của mỗi neuron lớp ra và đầu ra mong muốn tương đương của nó. Đây là lỗi được kết hợp với mỗi neuron lớp ra.
- Bước 4: Lan truyền ngược lỗi này xuyên qua mỗi kết nối bằng cách dùng luật học lan truyền ngược và từ đó xác định giá trị mà mỗi trọng số phải được thay đổi để giảm lỗi tại lớp ra.
- Bước 5: Điều chỉnh mỗi trọng số bằng cách cập nhật trọng số riêng của nó.
- Bước 6: Đưa vào mẫu tiếp theo và thực hiện lan truyền xuôi. Lặp lại bước 2-6 cho tới khi một tiêu chuẩn dừng nhất định được với tới.

Lan truyền ngược

Lan truyền ngược là một quá trình lặp bắt đầu với lớp cuối cùng và di chuyển ngược qua các lớp cho tới khi lớp đầu tiên được với tới. Giả sử cho mỗi lớp, chúng ta biết lỗi trong đầu ra của lớp. Nếu chúng ta biết lỗi của đầu ra, thì không khó để tính các thay đổi cho các trọng số, để mà giảm lỗi đó. Nhưng vấn đề là chúng ta chỉ có thể quan sát được lỗi trong đầu ra của lớp cuối cùng.

Lan truyền ngược cho ta một cách để xác định lỗi trong đầu ra của một lớp trước đó dựa vào đầu ra của lớp hiện thời. Bởi vậy đó là một quá trình lặp: bắt đầu ở lớp cuối cùng và tính thay đổi trong các trọng số cho lớp cuối cùng. Rồi tính lỗi trong đầu ra của lớp trước đó.

Để bắt đầu quá trình trên, đầu tiên ta phải tính được đạo hàm riêng của lỗi do một mẫu ảnh đầu vào gây ra đối với các đầu ra của các neuron tại lớp cuối cùng. Lỗi đó được tính như sau:

$$E_n^P = \frac{1}{2} \cdot \sum \left(x_n^i - T_n^i \right)^2 \quad (1)$$

Trong đó:

- E_n^P là lỗi gây ra bởi mẫu *P* tại lớp cuối cùng *n.*
- T_n^i là đầu ra mong muốn tại lớp cuối cùng.
- x_n^i là giá trị đầu ra thực tế tại lớp cuối cùng.

Từ phương trình (1), lấy đạo hàm riêng, ta được:

$$\frac{\partial E_n^P}{\partial x_n^i} = x_n^i - T_n^i \quad (2)$$

Phương trình (2) cho chúng ta một giá trị bắt đầu cho qua trình lan truyền ngược. Chúng ta dùng các giá trị số cho các đại lượng ở vế phải của phương trình (2) để tính các giá trị số cho đạo hàm. Dùng các giá trị số này của đạo hàm, chúng ta tính được các giá trị số cho những thay đổi trong các trọng số này thông qua hai phương trình (3) và (4) sau:

$$\frac{\partial E_n^P}{\partial y_n^i} = G(x_n^i) \cdot \frac{\partial E_n^P}{\partial x_n^i} \quad (3)$$

Ở đây, $G(x_n^i)$ là đạo hàm của hàm kích hoạt.

$$\frac{\partial E_n^P}{\partial w_n^{ij}} = x_{n-1}^i \cdot \frac{\partial E_n^P}{\partial y_n^i} \quad (4)$$

Từ phương trình (2) và (3), chúng ta tính được lỗi cho lớp trước:

$$\frac{\partial E_{n-1}^P}{\partial x_{n-1}^k} = \sum w_n^{ik} \cdot \frac{\partial E_n^P}{\partial y_n^i} \quad (5)$$

Các giá trị chúng ta đạt được từ phương trình (5) được dùng như là các giá trị bắt đầu cho các phép tính ở lớp nằm ngay trước. Hay nói cách khác, chúng ta lấy các giá trị đạt được từ phương trình (5), và dùng chúng trong một sự lặp lại các phương trình (3), (4) và (5) cho lớp nằm ngay trước.

Trong khi đó, các giá trị từ phương trình (4) cho chúng ta biết cần bao nhiêu để thay đổi các trọng số trong lớp hiện thời. Cụ thể, chúng ta cập nhật giá trị của mỗi trọng số theo công thức sau:

$$(w_n^{ij})_{new} = (w_n^{ij})_{old} - eta \cdot (\frac{\partial E_n^P}{\partial w_n^{ij}}) \quad (6)$$

Ở đây, eta là "hệ số học", là một số nhỏ như 0.0005 và giảm dần trong suốt quá trình huấn luyện.

2.3 Nhận dạng kí tự dùng mạng neuron

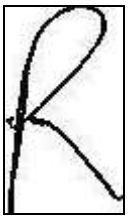
2.3.1 Trích chọn đặc trưng

Có nhiều phương pháp trích chọn đặc trưng khác nhau, từ đơn giản đến phức tạp. Đối với bài toán đặt ra, phương pháp đơn giản nhất đã được chọn. Đó là việc biến đổi ảnh kí tự thành một vector chứa các giá trị thích hợp. Trước đó thì ảnh phải được chuẩn hóa về một kích thước phù hợp.

Các bước thực hiện lần lượt như sau:

- Bước 1: Co giãn ảnh kí tự về kích thước m x n bằng thuật toán co giãn ảnh với tỉ lệ k, tỉ lệ này được tính theo tỉ lệ giữa chiều rộng của ảnh trên chiều rộng của kích thước cố định, hoặc theo tỉ lệ giữa chiều cao của ảnh trên chiều cao của kích thước cố định. Tùy theo giá trị chiều rộng và cao của ảnh chúng ta sẽ chọn tỉ lệ của chiều lớn nhất.
- Bước 2: Đưa ảnh về kích thước cố định (chuyển ảnh vào khuôn), ở trong đề tài này kích thước cố định của ảnh được chọn để đưa vào đó là một ma trận vuông 32 x 32. Sau khi đã co giãn ảnh theo tỉ lệ k, thì ảnh thu được có 1 chiều giá trị bằng 32, và một chiều có giá trị nhỏ hơn 32 do chúng ta đã co giãn theo tỉ lệ của chiều lớn nhất trên kích thước cố định. Và công việc bây giờ là chúng ta sẽ đặt ảnh đó vào khuôn, sao chép ảnh ký tự đã co giãn vào chính giữa khuôn.
- Bước 3: Tiến hành mã hoá ảnh đã được đặt vào khuôn (kích thước 32 x 32) về các giá trị -1 và 1, bằng cách tạo một ma trận tương ứng với kích thước của ảnh, và tại mỗi vị trí tương ứng nếu điểm ảnh là điểm đen thì tại vị trí đó là 1, còn nếu là điểm trắng thì ở vị trí đó sẽ là -1.

Kết quả minh họa bằng ví dụ sau:

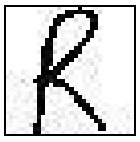


Hình 2.9: Ảnh kí tự đầu vào có kích thước 59 x 104.

B1: Co giãn

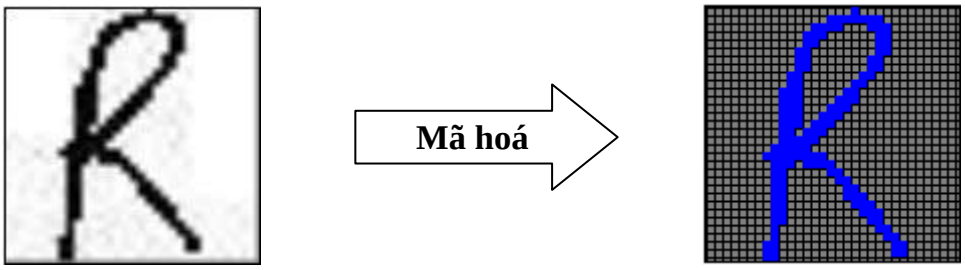
- Ta lấy tỉ lệ co giãn của chiều lớn nhất k = 104 / 32 = 3.25
- Như vậy kích thước mới của ảnh là:
- nW = 59 / 3.25 ≈ 18, nH = 104/3.25 = 32 : 18 x 32

B2: Đặt ảnh vào khuôn



Hình 2.10: Ảnh được co giãn có kích thước 32 x 32.

