TẠP CHÍ KHOA HỌC
Khoa học Tự nhiên và Công nghệ, Số 14 (4/2019) tr.107-118

NHẬN DẠNG CHỮ SỐ VIẾT TAY DỰA TRÊN MẠNG NƠ-RON TÍCH CHẬP SÂU

Nguyễn Văn Tú, Hoàng Thị Lam, Nguyễn Thị Thanh Hà
Trường Đại học Tây Bắc

Tóm tắt: *Trong lĩnh vực xử lý ảnh, nhận dạng mẫu là một trong các thách thức lớn nhất của các nhà nghiên cứu trong những năm qua. Mục tiêu của nhận dạng mẫu là phát hiện, trích chọn các đặc trưng trong ảnh để phân loại các mẫu vào các lớp khác nhau. Một bài toán nổi tiếng trong lĩnh vực này là nhận dạng chữ số viết tay, trong đó mỗi chữ số phải được gán vào một trong 10 lớp sử dụng một số phương pháp phân loại. Mục đích của chúng tôi trong bài báo này là trình bày một phương pháp học sâu để so sánh với các phương pháp dựa trên các kỹ thuật thống kê đã có để giải quyết bài toán nhận dạng chữ số viết tay. Chúng tôi sẽ xây dựng mô hình mạng nơ-ron tích chập sâu với việc sử dụng nhiều lớp khác nhau của mạng để có thể trích chọn tự động được các đặc trưng tốt nhất trong ảnh. Đồng thời, chúng tôi cũng kết hợp giữa mạng nơ-ron tích chập và Multi-layer Perceptron nhằm cải thiện hiệu suất của mô hình. Chúng tôi đã xây dựngcác thực nghiệm sử dụng tập dữ liệu MNIST và đã đạt được độ chính xác phân loại cao nhất là 99,34% và tỷ lệ lỗi là 0,74%. Các kết quả này cho thấy mô hình đề xuất của chúng tôi cho kết quả cao hơn so với nhiều mô hình đã xây dựng trước đó trên cùng tập dữ liệu.*

Từ khóa: *Nhận dạng chữ số viết tay, mạng nơ-ron tích chập, multi-layer perceptron, phân loại.*

1. Tổng quan

Trong những năm gần đây, chúng ta đã được chứng kiến nhiều thành tựu vượt bậc trong lĩnh vực xử lý ảnh (image processing). Các hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một trong những mô hình học sâu (Deep Learning) tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

Các bài toán nhận dạng đang được ứng dụng trong thực tế hiện nay tập trung vào nhận dạng mẫu, nhận dạng tiếng nói và nhận dạng chữ viết… Nhận dạng chữ viết tay là bài toán được quan tâm rất nhiều vì nó là một trong các yêu cầu trong nhiều ứng dụng thực tế. Các ứng dụng của nhận dạng chữ viết tay đã và đang được ứng dụng vào đời sống như phục vụ cho công việc tự động hóa đọc tài liệu, tăng tốc độ và hiệu quả nhập thông tin vào máy tính. Nhận dạng chữ viết tay có thể phục vụ cho các ứng dụng đọc và xử lý các chứng từ, hóa đơn, phiếu ghi, bản viết tay chương trình...

Hiện nay, đã có một số đề tài nghiên cứu nhận dạng chữ viết tay sử dụng các mô hình như: K láng giềng gần nhất (K-Nearest Neighbor - KNN), máy hỗ trợ véc-tơ (Support Vector Machine - SVM), mô hình Markov ẩn (Hidden Markov Model - HMM)… Tuy nhiên, các mô

Ngày nhận bài: 07/9/2018. Ngày nhận đăng: 09/11/2018.
Liên lạc: Nguyễn Văn Tú, e-mail: tuspttb@utb.edu.vn

hình này cho kết quả nhận dạng không cao, mất nhiều thời gian cho việc trích rút các đặc trưng trong ảnh. Chính vì vậy, trong nghiên cứu này chúng tôi sẽ xây dựng một mô hình mới để có thể trích rút tự động các đặc trưng trong ảnh và mô hình mới này cũng phải cho kết quả tốt hơn các mô hình đã xây dựng trước đó.

Từ những thành công của mạng nơ-ron trong lĩnh vực xử lý ảnh, chúng tôi sẽxây dựng mô hình học máy tiên tiến mạng nơ-ron tích chập sâu (Deep Convolutional Neural Network - DCNN) kết hợp với Multi-layer Perceptron (MLP) vào giải quyết bài toán nhận dạng chữ số viết tay.

Nhận dạng chữ viết tay được thực hiện qua hai hình thức đó là nhận dạng online và nhận dạng offline. Nhận dạng online có nghĩa là máy tính sẽ nhận dạng các chữ được viết lên màn hình ngay khi nó được viết. Đối với những hệ nhận dạng này, máy tính sẽ lưu lại các thông tin về nét chữ như thứ tự nét viết, hướng và tốc độ của nét viết trong khi nó đang được viết. Còn nhận dạng offline tức là việc nhận dạng được thực hiện sau khi chữ đã được viết hay in lên giấy rồi, lúc đó thông tin đầu vào là hình ảnh văn bản hoặc ký tự cần nhận dạng.

