Nghiên cứu nhận dạng số viết tay và ứng dụng giải SUDOKU

The study on handwritten digit recognition and its application in solving SUDOKU

ThS. Đặng Thị Dung[1,*], Nguyễn Huỳnh Thiên Quốc[1] , Đặng Hào Phú[1]

[1] Khoa Công nghệ Thông tin, Trường Đại học Kỹ thuật - Công Nghệ Cần Thơ.

*Email: dtdung@ctuet.edu.vn

■Nhận bài: 11/05/2024 ■Sửa bài: 20/05/2024 ■Duyệt đăng: 18/06/2024

TÓM TẮT

Bài nghiên cứu tập trung nghiên cứu các mô hình máy học CNN, KNN, Mobilenet và áp dụng vào giải Sudoku bằng các phương pháp tiền xử lý dữ liệu giúp áp dụng các mô hình nhận diện số viết tay, sau khi cho các mô hình lần lượt huấn luyện và kiểm tra trên tập dữ liệu MNIST kết quả cho thấy mô hình CNN là 99,08 % mô hình Mobilenet là 98,36% và mô hình KNN là 97,7%. Qua bài nghiên cứu giúp có thể thấy được điểm mạnh và điểm yếu của các mô hình giúp độc giả có thể đưa ra gợi ý về việc áp dụng mô hình nào cho phù hợp vào từng đề tài hay dự án khác nhau .

Từ khóa: *CNN : Convolutional Neural Network, KNN: K-Nearest Neighbors, MNIST: Modified National Institute of Standards and Technology database.*

ABSTRACT

The study focuses on researching machine learning models such as CNN, KNN, and MobileNet, and applying them to solve Sudoku using data preprocessing methods to enable handwritten digit recognition. After training and testing the models on the MNIST dataset, the results showed that the CNN model achieved 99.08%, the MobileNet model achieved 98.36%, and the KNN model achieved 97.7%. Through this study, we can see the strengths and weaknesses of each model, providing suggestions on which model is suitable for different topics or projects.

Keywords: *CNN : Convolutional Neural Network, KNN: K-Nearest Neighbors, MNIST: Modified National Institute of Standards and Technology database.*

1. GIỚI THIỆU

Với các nhu cầu thị trường khác nhau hiện nay đã kích thích sự đa dạng của các phương pháp máy học khác nhau, từ những phương pháp máy học phức tạp giúp cho việc nhận dạng chuẩn xác và nhanh chóng đến các mô hình máy học đơn giản và dùng ít tài nguyên của máy nhưng vẫn hoàn thành nội dung công việc một cách tốt nhất và giúp nâng cao năng suất trong công việc. Để có thêm nhiều dữ liệu trực quan về các mô hình giúp các độc giả có góc nhìn về các mô hình, nhóm tác giả đã tiến hành nghiên cứu và so sánh các mô hình máy học hiện đại ngày nay với các mô hình trước đây, nhóm tác giả chọn đại diện những mô hình máy học đã được đánh tốt như mô hình máy học CNN (Convolutional Neural Network) , mô hình máy học KNN (K-Nearest Neighbors) và mô hình máy học MobileNet [1-3].

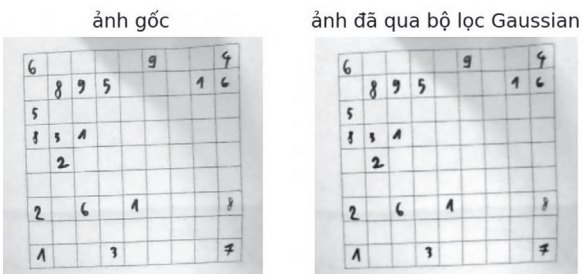
2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Tiền xử lý hình ảnh:

Trong công nghệ máy học (Machine learning), tiền xử lý ảnh đầu vào là một trong

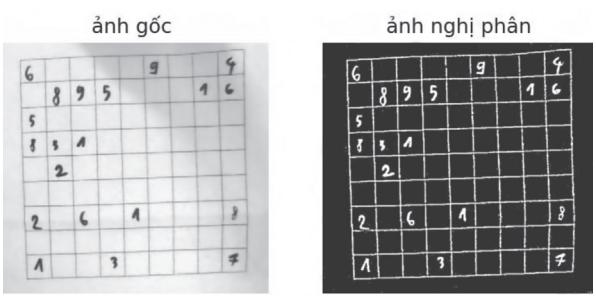
những bước quan trọng không thể thiếu giúp cải thiện chất lượng ảnh tốt hơn và định dạng lại ảnh đầu vào, giúp cho hệ thống nhanh chóng nhận diện ảnh và làm tăng hiệu suất nhận diện ảnh.