B3: Mã hóa



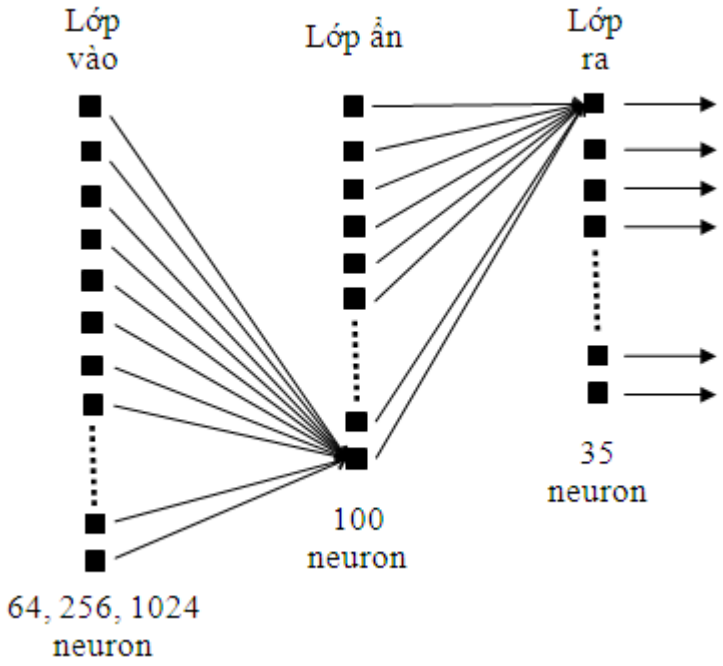
Hình 2.11: Ảnh được mã hoá.

Kết quả thu được đó là một ma trận vuông 32 x 32 chứa các giá trị -1 và 1 là thông tin của ảnh kí tự đầu vào. Vì mạng sẽ được huấn luyện theo phương pháp multiscale nên chúng ta cần thêm các ma trận 16 x 16 và 8 x 8. Đầu tiên, chúng ta sẽ biến đổi ảnh lớn về ảnh nhỏ bằng cách lấy trung bình 4 giá trị pixel của ảnh lớn thành giá trị pixel cho ảnh nhỏ. Sau đó tiến hành mã hóa như trên. Cuối cùng, các ma trận sẽ được biến đổi thành các vector làm đầu vào của mạng cho huấn luyện.

2.3.2 Xây dựng mạng

Mạng bao gồm 3 lớp:

- Lớp vào: số neuron tùy thuộc vào ảnh đầu vào độ phân giải của ảnh, 8 x 8, 16 x 16 hay 32 x 32.
- Lớp ẩn: số neuron được chọn tùy ý sao cho phù hợp với số lượng các mẫu trong tập huấn luyện.
- Lớp ra: số neuron cố định là 35 (25 chữ cái và 10 chữ số). Neuron đầu ra đầu tiên tương ứng với số 0, tiếp theo là 1, 2 … A, B, C, …, X, Y, Z.



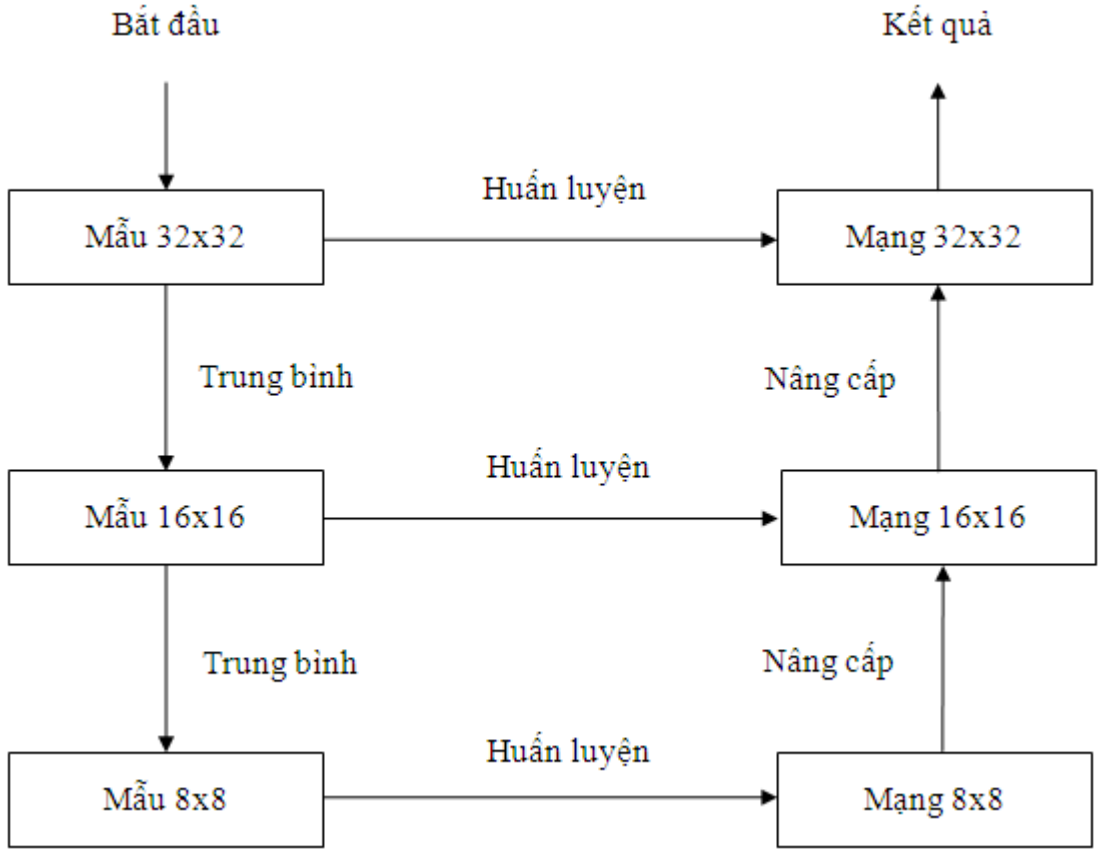
Hình 2.12: Mô hình mạng neuron được xây dựng.

2.3.3 Huấn luyện mạng

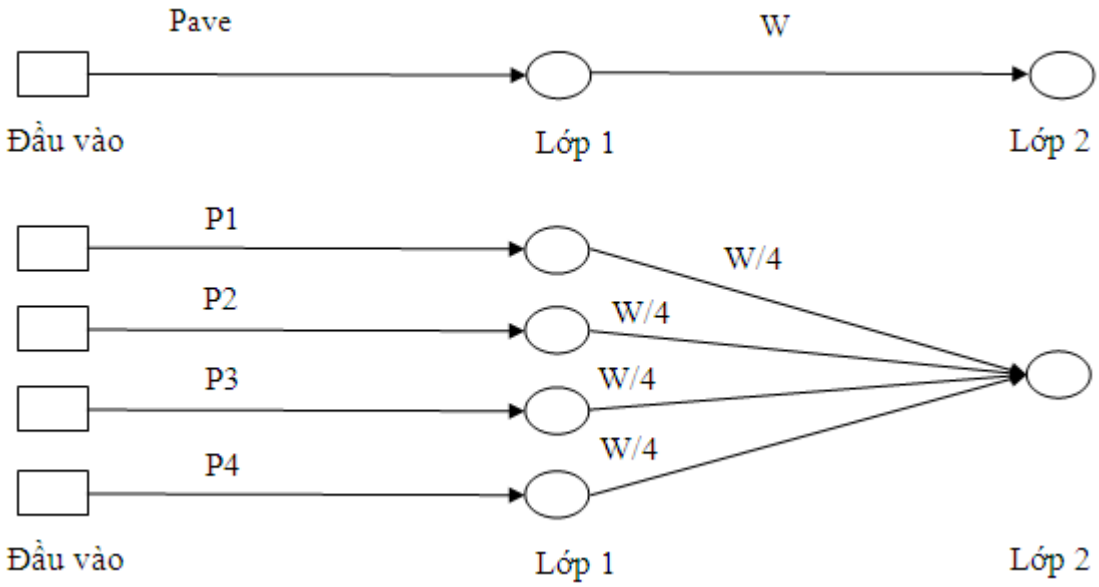
Mạng neuron lan truyền ngược yêu cầu thời gian dài để nhớ tất cả vector có khả năng được đưa vào mạng. Tuy nhiên, luôn có khả năng là mạng sẽ đưa ra kết quả sai do bởi khả năng tổng quát hóa nghèo nàn. Vấn đề đó có thể được khắc phục bằng cách dùng kĩ thuật huấn luyện multiscale [3].