Trong khuôn khổ nội dung bài báo này, chúng tôi chỉ xét hình thức nhận dạng offline cho các chữ số viết tay.

2. Các nghiên cứu liên quan

Bài toán nhận dạng chữ viết tay được ứng dụng rất nhiều trong thực tế, được tích hợp vào hệ thống nhận dạng form tự động, tích hợp trong các máy PDA có màn hình cảm ứng, nhận dạng chữ ký... Do có nhiều ứng dụng quan trọng như vậy nên từ lâu bài toán nhận dạng chữ viết tay đã thu hút sự quan tâm của nhiều nhà nghiên cứu. Nghiên cứu của Norhidayu và các tác giả [5] sử dụng các mô hình phân loại SVM, KNN và mạng nơ-ron. Kết quả thực nghiệm cho thấy mô hình sử dụng thuật toán phân loại KNN cho kết quả phân loại cao nhất là 99,26%. Nghiên cứu của Ana và các tác giả [1] đã sử dụng mô hình nhị phân cục bộ (Local Binary Pattern - LBP) như là một bộ trích xuất đặc trưng và bộ phân loại KNN trên hệ thống nhận dạng chữ viết tay của họ trên mẫu C1 được sử dụng bởi ủy ban bầu cử ở Indonesia. Kết quả thực nghiệm cho thấy phương pháp LBP có thể nhận dạng ký tự chữ số viết tay trên bộ dữ liệu MNIST với độ chính xác 89,81% và trên dữ liệu C1 với độ chính xác là 70,91%. Souici-Meslati [8] trình bày một cách tiếp cận lại để nhận dạng số lượng chữ trên ngân phiếu. Các tác giả sử dụng ba bộ phân loại chạy song song: mạng nơ-ron, KNN và Fuzzy K-nearest neighbor. Các kết quả đầu ra được kết hợp từ cả ba bộ phân loại này. Kết quả thực nghiệm trên bộ dữ liệu của họ đạt độ chính xác là 96%. Burrow [7] áp dụng bộ phân loại KNN trên tập dữ liệu của họ và tác giả đạt được độ chính xác là 74%. Jason [3] sử dụng mô hình mạng CNN với việc xây dựng các bộ lọc với các kích thước khác nhau nhằm trích lọc được nhiều thông tin hữu ích trong ảnh. Tác giả đã xây dựng thực nghiệm trên tập dữ liệu MNIST và đạt được kết quả cao với độ chính xác đạt 99,31%.

Từ việc phân tích các nghiên cứu trên, chúng tôi nhận thấy các nghiên cứu này đã sử dụng nhiều mô hình khác nhau cũng như xây dựng các thực nghiệm trên các bộ dữ liệu khác

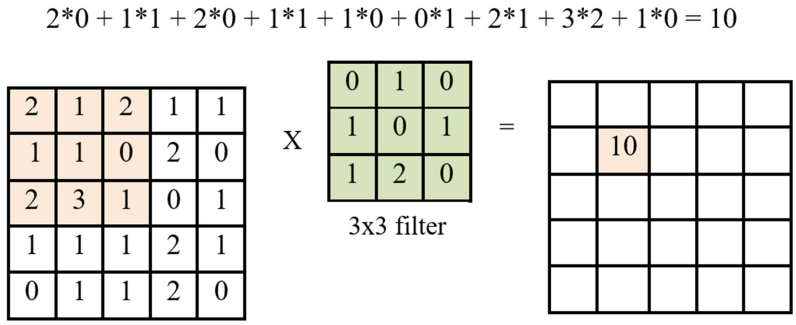
nhau. Tuy mạng nơ-ron đã được áp dụng trong một số nghiên cứu, nhưng cấu trúc mạng của các nghiên cứu này là tương đối đơn giản, chưa khai thác hết các tính năng của các lớp trong mạng. Chính vì vậy, trong nghiên cứu này chúng tôi muốn đề xuất xây dựng một mô hình mới với sự kết hợp của các lớp trong CNN với Multi-layer Perceptron (MLP) nhằm đạt được kết quả tốt hơn cho bài toán nhận dạng chữ số viết tay.

3. Mạng nơ-ron tích chập

Trong phần này, chúng tôi trình bày tóm tắt về CNN và một số lớp thông dụng nhất của mạng này dùng cho lĩnh vực xử lý ảnh.

3.1. Tích chập

Tích chập (convolution) được sử dụng đầu tiên trong xử lý tín hiệu số (signal processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số. Để dễ hình dung, chúng ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận. Hình 1 minh họa cơ chế của tích chập.