Để giúp mô hình có thể xử lý ảnh ta cần phải tiền xử lý ảnh như: làm mờ ảnh, làm giảm các biến động đột ngột giữa các pixel trong ảnh, tạo ra hiệu ứng mịn và trơn. Điều này giúp làm mịn ảnh và giảm độ nổi bật của đối tượng trong ảnh giúp giảm các chi tiết không mong muốn trong ảnh, làm cho ảnh trở nên đơn giản, dễ hiểu hơn.



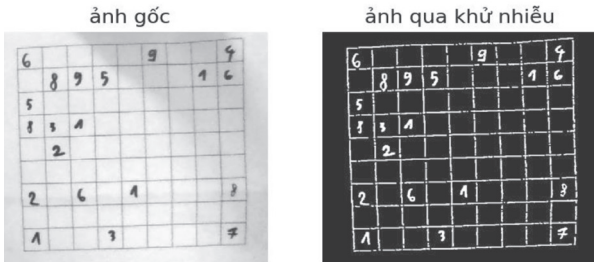
***Hình: 1**. Ảnh đã được làm mờ*

Nhị phân hóa ảnh giúp trích xuất các đặc trưng của đối tượng trong ảnh một cách dễ dàng hơn. Các đặc trưng này có thể sử dụng trong các nhiệm vụ: nhận dạng đối tượng, phát hiện biên...



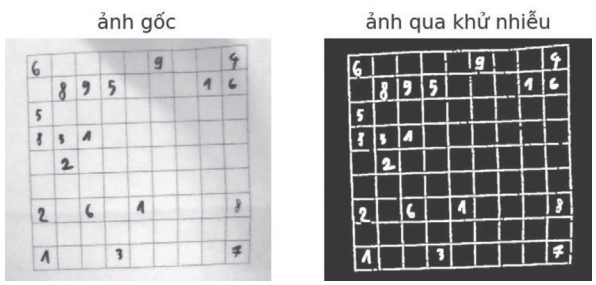
***Hình: 2**. Nhị phân hóa ảnh*

Lọc nhiễu (khử nhiễu) là một vấn đề thường gặp trong nhận dạng. Có nhiều loại nhiễu như: nhiễu muối tiêu, nhiễu đốm, nhiễu vệt,... Trong ứng dụng này, nhóm tác giả sử dụng phương pháp biến đổi hình thái học (Morphological transformations). Trước tiên, nhóm tác giả tạo ra một phần tử cấu trúc (Structuring element) có kích thước là (2, 2). Kết quả khi áp dụng phương pháp biến đổi giúp làm mượt hơn hình ảnh và loại bỏ nhiễu[4].



***Hình: 3**. Ảnh sau khi qua khử nhiễu*

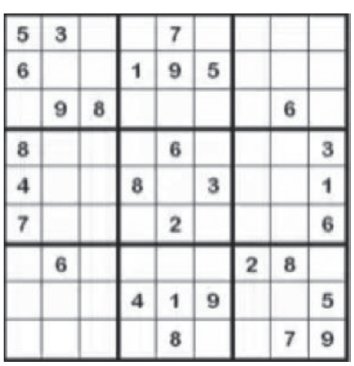
Để khắc phục các đường nét bị xói mòn do phần tử cấu trúc (Structuring element) hoặc do chất lượng ảnh, nhóm nghiên cứu tận dụng lại phần tử cấu trúc (Structuring element) để mở rộng biên ảnh bằng cách thực hiện phép mở rộng (Dilation) cũng là một trong những phép biến đổi hình thái học được sử dụng phổ biến trong xử lý hình ảnh giúp các số đầy đặng hơn.



***Hình: 4**. Biên ảnh được mở rộng*

2.2. Trích xuất lưới Sudoku:

Để trích xuất lưới Sudoku chúng ta cần tìm các viền trên ảnh, sử dụng tìm kiếm đường viền để tìm các đường viền trên hình ảnh.



***Hình: 5**. Ảnh SUDOKU*

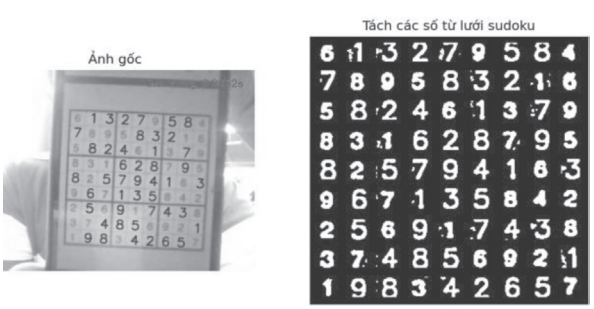
Tiếp theo, hàm lặp qua từng đường viền để tìm đa giác có bốn góc và diện tích lớn hơn một ngưỡng nhất định. Nếu tìm thấy đa giác phù hợp, được xác định là hình Sudoku cần tìm. Sau đó xác định tọa độ của bốn góc của Sudoku. Quá trình này sử dụng các chiến lược như tìm góc trên trái với tổng tọa độ x và y

nhỏ nhất, góc trên phải với hiệu tọa độ x và y lớn nhất và tương tự cho các góc còn lại.