Việc huấn luyện bao gồm 3 giai đoạn. Giai đoạn 1 thực hiện với các mẫu 8 x 8, giai đoạn 2 với các mẫu 16 x 16 và gai đoạn 3 với các mẫu 32 x 32. Ban đầu, các vector biểu diễn các mẫu 8 x 8 được đưa vào mạng cho huấn luyện. Sau khi được huấn luyện qua vài lần (epoch), mạng được nâng cấp bằng cách điều chỉnh các trọng số giữa lớp đầu tiên và lớp thứ hai. Mạng sau khi được nâng cấp được huấn

luyện tiếp qua vài epoch khác với các mẫu 16 x 16 (Giai đoạn 2). Sau đó mạng lại được nâng cấp cho phiên huấn luyện tiếp theo. Tương tự, mạng kết quả được huấn luyện qua vài epoch khác cho tới khi đạt tới sự hội tụ thỏa mãn (Giai đoạn 3). Mô hình mạng neuron multiscale được thể hiện qua hình 2.13 sau:



Hình 2.13: Mô hình khái niệm kĩ thuật huấn luyện multscale.



Hình 2.14: Mạng neuron ban đầu (trên) và mạng neuron được nâng cấp (dưới).

Hình 2.14 mô tả quá trình điều chỉnh trọng số sau mỗi lần nâng cấp mạng. P_1, P_2, P_3, và P_4 là các giá trị mật độ pixel và P_{ave} là giá trị mật độ pixel trung bình của những pixel này. Sau quá trình nâng cấp, giá trị trọng số ban đầu W bị tách thành 4, mỗi cái được kết nối tới một vị trí pixel.

CHƯƠNG 3: NHẬN DẠNG PHIẾU KIỂM KÊ SẢN PHẨM

3.1 Tiền xử lý

Khi đã có ảnh quét của phiếu kiểm kê sản phẩm, chúng ta chưa thể tiến hành nhận dạng ngay mà phải qua bước phân đoạn để tìm ra vùng chứa các kí tự trong ảnh rồi trích chọn đặc trưng của kí tự. Muốn phân đoạn được chính xác, thì ảnh nhất thiết phải được tiền xử lý để cải thiện chất lượng. Bởi vì ảnh quét thường có nhiễu, chúng ta dùng lọc low-pass để khử nhiễu. Sau khi lọc có thể kết hợp với lọc high-pass để làm tăng độ sắc nét của ảnh.

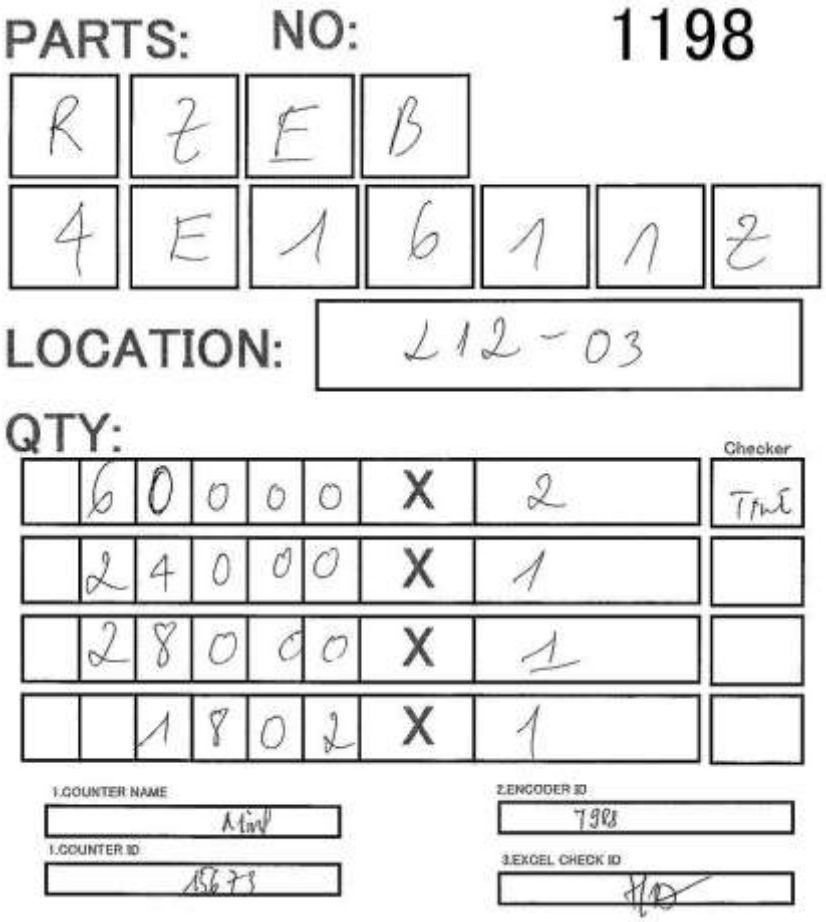
Tiếp theo, chúng ta cần biến đổi ảnh về ảnh nhị phân bằng kĩ thuật phân ngưỡng. Đây là bước đơn giản nhưng là điều kiện tiên quyết cho thực hiện phân đoạn. Như đã đề cập trong Chương 2 thì có hai cách để phân ngưỡng là thủ công và tự động. Trong đó phương pháp thủ công là đơn giản và đủ hiệu quả đối với bài toán đặt ra nên được chọn để thực thi. Ngưỡng cố định được chọn trước là 192.

Sau khi ảnh đã là ảnh nhị phân, chúng ta có thể tiến hành loại bỏ nghiêng cho ảnh. Đây cũng là một bước tiền xử lý không kém phần quan trọng khi mà các thuật toán ở bước sau không thể thực hiện đối với ảnh quá nghiêng. Vì không có khả năng để loại bỏ nghiêng trong suốt quá trình quét ảnh nên chúng ta cần phải dò và chỉnh nghiêng cho ảnh. Thuật toán dò và chỉnh nghiêng được áp dụng như đã trình bày trong Chương 2.

3.2 Phân đoạn

Đối với ảnh phiếu kiểm kê sản phẩm, phân đoạn sẽ là việc đi tìm các hàng, từ các hàng sẽ đi tìm các cột, từ các cột sẽ tìm ra các kí tự. Có nhiều phương pháp phân đoạn để tìm ra các đối tượng trong ảnh, trong đó phương pháp dựa trên histogram tỏ ra là một phương pháp đơn giản, hiệu quả và rất phù hợp với bài toán đặt ra.

3.2.1 Tìm các hàng



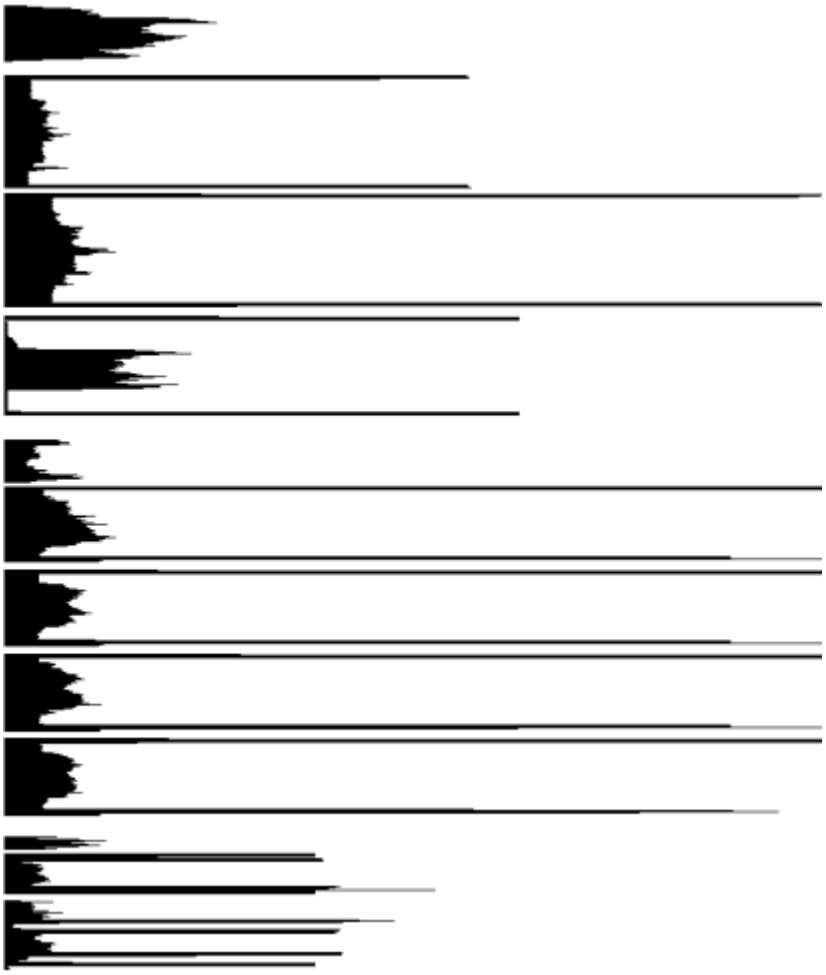
Hình 3.1: Phiếu kiểm kê sản phẩm.