Hình 1. Minh họa tích chập

Ma trận bên trái là một ảnh xám, mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel) có giá trị biến thiên từ 0 đến 255. Sliding window còn có tên gọi là kernel, filter hay feature detector. Ở đây, ta dùng một ma trận filter kích thước 3×3 nhân từng thành phần tương ứng với ma trận ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại. Kết quả của tích chập là một ma trận sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái.

3.2. Mô hình mạng nơ-ron tích chập

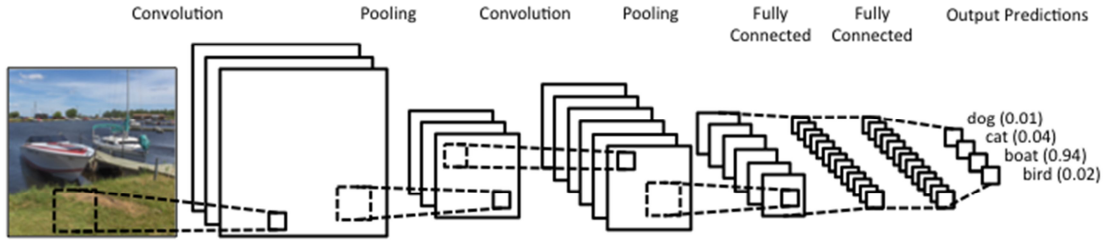
Mô hình CNN chỉ đơn giản gồm một vài layer của convolution kết hợp với các hàm kích hoạt phi tuyến như *ReLU* hay *tanh* để tạo ra thông tin ở mức trừu tượng hơn cho các layer tiếp theo.

Trong mô hình mạng nơ-ron truyền thẳng (feedforward nơ-ron network), các layer kết nối trực tiếp với nhau thông qua một trọng số *w* (weighted vector). Các layer này còn được gọi là có kết nối đầy đủ (fully connected layer) hay affine layer.

Trong mô hình CNN thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được

các kết nối cục bộ. Nghĩa là mỗi nơ-ron ở layer tiếp theo sinh ra từ filter áp đặt lên một vùng ảnh cục bộ của nơ-ron layer trước đó. Mỗi layer như vậy được áp đặt các filter khác nhau, thông thường có vài trăm đến vài nghìn filter như vậy. Một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong suốt quá trình huấn luyện, CNN sẽ tự động học được các thông số cho các filter. Ví dụ, trong nhiệm vụ phân lớp ảnh như được minh họa trong hình 2, CNN sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel > edges > shapes > facial > high-level features. Layer cuối cùng được dùng để phân lớp ảnh.

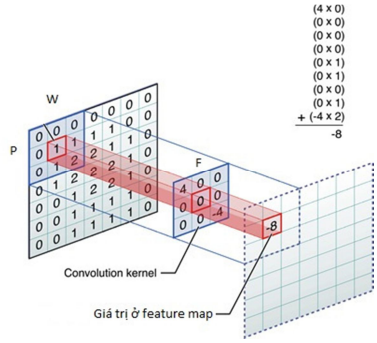


Hình 2. Minh họa kiến trúc CNN dùng trong phân loại ảnh

CNN có tính bất biến và tính kết hợp cục bộ (Location Invariance and Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer sẽ cho tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNN cho ra mô hình với độ chính xác rất cao. Tiếp theo, chúng tôi sẽ trình bày chi tiết các lớp trong mô hình.

Convolutional Layer

Layer này chính là nơi thể hiện tư tưởng ban đầu của CNN. Thay vì kết nối toàn bộ điểm ảnh, layer này sẽ sử dụng một tập các bộ lọc (filters) có kích thước nhỏ so với ảnh (thường là 5×5 hoặc 3×3) áp vào một vùng trong ảnh và tiến hành tính tích chập giữa bộ lọc và giá trị điểm ảnh trong vùng cục bộ đó. Bộ lọc sẽ lần lượt được dịch chuyển theo một giá trị bước trượt (stride) chạy dọc theo ảnh và quét toàn bộ ảnh.



Hình 3. Tính tích chập với các bộ lọc

Như vậy, với một bức ảnh 32×32 và một filter 3×3, ta sẽ có kết quả là một tấm ảnh mới có kích thước 32×32 (với điều kiện đã thêm padding vào ảnh gốc để tính tích chập cho các trường hợp filter quét ra các biên cạnh) là kết quả tích chập của filter và ảnh. Với bao nhiêu filter trong lớp này thì ta sẽ có bấy nhiêu ảnh tương ứng mà lớp này trả ra và được truyền vào lớp tiếp theo. Các trọng số của filter ban đầu sẽ được khởi tạo ngẫu nhiên và sẽ được học dần trong quá trình huấn luyện mô hình. Hình 3 minh họa của một phép tính convolution với bộ lọc có kích thước 3×3.

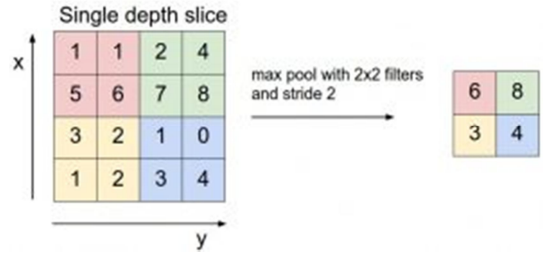
Rectified Linear Unit (*ReLU*) Layer

Layer này thường được cài đặt ngay sau layer Convolution. Layer này sử dụng hàm kích hoạt f(x) = max(0, x). Nói một cách đơn giản, layer này có nhiệm vụ chuyển toàn bộ giá trị âm trong kết quả lấy từ lớp Convolution thành giá trị 0. Ý nghĩa của cách cài đặt này chính là tạo nên tính phi tuyến cho mô hình. Tương tự như trong mạng truyền thẳng, việc xây dựng dựa trên các phép biến đổi tuyến tính sẽ khiến việc xây dựng đa tầng đa lớp trở nên vô nghĩa. Có rất nhiều cách để khiến mô hình trở nên phi tuyến như sử dụng các hàm kích hoạt *sigmoid*, *tanh*,… nhưng hàm f(x) = max(0, x) dễ cài đặt, tính toán nhanh mà vẫn hiệu quả.

Pooling Layer

Layer này sử dụng một cửa sổ trượt quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một bước trượt (stride) cho trước. Khác với layer Convolution, layer Pooling không tính tích chập mà tiến hành lấy mẫu (subsampling). Khi cửa sổ trượt trên ảnh, chỉ có một giá trị được xem là giá trị đại diện cho thông tin ảnh tại vùng đó (giá trị mẫu) được giữ lại. Các phương thức lấy phổ biến trong layer Pooling là *MaxPooling* (lấy giá trị lớn nhất), *MinPooling* (lấy giá trị nhỏ nhất) và *AveragePooling* (lấy giá trị trung bình).

Xét một ảnh có kích thước 32×32 và layer Pooling sử dụng bộ lọc có kích thước 2×2 với bước trượt stride là 2, phương pháp sử dụng là *MaxPooling*. Bộ lọc sẽ lần lượt trượt qua ảnh, với mỗi lần trượt chỉ có giá trị lớn nhất trong 4 giá trị nằm trong vùng cửa sổ 2×2 của bộ lọc được giữ lại và đưa vào ma trận đầu ra. Như vậy, sau khi qua layer Pooling, ảnh sẽ giảm kích thước xuống còn 16×16 (kích thước mỗi chiều giảm 2 lần).



Hình 4. Tính toán với phương pháp MaxPooling

Pooling Layer có vai trò giảm kích thước dữ liệu. Với một bức ảnh kích thước lớn qua nhiều Pooling Layer sẽ được thu nhỏ lại tuy nhiên vẫn giữ được những đặc trưng cần cho việc nhận dạng (thông qua cách lấy mẫu). Việc giảm kích thước dữ liệu sẽ làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting).

Fully Connected (FC) Layer

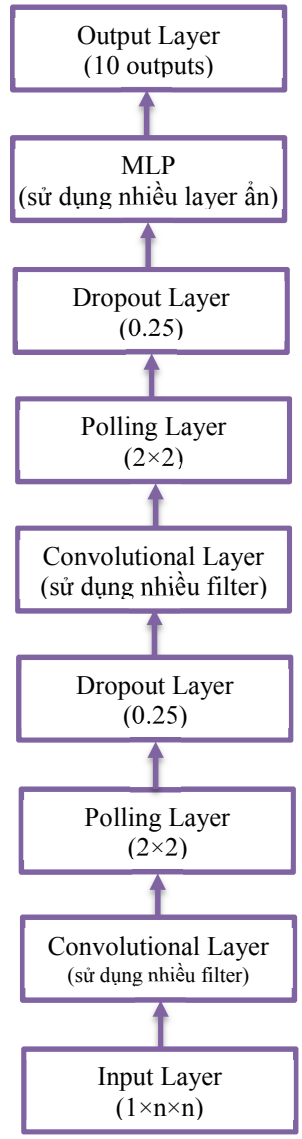
Layer này tương tự với layer trong mạng nơ-ron truyền thẳng, các giá trị ảnh được liên kết đầy đủ vào các nơ-ron trong layer tiếp theo. Sau khi ảnh được xử lý và rút trích đặc trưng từ các layer trước đó, dữ liệu ảnh sẽ không còn quá lớn so với mô hình truyền thẳng nên ta có thể sử dụng mô hình truyền thẳng để tiến hành nhận dạng.

3.3. Hoạt động của mô hình CNN

Mô hình CNN được hình thành bằng cách kết nối các layer nêu trên lại với nhau. Mô hình bắt đầu với Convolutional Layer. ReLU Layer thường được cài đặt ngay sau Convolutional Layerhoặc thậm chí kết hợp cả hai layer này thành một layer. Các layer tiếp theo có thể là Convolutional hay Pooling tùy theo kiến trúc mà ta muốn xây dựng. Cuối cùng sẽ là Fully-Connected Layer để tiến hành phân lớp.