Từ phiếu kiểm kê hình 3.1 chúng ta thấy có 4 vùng dữ liệu cần xét tới đó là NO, PARTS, LOCATION, QTY. Nếu coi vùng NO là hàng đầu tiên, vùng PARTS gồm hàng thứ 2 và 3, vùng LOCATION là hàng thứ 4, vùng QTY gồm hàng thứ 5, 6, 7, 8 thì việc tìm hàng sẽ là tìm vị trí điểm đầu và điểm cuối của mỗi trong số 8 hàng này. Ngoại trừ hàng đầu tiên, 7 hàng còn lại có một điểm chung là vị trí điểm đầu và vị trí điểm cuối đều nằm trên các đường kẻ. Vị trí các hàng này sẽ dễ dàng được tính dựa vào histogram chiếu theo chiều ngang của ảnh:

- Tìm giá trị max trong histogram ngang.
- Tính ngưỡng $\mu = max - (max / 2)$.
- Vị trí các hàng là các vị trí i thỏa mãn điều kiện $HisValue[i] > \mu$.

Đối với hàng đầu tiên thì việc tính các vị trí đơn giản hơn và cũng dựa vào histogram ngang. Chúng ta bắt đầu duyệt từ vị trí 0 cho đến vị trí i mà HisValue[i]

> 0 thì i là vị trí điểm đầu hàng. Duyệt tiếp cho đến vị trí j mà HisValue[j] = 0 thì j – 1 là vị trí điểm cuối hàng.



Hình 3.2: Histogram chiếu theo chiều ngang của hình 3.1.

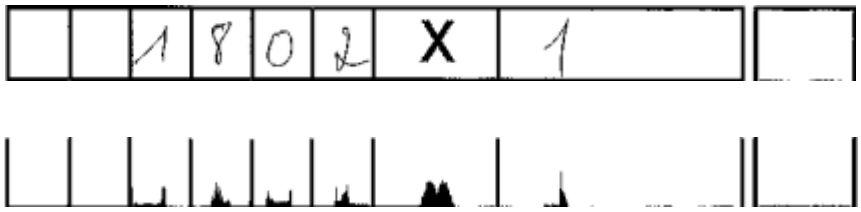
3.2.2 Tìm các cột

Sau khi đã tìm thấy các hàng thì công việc tiếp theo là tìm vị trí các cột. Lại một lần nữa có hai trường hợp xảy ra, một cho hàng 1 và một cho 7 hàng còn lại. Đối với hàng 1, xác định vị trí các cột là tìm chính xác left và right của kí tự cần nhận dạng. Dựa vào histogram dọc được chiếu từ vị trí điểm đầu hàng 1 đến vị trí điểm cuối hàng 1, bắt đầu duyệt từ vị trí width – 1 cho đến width / 2:

- Nếu HisValue[i] > 0 thì i là vị trí điểm cuối cột (right).
- Nếu HisValue[j] = 0 thì j – 1 là vị trí điểm đầu cột (left).

Đối với các hàng còn lại, ta chiếu histogram theo chiều dọc từ vị trí điểm đầu hàng đến vị trí điểm cuối hàng, bắt đầu duyệt từ vị trí 0 cho đến width – 1:

- Tìm giá trị max trong histogram dọc.
- Tính ngưỡng μ = max – 5.
- Vị trí các cột là các vị trí i thỏa mãn điều kiện HisValue[i] > μ.



Hình 3.3: Một hàng QTY (trên) và histogram chiếu theo chiều dọc của hàng (dưới).

3.2.3 Loại bỏ các hàng, cột thừa

Sau khi tìm ra các hàng chúng ta cần loại bỏ đi các hàng không cần thiết. Chúng ta cần loại bỏ hàng vì đôi khi số hàng được tìm thấy lớn hơn 8. Điều này đươc thấy rõ qua lược đồ histogram hình 3.2, chúng ta thấy 4 đường kẻ cuối cùng tương ứng với 2 hàng có thể được tìm thấy khi dùng thuật toán tìm hàng ở trên. Vì vậy sau khi tìm hàng cần so sánh số hàng tìm được với 8, nếu lớn hơn thì loại bỏ đi.

Công việc iếp theo là loại bỏ các cột thừa. Từ hình 3.1 chúng ta thấy công việc này chỉ cần áp dụng đối với vùng QTY. Đối với mỗi hàng thuộc QTY ta sẽ tìm được chính xác 10 cột, trong đó cột thứ 10, 9 và 7 là các cột cần loại bỏ. Tiếp theo là loại bỏ các cột trống. Công việc được thực hiện đơn giản dựa vào histogram chiếu theo chiều dọc của hàng, chúng ta duyệt từ điểm đầu cột cho tới điểm cuối cột, tại mỗi vị trí i:

- Nếu HisValue[i] = 0 thì xét tiếp vị trí i + 1.
- Nếu HisValue[i] > 0 thì thôi không xét tiếp.
- Nếu xét đến được vị trí điểm cuối cột thì kết luận cột cần loại bỏ.

Sau khi đã loại bỏ các cột không cần thiết và các cột trống thì xét xem hàng còn bao nhiêu cột. Nếu hàng không còn cột nào thì loại bỏ hàng.

3.3 Trích chọn đặc trưng

Sau khi đã có các hàng và cột cần thiết, chúng ta chưa thể trích chọn đặc trưng của kí tự được ngay, bởi vì vùng ảnh trong các ô cột chưa hoàn toàn chứa kí tự. Vì vậy công việc tiếp theo là tìm ra chính xác top, bottom, left, right của kí tự để cắt ra được vùng ảnh hình chữ nhật chứa hoàn toàn kí tự.

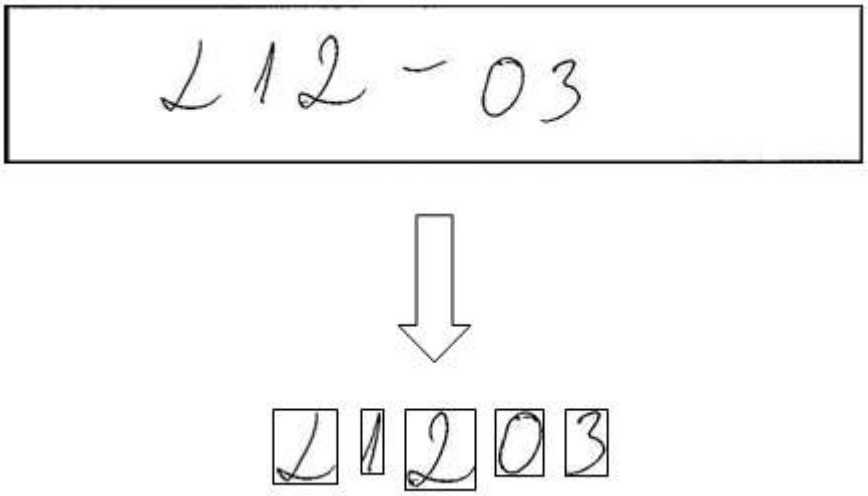
Có hai trường hợp xảy ra khi xác định kí tự: trường hợp ô chứa chỉ một kí tự và trường hợp ô chứa nhiều kí tự (hàng LOCATION). Với trường hợp thứ nhất, tư tưởng là dùng một khung di động có kích cỡ nhỏ hơn ô, với trung tâm được đặt vào trung tâm của ô. Việc xác định top, bottom, left, right của khung được ước lượng sao cho có thể loại bỏ các điểm đen ở các cạnh của ô. Khi đã xác định các vị trí của khung, chúng ta đầu tiên xác định left của kí tự, bắt đầu từ vị trí i tại left của khung như sau:

- Duyệt theo chiều dọc từ top của khung đến bottom của khung.
- Nếu thấy xuất hiện điểm đen, giảm i rồi quay lại bước 1 duyệt tiếp cho tới khi gặp điểm trắng thì dừng lại, left của kí tự là vị trí i – 1.
- Nếu thấy không xuất hiện điểm đen, tăng i rồi quay lại bước 1 duyệt tiếp cho tới khi gặp điểm đen thì dừng lại, left của kí tự là vị trí i.

Tương tự chúng ta áp dụng thuật toán trên để tìm ra right, top, và bottom của kí tự. Chỉ chú ý là khi áp dụng cho top và bottom thì chúng ta duyệt theo chiều ngang từ vị trí left tới vị trí right của kí tự chứ không phải của khung.

Với trường hợp thứ hai, thủ tục có hơi khác một chút nhưng thuật toán chính vẫn là thuật toán trên. Đầu tiên chúng ta phải tách ra được vùng chứa từng kí tự từ ô. Điều này đạt được bằng cách dựa vào histogram chiếu theo chiều dọc của hàng như đã dược đề cập trong phần tìm cột. Sau đó áp dụng thuật toán trên cho từng vùng chứa kí tự đã được tách.