4. Mô hình mạng nơ-ron tích chập sâu cho bài toán

Trong phần này, chúng tôi sẽ xây dựng mô hình để giải quyết bài toán nhận dạng chữ số viết tay. Mô hình của chúng tôi sẽ là sự kết hợp giữa các layer của CNNvới MLP. Trong đó, các layer của CNN được dùng để chắt lọc thông tin trong ảnh nhằm xây dựng véc-tơ đặc trưng dùng để phân loại ảnh. MLP đóng vai trò như một bộ phân loại, nhận đầu vào là véc-tơ đặc trưng xây dựng bởi các layer của CNN và đầu ra là các kết quả phân loại. Hình 5 là kiến trúc chung của mô hình chúng tôi xây dựng.



Hình 5. Mô hình DCNN cho bài toán nhận dạng chữ số viết tay

Tiếp theo, chúng tôi sẽ mô tả các layer của mạng cùng với chức năng của mỗi layernày.

1. Layer đầu tiên của mạng là một Input Layer, layer này chứa các ảnh cần phân loại. Mỗi ảnh là một ma trận xám có kích thước n × n. Ví dụ, kích thước của ảnh là 28×28, khi đó mỗi ảnh có 784 phần tử, mỗi phần tử là một giá trị mức xám.
2. Layertiếp theo của mạng là một Convolutional Layer được gọi là Conv2D. Layer này nhận đầu vào là các ảnh từ lớp Input. Trong Convolutional Layer, chúng tôi sử dụng nhiều filter với kích thước bằng nhau để quét trên ảnh đầu vào (từ Input layer) và tạo ra các ánh xạ đặc trưng cho ảnh. Sau Convolutional Layer, chúng tôi cũng sử dụng hàm kích hoạt *ReLU*.
3. Tiếp theo, chúng tôi định nghĩa một Pooling Layer có giá trị tối đa được gọi là MaxPooling2D. Layer này nhận đầu vào là kết quả của Convolutional Layer ở trên và nó thực hiện chắt lọc lại thông tin, loại bỏ thông tin nhiễu trước khi truyền cho layer tiếp theo của mạng. Trong phần thực nghiệm, chúng tôi sử dụng cửa sổ với kích thước pool size là 2×2 để lấy giá trị lớn nhất trong 4 giá trị mà cửa sổ này quét qua trên ma trận đầu ra của Convolutional Layer.
4. Sau Pooling Layer, chúng tôi sử dụng một Dropout Layer với

giá trị Dropout được thiết lập là 0,25. Nó được cấu hình để loại trừ ngẫu nhiên 25% tổng số các nơ-ron trong layer để giảm vấn đề overfitting.

5. Để chắt lọc được nhiều thông tin hữu ích từ ảnh, chúng tôi xây dựng mạng CNN sâu hơn bằng cách bổ sung thêm một số layercủa mạng, chúng bao gồm các layer sau: Convolutional, Pooling, Dropout. Trong đó, Convolutional Layer sẽ sử dụng nhiều filter với kích thước bằng nhau, Pooling Layer sử dụng cửa sổ với kích thước pool size là 2×2, Dropout Layer với giá trị Dropout là 0,25.
6. Tiếp theo, chúng tôi sử dụng một layer chuyển đổi dữ liệu ma trận 2D thành một véc-tơ gọi là Flatten. Kết quả chúng tôi thu được một véc-tơ các giá trị đặc trưng của ảnh, véc-tơ này phù hợp với định dạng đầu vào của một MLP.
7. Sau khi thu được véc-tơ đặc trưng của ảnh qua các layer của CNN, chúng tôi sử dụng MLP làm bộ phân loại để phân loại ảnh. MLP của chúng tôi sử dụng nhiều layer ẩn với số nơ-ron được cấu hình trong quá trình thực nghiệm. Do đây là nhiệm vụ phân loại đa lớp (10 lớp) nên đầu ra của MLP chúng tôi sử dụng 10 nơ-ron và một hàm kích hoạt *softmax* để đưa ra các dự đoán là các giá trị xác suất cho mỗi lớp.