Hình 3.4: Xác định kí tự trường hợp ô chứa 1 kí tự.



Hình 3.5: Xác định kí tự trường hợp ô chứa nhiều kí tự.

Khi đã có được ảnh chứa hoàn toàn kí tự, chúng ta bắt đầu trích chọn đặc trưng của nó như được giải thích trong Chương 2.

3.4 Nhận dạng

Trái ngược với quá trình huấn luyện, quá trình nhận dạng diễn ra đơn giản. Ảnh kí tự sau khi trích chọn đặc trưng là một vector được đưa trực tiếp vào mạng đã qua huấn luyện. Ở đây, vector được lan truyền xuôi dùng các phần tử trọng số đã qua huấn luyện để đưa ra 35 giá trị phần tử tại lớp ra. Giá trị nào tại đầu ra là lớn nhất thì có thể kết luận kí tự thuộc về phân lớp đó.

Đối với nhận dạng phiếu kiểm kê sản phẩm, chúng ta có thể tiến hành nhận dạng kí tự chữ số và kí tự chữ cái riêng để đạt kết quả chính xác hơn. Ví dụ như hàng 2 của phiếu chỉ chứa kí tự chữ cái viết tay thì chúng ta chỉ xét đến 25 phần tử cuối tại lớp ra. Tương tự vùng QTY của phiếu chỉ chứa kí tự chữ số viết tay nên chúng ta chỉ cần xét 10 phần tử đầu tiên tại lớp ra. Ngoài ra, chúng ta thấy vùng NO của phiếu bao gồm các kí tự chữ số in nên có thể tiến hành nhận dạng dựa trên các trọng số chỉ được huấn luyện với 10 kí tự chữ số in.

CHƯƠNG 4: THỰC NGHIỆM

4.1 Thiết kế và cài đặt hệ thống

Hệ thống gồm có 3 phần, mỗi phần là một project được thực thi bằng ngôn ngữ C#, Framework .NET 3.5.

Project AutoAuditInvoiceUpdate thực hiện công việc nhận dạng vùng chứa dữ liệu của phiếu kiểm kê sản phẩm và cập nhật vào cơ sở dữ liệu. Đầu vào là tập các ảnh phiếu quét chứa trong một thư mục và một file text lưu các trọng số của mạng đã qua huấn luyện.

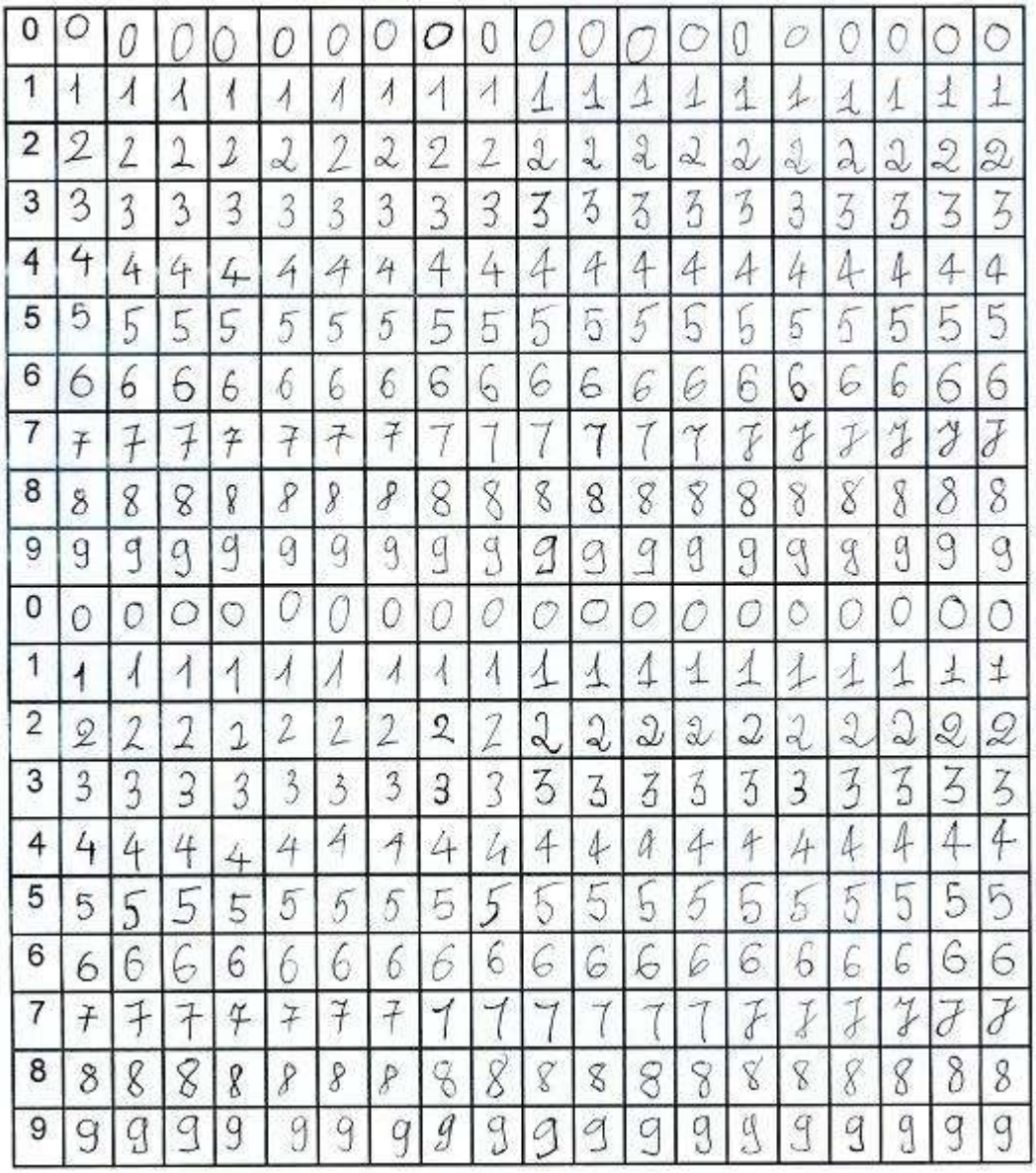
Project HandwrittenCharacterRecognition xây dựng một giao diện có chức năng huấn luyện tập mẫu, cập nhật tập mẫu huấn luyện, tập mẫu kiểm thử và nhận dạng kiểm thử.

Project NeuronLib là một thư viện bao gồm các lớp mô phỏng các mạng neuron nhân tạo và đặc trưng khác nhau. Ngoài ra thư viện còn có chức năng tiền xử lý ảnh.