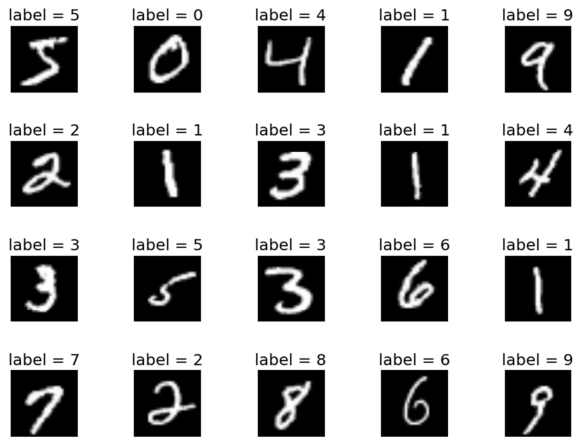
5. Thực nghiệm

5.1. Tập dữ liệu

Trong phần thực nghiệm, chúng tôi sử dụng tập dữ liệu MNIST (Yann LeCun, Corinna Cortes và Christopher, 1989). Đây là tập dữ liệu thường dùng để đánh giá hiệu quả của cácmô hình nhận dạng ký tự số viết tay. Tập dữ liệu MNIST có nguồn gốc từ tập NIST do tổ chức National Institute of Standards and Technology (NIST) cung cấp, sau đó được LeCun cập nhật và chia thành 2 tập riêng biệt:

Tập dữ liệu huấn luyện gồm có 60.000 ảnh kích thước 28×28 của chữ số viết tay được dùng cho việc huấn luyện mô hình học máy. Tất cả các ảnh trong tập dữ liệu đều được căn chỉnh và biến đổi thành dữ liệu dạng điểm gồm 60.000 phần tử (ký tự số) có 784 chiều là giá trị mức xám của các điểm ảnh, 10 lớp (giá trị từ 0 đến 9).

Tập dữ liệu kiểm tra gồm có 10.000 ảnh của chữ số viết tay được dùng cho việc kiểm thử. Các ảnh trong tập dữ liệu kiểm tra cũng được biến đổi và căn chỉnh thành dữ liệu điểm gồm 10.000 phần tử trong 784 chiều, 10 lớp (giá trị từ 0 đến 9). Hình 6 là ví dụ về một số mẫu của tập dữ liệu.



Hình 6. Ví dụ về một số mẫu của tập dữ liệu MNIST

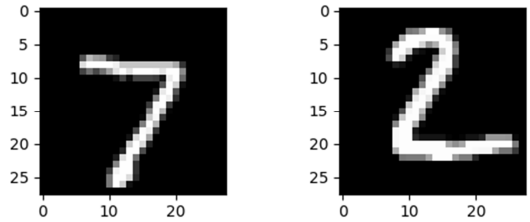
5.2 Chuẩn bị dữ liệu cho huấn luyện mô hình

Để tải tập dữ liệu MNIST về máy tính, chúng tôi sử dụng thư viện học sâu của Keras. Tập dữ liệu được tải xuống tự động lần đầu tiên được gọi và lưu trữ trong thư mục chính của người dùng trong ~/.keras/datasets/mnist.pkl.gz dưới dạng một tệp tin. Điều này rất thuận tiện cho việc thực nghiệm các mô hình học máy. Ở đây, chúng tôi sử dụng đoạn chương trình như đã được viết trong [3] để để tải xuống và đưa ra 2 hình ảnh đầu tiên trong tập dữ liệu kiểm tra.

```
import numpy

from keras.datasets import mnist
import matplotlib.pyplot as plt
X_train, y_train), (X_test, y_test) = mnist.load_data()
plt.subplot(221)
plt.imshow(X_test[0], cmap=plt.get_cmap('gray'))
plt.subplot(222)
plt.imshow(X_test[1], cmap=plt.get_cmap('gray'))
plt.show()
```

Thực thi đoạn chương trình trên, tập dữ liệu MNIST sẽ được tải và lưu trữ trên máy tính. Đồng thời, chương trình cũng đưa ra 2 hình ảnh đầu tiên của tập dữ liệu kiểm tra. Kết quả minh họa trong hình 7.



Hình 7. Ví dụ về 2 mẫu dữ liệu từ tập dữ liệu kiểm tra của MNIST

5.3. Xây dựng và huấn luyện mô hình

Sau khi dữ liệu MNIST đã được tải về máy tính, chúng tôi sẽ huấn luyện một mô hình DCNN trên tập dữ liệu này. Trong phần này, chúng tôi sẽ xây dựng một DCNN bằng cách kết hợp các lớp của CNN với MLP (như đã được trình bày trong mục 4).

Trong các thực nghiệm, chúng tôi xây dựng mô hình với việc áp dụng một số thuật toán tối ưu hóa và thiết lập các giá trị mức học (*learning rate*) khác nhau. Chúng tôi cũng thiết lập kích thước cho các filter lần lượt là 5×5 và 3×3. Dưới đây là mô hình huấn luyện với việc sử dụng hàm lỗi *logarithmic*, thuật toán tối ưu hóa *adam* và giá trị *learning_rate*được thiết lập là 0,01.

```
def DCNN1_model(model = Sequential()
model.add(Conv2D(64, (5, 5), input_shape=(1, 28, 28), activation = 'relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

```
model.add(Conv2D(32, (3, 3), activation= ‘relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
# MLP with 3 hidden layer
model.add(Dense(375, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(225, activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(135, activation='relu'))
model.add(Dropout(0.25))model.add(Dense(10, activation= ‘softmax’))
model.compile(loss= ‘categorical_crossentropy’, optimizer=
‘adam’,optimizer_params={'learning_rate':0.01},
metrics=['accuracy']
return model
```

5.4. Đánh giá mô hình

Để đánh giá hiệu suất của mô hình đã xây dựng, chúng tôi sử dụng tập dữ liệu kiểm tra như đã được trình bày trong mục 5.1. Chúng tôi đánh giá mô hình với các giá trị epoch là 10 và batch size là 256.