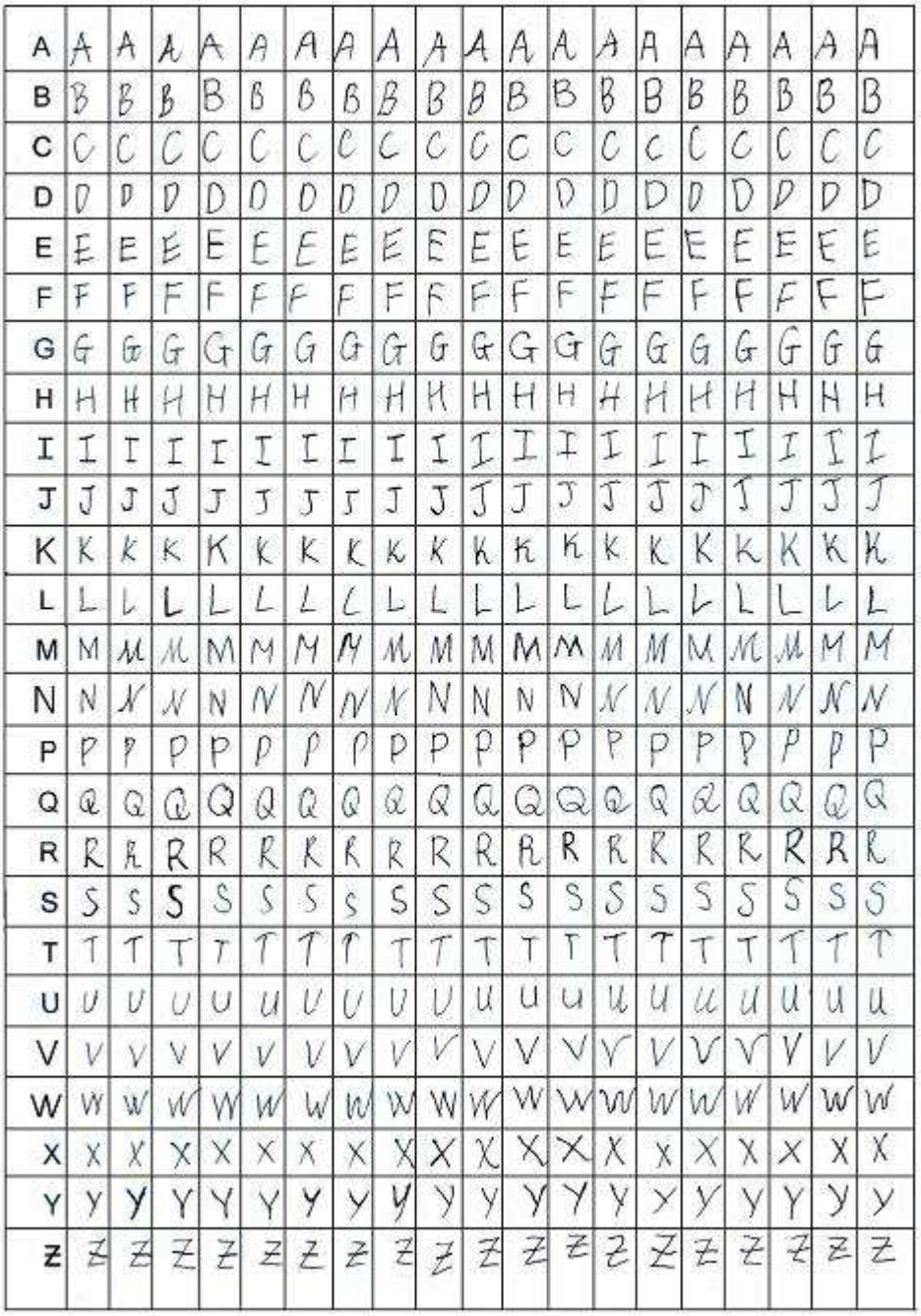
4.2 Xây dựng tập mẫu huấn luyện

Như đã nói từ trước, chúng ta cần phải có một tập mẫu bao gồm các ảnh là biến thể của 35 kí tự cần nhận dạng để huấn luyện mạng. Những ảnh này chỉ bao gồm hai mức xám là đen (foreground) và trắng (background), và được chuẩn hóa về một kích cỡ cố định là 32 x 32. Chất lượng cũng như số lượng của tập mẫu ảnh hưởng quyết định tới khả năng nhận dạng của mạng sau khi học. Để thu thập được một tập mẫu đủ chất lượng và đa dạng là một công việc đòi hỏi nhiều nỗ lực. Vì thời gian cũng như khả năng có hạn nên em chỉ kịp thu thập được 2690 kí tự. Trong đó 10 kí tự đầu tiên là chữ số in, 790 kí tự tiếp theo là chữ số viết tay, 1900 kí tự cuối là chữ cái viết tay.

Để tiện cho việc thu thập mẫu chúng ta cần xây dựng một chương trình thực hiện việc cập nhật tập mẫu huấn luyện cũng như tập mẫu kiểm thử. Đầu vào của chương trình là một ảnh chứa một bảng các hàng và các cột. Ứng với mỗi hàng và mỗi cột sẽ là một ô chứa kí tự mẫu. Một bảng sẽ bao gồm chỉ các kí tự số hoặc chỉ các kí tự chữ cái. Các kí tự sẽ được sắp xếp lần lượt theo chiều dọc hay theo chiều ngang như 0, 1, …, 8, 9 hay A, B, …, Y, Z để thuận lợi cho việc lấy nhãn của kí tự khi cắt ảnh.

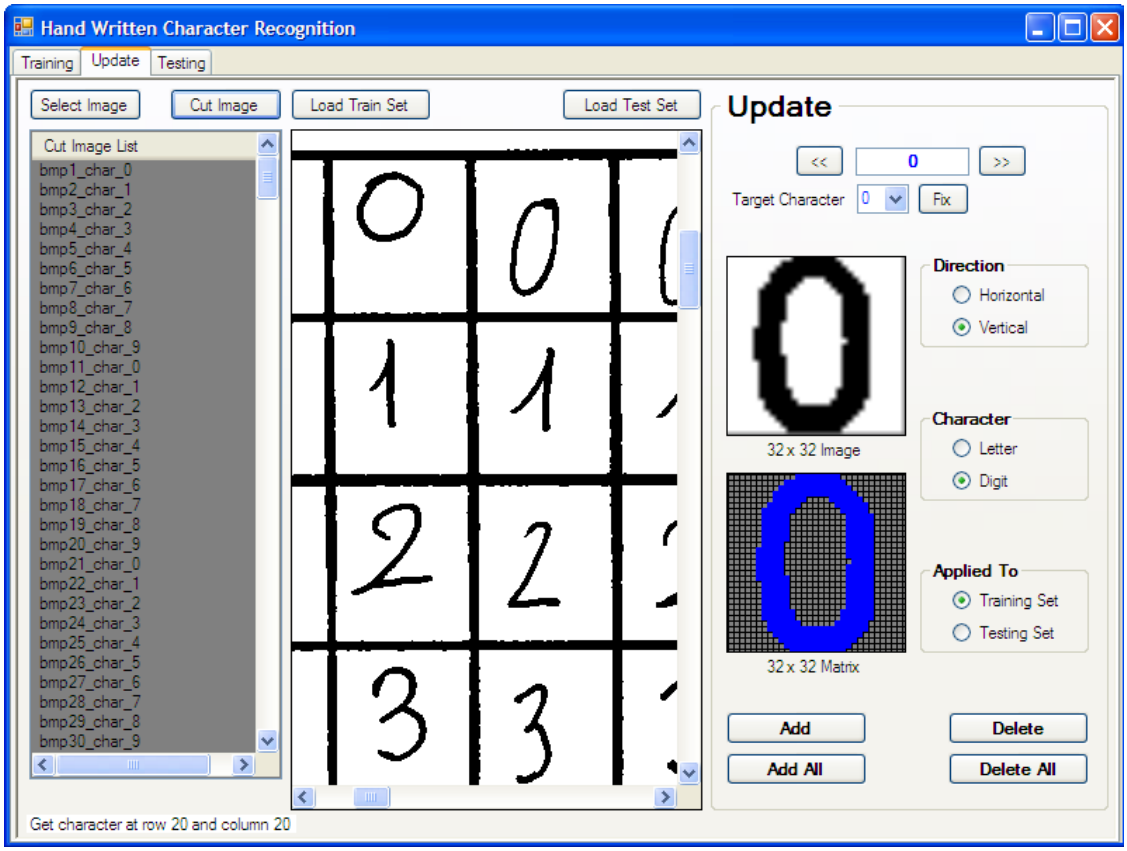


Hình 4.1: Ảnh chứa các mẫu kí tự số.



Hình 4.2: Ảnh chứa các mẫu kí tự chữ cái.

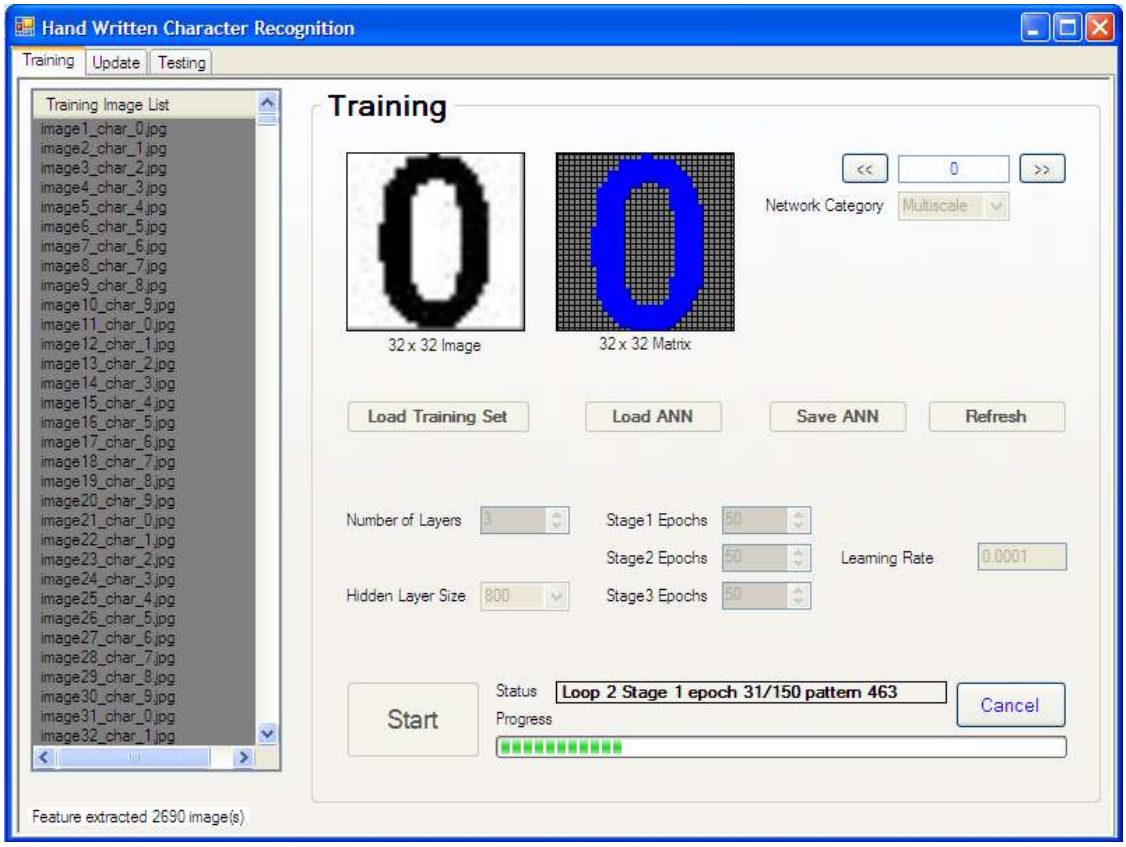
Khi đầu vào đã được chuẩn hóa, công việc còn lại dành cho chương trình chỉ là việc tách hàng, tách cột, tìm ô dựa và histogram tương tự như những phần trước đã trình bày. Cuối cùng, khi ghi file ảnh kí tự lên đĩa, chúng ta gắn nhãn của kí tự vào cuối tên file để thuận lợi cho hình thành tập đích sau này.



Hình 4.3: Giao diện cập nhật mẫu kí tự.

4.3 Huấn luyện mạng

Huấn luyện mạng là một quá trình yêu cầu thử nghiệm nhiều lần. Mỗi lần thử chúng ta thay đổi các tham số cần thiết cho tới khi mạng đạt tới khả năng nhận dạng cao nhất. Bởi vậy, xây dựng được một chương trình cho phép chúng ta sửa đổi các tham số như epoch (số lần huấn luyện), hệ số học, số neuron của lớp ẩn là một yêu cầu cần thiết. Ngoài ra, cũng cần có chức năng ghi và nạp dữ liệu mạng cho sử dụng lần sau.

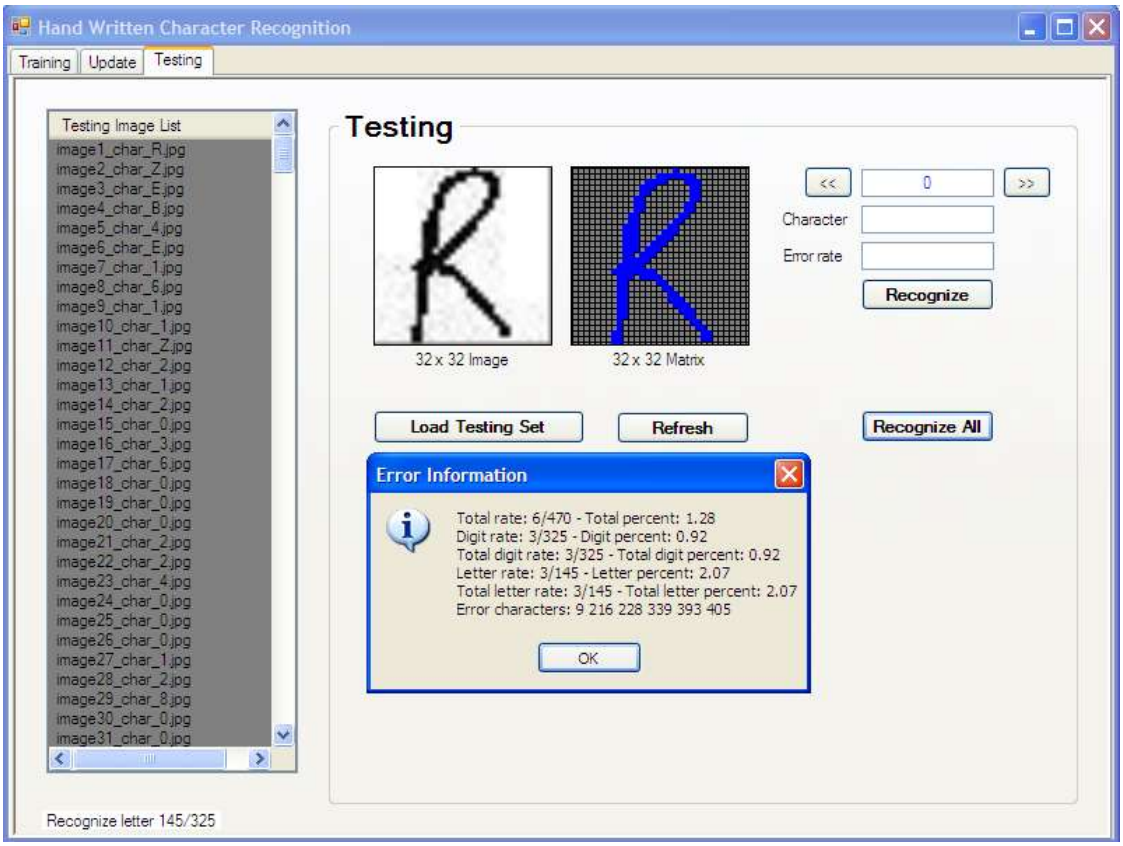


Hình 4.4: Giao diện huấn luyện mạng.

Trước khi huấn luyện mạng chúng ta phải có 2 thành phần: danh sách các vector đầu vào và danh sách các vector đầu ra mong muốn của kí tự. Các vector đầu vào có 2 giá trị: -1 cho màu nền (trắng), 1 cho màu kí tự (đen). Các vector đầu ra cũng có 2 giá trị -1 và 1: giá trị 1 ứng với vị trí của kí tự, giá trị -1 ứng với tất cả các vị trí còn lại. Có tất cả 35 vị trí, sắp xếp lần lượt là: 0, 1, …, 8, 9, A, B, …, X, Y, Z.

Mạng được huấn luyện theo kĩ thuật multiscale với hàm kích hoạt và trọng số khởi tạo được chọn như đã giải thích trong Chương 2. Về lý thuyết mạng có thể có số lớp ẩn tùy ý nhưng một lớp ẩn thường là đủ hiệu quả. Số neuron lớp ẩn được chọn sao cho phù hợp với kích cỡ của tập huấn luyện. Đối với tập huấn luyện 2690 mẫu thì 100 neuron có lẽ là phù hợp. Cuối cùng, hệ số học phải được chọn sao cho đủ nhỏ, có thể trong khoảng [0.0001, 0.01].

4.4 Nhận dạng kiểm thử



Hình 4.5: Giao diện cho nhận dạng kiểm thử.

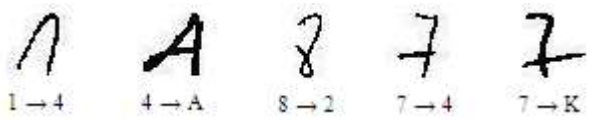
Tập mẫu kiểm thử sẽ bao gồm các kí tự được cắt ra trực tiếp từ các phiếu kiểm kê sản phẩm. Sau khi thu thập và loại bỏ đi các phiếu không đạt tiêu chuẩn (kí tự bị viết chồng lên nhau, tẩy xóa…) thì còn lại 20 phiếu với 470 kí tự viết tay được cắt ra. Bằng phương pháp huấn luyện multiscale trên một mạng 3 lớp với 100 neuron tại lớp ẩn, một số kết quả được rút ra qua bảng sau:

Bảng 4.1: Kết quả kiểm thử với các thông số khác nhau.