```
model = DCNN1_model()
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10,
batch_size=256, verbose=2)
scores = model.evaluate(X_test, y_test, verbose=0)
print("DCNN1 Error: %.2f%%" % (100-scores[1]*100))
```

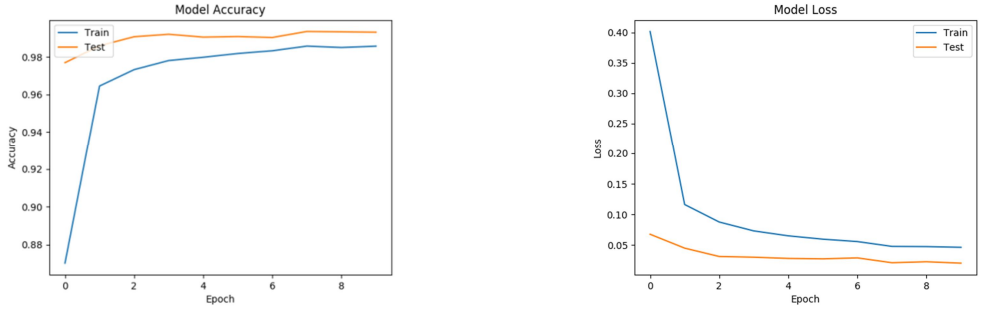
Khi thực thi chương trình, độ chính xác của mô hình huấn luyện sẽ được in ra theo từng epoch và ở cuối lỗi phân loại cũng được in ra. Bảng 1 tóm tắt độ chính xác phân loại, tỷ lệ lỗi và thời gian trung bình khi thực thi mô hình với các cấu hình mạng khác nhau.

Bảng 1. Các kết quả thực nghiệm với các cấu hình mạng khác nhau

Cấu hình mạng			**Kết quả phân loại**		**Thời gian trung bình thực hiện 1 epoch (giây)**
Số filter trong mỗi Convolutional Layer	**Hàm tối ưu hóa**	**Mức học**	**Độ chính xác cao nhất (%)**	**Tỷ lệ lỗi (%)**	
32; 16	sgd	0,01	96,85	3,13	402,1
32; 16	adam	0,01	99,20	0,80	367,8
32; 16	adam	0,05	99,15	0,85	386,0
32; 16	adam	0,001	99,06	0,94	397,2
32; 16	adam	0,005	99,09	0,91	463,0
64; 32	adam	0,01	99,34	0,74	844,5

Từ các kết quả thực nghiệm cho thấy, mô hình chúng tôi xây dựng đạt kết quả phân loại cao nhất với độ chính xác là 99,34% và tỷ lệ lỗi thấp nhất là 0,74% khi sử dụng các filter

với số lượng tương ứng là 64 và 32, hàm tối ưu hóa *adam*, mức học là 0,01. Hình 8 là kết quả khi đánh giá mô hình này ở mỗi giá trị epoch khác nhau.



Hình 8. Độ chính xác và tỷ lệ lỗi ở mỗi epoch khi đánh giá mô hình

Chúng tôi cũng thực hiện so sánh kết quả nghiên cứu của chúng tôi với các kết quả nghiên cứu trước đó trên cùng tập dữ liệu huấn luyện và kiểm tra. Bảng 2 trình bày các kết quả so sánh này.

Bảng 2. So sánh với các kết quả nghiên cứu khác

Nghiên cứu của tác giả	Độ chính xác cao nhất (%)	Tỷ lệ lỗi (%)
Jason Brownlee [3]	99,31	0,82
LeCun và các tác giả [6]		0,95
Kasun và các tác giả [4]	99,03	0,97
Hinton và các tác giả [2]		1,25
Tapson và các tác giả [9]	96,00	1,52
Tapson và các tác giả [10]	90,00	2,75
Norhidayu và các tác giả [5]	99,26	
Nghiên cứu của chúng tôi	99,34	0,74

Từ kết quả so sánh trong Bảng 2 cho thấy mô hình đề xuất của chúng tôi cho hiệu quả cao hơn so với các mô hình đã được xây dựng gần đây. Mô hình của chúng tôi đạt độ chính xác phân loại là 99,34% và tỷ lệ lỗi thấp nhất là 0,74%.

Tuy nhiên, trong các thực nghiệm chúng tôi cũng thấy rằng thời gian để huấn luyện mô hình tăng đáng kể khi tăng số lượng các filter trong mỗi ConvolutionalLayer. Cụ thể khi sử dụng các filter với số lượng là 32 và 16 trong các Convolutional Layer, mô hình sẽ mất khoảng 367,8 giây cho mỗi epoch, trong khi tăng số filter lên 64, 32 thì thời gian cho mỗi epoch là khoảng 844,5 giây. Như vậy, để đạt được kết quả cao thì đòi hỏi phải mất nhiều thời gian để huấn luyện mô hình.

6. Kết luận và hướng phát triển

Trong bài báo này, chúng tôi đã đề xuất phương pháp mới nhằm giải quyết bài toán nhận dạng chữ số viết tay. Chúng tôi đã xây dựng mô hình học sâu với việc kết hợp các lớp của CNN với MLP. Trong đó, các lớp của CNN có chức năng như một bộ trích trọn các đặc trưng ảnh một cách tự động. MLP có chức năng là một bộ phân loại, nó nhận đầu vào là véc-tơ đặc trưng sinh ra sau các lớp của CNN và cho kết quả là các giá trị phân loại (10 lớp). Chúng tôi sử dụng tập dữ liệu chữ số viết tay thông dụng MNIST để huấn luyện và đánh giá mô hình