Giai đoạn 1 Epoch	Giai đoạn 2 Epoch	Giai đoạn 3 Epoch	Hệ số học	Tỉ lệ lỗi
25	25	25	0.0001	41.7%
25	25	25	0.0005	43.4%
25	25	25	0.001	41.06%
25	25	25	0.005	34.26%
25	25	25	0.01	50.58%
25	25	50	0.0001	36.17%
25	25	50	0.0005	37.45%
25	25	50	0.001	36.17%
25	25	50	0.005	28.51%
25	25	50	0.01	34.47%
50	50	25	0.0001	40.64%
50	50	25	0.0005	34.68%
50	50	25	0.001	33.4%
50	50	25	0.005	37.23%
50	50	25	0.01	55.11%
50	50	50	0.0001	35.96%
50	50	50	0.0005	30.64%
50	50	50	0.001	30%
50	50	50	0.005	29.15%
50	50	50	0.01	32.34%

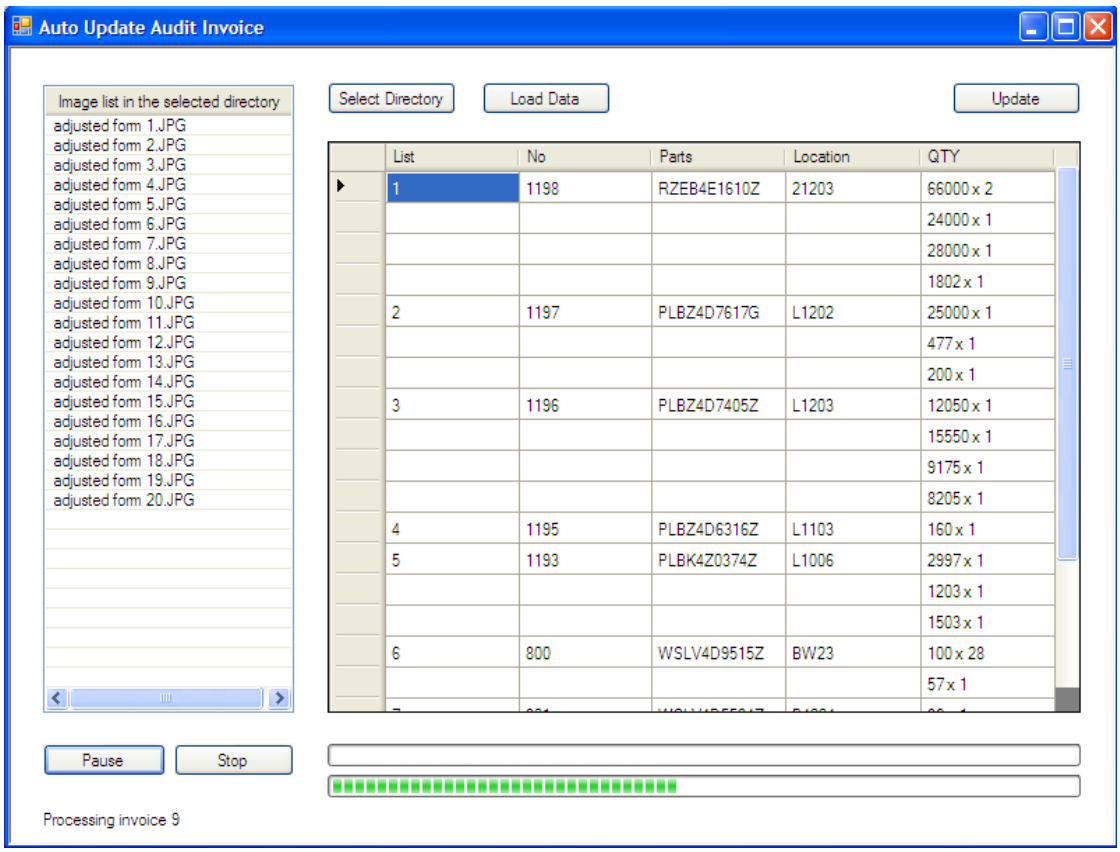
Từ bảng kết quả trên chúng ta thấy mạng đạt tỉ lệ lỗi thấp nhất khi hệ số học là 0.005 với 25-25-50 epoch. Mặc dù đây có thể vẫn chưa là tỉ lệ lỗi thấp nhất nhưng có thể lấy nó làm thông số cho các lần thử sau với một kiến trúc mạng khác. Kết quả, khi tăng số neuron lên thành 200 thì tỉ lệ lỗi giảm còn 26.81%. Tăng số lớp ẩn thành 2 cũng giảm tỉ lệ lỗi xuống còn 25.96% và 23.62%, lần lượt với 100 và

200 neuron tại các lớp ẩn. Tất cả các kết quả trên đều cho thấy khả năng nhận dạng chưa cao của mạng. Lý do là số lượng tập mẫu cho huấn luyện còn ít, chưa đủ tính đa dạng, hơn nữa có nhiều kí tự kiểm thử được viết theo dạng chưa từng được huấn luyện. Để làm phép so sánh, mạng tiếp tục được huấn luyện với 2690 mẫu từ trước cộng với 470 mẫu kiểm thử thì tỉ lệ lỗi chỉ còn 1.06% - một tỉ lệ được chấp nhận. Điều đó chứng tỏ khả năng học của mạng multiscale là rất tôt. Chúng ta có thể cải thiện khả năng nhận dạng của mạng bằng cách tăng số mẫu huấn luyện, đặc biệt khi số mẫu được lấy từ chính những người viết phiếu.



Hình 4.6: Các kí tự bị nhận dạng nhầm.

4.5 Cập nhật phiếu kiểm kê sản phẩm tự động



Hình 4.7: Giao diện cập nhật phiếu kiểm kê sản phẩm.

Khi mạng đã qua huấn luyện thì công việc tiếp theo là xây dựng chương trình cập nhật phiếu tự động sẽ diễn ra đơn giản. Đầu vào của chương trình sẽ là một thư mục chứa các phiếu cần cập nhật và một file text chứa các trọng số đã qua huấn luyện. Khi các đầu vào là hợp lệ thì quá trình xử lý có thể diễn ra. Vì quá trình nhận dạng sẽ có lỗi, chúng ta nên xử lý từng phiếu một, rồi kiểm tra chỉnh sửa kết quả, trước khi xử lý phiếu tiếp theo. Sau khi đã xử lý toàn bộ các phiếu và xác minh kết quả là chính xác thì tiến hành cập nhật vào cơ sở dữ liệu. Chương trình sẽ kết nối với SQL Server của máy bằng công nghệ ADO.NET, sử dụng câu lệnh truy vấn INSERT để cập nhật dữ liệu vào cơ sở dữ liệu.

CHƯƠNG 5: KẾT LUẬN

5.1 Kết quả nghiên cứu

Như đã trình bày trong 4 Chương vừa qua, đề tài đã tập trung nghiên cứu về những vấn đề chính sau:

- Mô hình mạng neuron nhân tạo.
- Ứng dụng mạng vào bài toán nhận dạng kí tự viết tay.
- Phương pháp phân đoạn dựa trên histogram để tìm ra các kí tự trong ảnh.
- Ngoài ra, đề tài cũng nêu lên một số kĩ thuật xử lý ảnh cho việc tiền xử lý và một số kĩ thuật cải thiện quá trình huấn luyện cũng như nhận dạng của mạng neuron nhân tạo.

5.2 Hướng nghiên cứu tiếp theo

Mặc dù đã xây dựng được một hệ thống có khả năng cập nhật phiếu kiểm kê sản phẩm tự động nhưng vẫn tồn tại những hạn chế, một phần bởi vì thời gian nghiên cứu còn ngắn. Những hạn chế đó nằm ở giai đoạn tiền xử lý, phân đoạn còn sơ sài, khả năng nhận dạng còn kém. Vì thế trong thời gian sắp tới, ưu tiên nghiên cứu của em sẽ là:

- Tìm hiểu thêm một số kĩ thuật xử lý ảnh.
- Tìm hiểu một số mô hình mạng neuron cũng như một số phương pháp trích chọn đặc trưng hiệu quả để có thể cho ra kết quả nhận dạng cao.
- Tìm hiểu một số phương pháp biến đổi ảnh để có thể mở rộng tập mẫu, đồng thời cũng tiến hành thu thập thêm tập mẫu chất lượng.

TÀI LIỆU THAM KHẢO

[1.] Phùng Công Định, *Ứng dụng mạng Neuron trong nhận dạng kí tự viết tay*, Đồ án tốt nghiệp đại học, trường đại học Dân Lập Hải Phòng, tr 30-34, 2008.

[2.] Y. LeCun, L. Bottou, G. Orr, and K. Muller, *"Efficient BackProp"* in *Neuron Networks: Tricks of the trade*, (G. Orr and Muller K., eds.), pages 5-13, 1998.

[3.] Velappa Ganapathy and Kok Leong Liew, *"Handwritten Character Recognition Using Multiscale Neuron Network Training Technique"* in *World Academy of Science, Engineering and Technology 39*, pages 32-36, 2008.

[4.] Dwayne Phillips, *Image Procesing in C Second Edition*, Electronic Edition 1.0, 26 April 2000.

[5.] Mike O'Neill, *Neuron Network for Recognition of Handwritten Digits*, www.codeproject.com, 26 November 2006.

[6.] Mackenb, *How to deskew an image*, www.codeproject.com, 25 April 2006.