đề xuất. Các kết quả khi huấn luyện và kiểm tra mô hình đã chứng minh rằng đề xuất của chúng tôi cho kết quả phân loại cao hơn so với nhiều nghiên cứu trước đó trên cùng tập dữ liệu. Từ thành công bước đầu trong việc áp dụng học sâu vào giải quyết bài toán nhận dạng chữ số viết tay, chúng tôi sẽ mở rộng bài toán để có thể nhận dạng được tất cả các kí tự chữ viết tay.

TÀI LIỆU THAM KHẢO

[1] Ana Riza F. Quiros, Rhen Anjerome Bedruz, Aaron Christian Uy, Alexander Abad, Argel Bandala, Elmer P. Dadios, Arvin Fernando, De La Salle (2017), *A kNN-based approach for the machine vision of character recognition of license plate numbers*. Region 10 Conference TENCON 2017, IEEE, pp. 1081-1086.

[2] Hinton GE, Osindero S, Teh YW (2006), *A fast learning algorithm for deep belief nets. Neural computation*, pp.1527–1554. doi: 10.1162/neco.2006.18.7.1527.

[3] ason Brownlee (2017), *Handwritten Digit Recognition using Convolutional Neural Networks in Python with Keras*. https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/.

[4] Kasun LLC, Zhou H, Huang GB, Vong CM (2013), *Representational learning with extreme learning machine for big data*. IEEE Intelligent Systems, pp. 31-34.

[5] Norhidayu binti Abdul Hamid, Nilam Nur Binti Amir Sjarif (2017), *Handwritten Recognition Using SVM, KNN and Neural Network*. eprint arXiv:1702.00723.

[6] LeCun Y, Bottou L, Bengio Y, Haffner P (1998), *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, pp. 2278–2324. doi: 10.1109/5.726791.

[7] Peter Burrow (2004), *Arabic handwriting recognition*. Technical report, School of Informatics, University of Edinburgh.

[8] Souici-Meslati, Sellami M (2006), *A Hybrid NeuroSymbolic Approach for Arabic Handwritten Word Recognition*. Journal of Advanced Computational Intelligence and Intelligent Informatics, FujiPress, Vol. 10, No. 1, pp. 17-25

[9] Tapson J, de Chazal P, van Schaik A (2014), *Explicit computation of input weights in extreme learning machines*. In: Proc. ELM2014 conference, arXiv:1406.2889.

[10] Tapson J, van Schaik A (2013), *Learning the pseudoinverse solution to network weights*. Neural Networks, pp. 94-100. doi: 10.1016/j.neunet.2013.02.008 PMID: 23541926.

HANDWRITTEN DIGIT RECOGNITION BASING ON DEEP CONVOLUTIONAL NEURAL NETWORK

Nguyen Van Tu, Hoang Thi Lam, Nguyen Thi Thanh Ha
Tay Bac University

Abstract: *In the field of image processing, pattern recognition has been one of the greatest challenges of researchers in recent years. The goal of pattern recognition is to extract the features of the image to classify the samples into different classes. A well-known problem in this area is the handwriting digit recognition, in which the digits must be assigned to one of the 10 classes using some classification method. Our aim in this paper is to present a deep learning method instead of existing statistical techniques to solve the problem of handwritten digit recognition. We built a deep convolution neural network model with multiple layers of the network to automatically extract the best features from the image. At the same time, we also combined convolutional neural networks and Multi-layer Perceptron to improve the performance of the model. The experimental results obtained on the dataset MNIST gave the highest accuracy of 99,34% and the error rate of 0,74%. These results show that our proposed model yields is much higher than previous models on the same dataset.*

Keywords: *Handwritten digit recognition, convolution neural network, multi-layer perceptron, classification*.