Điều chỉnh độ nghiêng lưới Sudoku do lúc quét vào không cẩn thận hoặc do rung lắc, lưới Sudoku bị lệch so với lề chuẩn một góc α, điều này tạo thử thách cho bước tách số, thỉnh thoảng không thể tách được. Để khắc phục, trước tiên hàm chuyển đổi các tọa độ của các góc từ kiểu dữ liệu chuỗi các số thành kiểu dữ liệu float32. Sau đó, tính toán chiều rộng tối ưu cho hình vuông bao quanh Sudoku bằng cách xác định khoảng cách lớn nhất giữa các cạnh của hình vuông.

Kế tiếp, hàm tạo một mảng mới chứa tọa độ của bốn góc của hình vuông với chiều rộng tính được. Sau đó, sử dụng hàm để tính toán ma trận biến đổi tạo thành hình vuông từ hình ảnh gốc. Cuối cùng, áp dụng ma trận biến đổi để biến đổi hình ảnh gốc thành hình vuông.

Để trích xuất các số khỏi lưới chúng ta cần phải bỏ các đường lưới khỏi ảnh bằng cách tạo ra một mặt lưới Sudoku, sau đó sử dụng hàm để tách các đường lưới chừa lại các số trên ảnh và bắt đầu cắt các số trong ảnh để thành một mảng chứa các ảnh số[4].



(a) Ảnh gốc
(b) Sau khi tách các số

***Hình: 6**. Hình Sudoku sau khi qua xử lý*

2.3 Giới thiệu dữ liệu huấn luyện:

Bộ dữ liệu MNIST là tập dữ liệu quan trọng và nổi tiếng trong lĩnh vực học máy và thị giác máy tính. Bao gồm 60,000 hình ảnh chữ số viết tay cho quá trình huấn luyện và 10,000 hình ảnh cho quá trình kiểm tra. Mỗi hình ảnh có kích thước là 28x28 Pixel và được gán nhãn chính xác cho mỗi chữ số từ 0 đến 9. Bộ dữ liệu này được sử dụng rộng rãi để huấn luyện và đánh giá các mô hình học máy trong việc nhận dạng và phân loại chữ số viết tay[5].

2.4. Các giải thuật phân loại văn bản:

Có nhiều thuật toán được dùng để nhận dạng số viết tay. Trong bài viết này, nhóm tác giả đã sử dụng ba thuật toán Mobilenet, KNN, CNN. Trong bài báo "Reconnaissance des chiffres manuscrits par un système immunitaire artificiel flou" được đăng trên hội nghị chuyên đề quốc tế ISKO-Maghreb vào tháng 11 năm 2014 của nhóm tác giả Merabti H, Kouahla M. N, Seridi H đã sử dụng mô hình KNN để nhận dạng số viết tay trong đó sử dụng hai phương pháp miễn dịch tự nhiên (NISs) và nhân tạo (AIRS) để học tập. Sau khi sử dụng AIRS_Fuzzy-KNN đã cho ra kết quả là 96,5 %[6].

Bài báo "Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition" được đăng trên tạp chí 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA) vào ngày 06/10/2020 của nhóm tác giả Wenfei Liu, Jingcheng Wei, Qingmin Meng đã sử dụng bốn mô hình KNN, SVM, BP và CNN để nhận diện chữ viết tay trên MNIST với độ chính xác lần lượt là 94.6, 94.1, 96.6 và 97.7[7].

Ngoài ra trong bài báo "Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM)" được đăng trên tạp chí 2017 Fifth International Conference on Information and Communication Technology (ICoICT) vào ngày 19/10/2017 của hai tác giả Darmatasia và Mohamad Ivan Fanany đã sử dụng mô hình CNN kết hợp SVM để nhận diện chữ viết tay trên tập dữ liệu NIST SD 19 thế hệ 2 với độ chính xác là 98.85[8].

Ngày 25/04/2019 nhóm tác giả Preman Ghadekar; Shubham Ingole; Dhruv Sonone đã sử dụng mô hình KNN và SVM để nhận diện chữ viết tay trên MNIST và EMNIST với độ chính xác là 97.33 và 97.74 được thể hiện trong bài báo "Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM classifier." được đăng trên tạp chí 2018 Fourth International Conference

on Computing Communication Control and Automation (ICCUBEA)[9].

Trong bài báo "Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM classifier." được đăng trên tạp chí 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) vào ngày 25/04/2019 của nhóm tác giả Preman Ghadekar; Shubham Ingole; Dhruv Sonone đã sử dụng mô hình KNN và SVM để nhận diện chữ viết tay trên MNIST và EMNIST với độ chính xác là 97.33 và 97.74[10].

Trong nghiên cứu này, nhóm tác giả tìm hiểu chi tiết các mô hình và ứng dụng các mô hình vào bài toán Sudoku.

2.4.1. Thuật toán KNN:

Thuật toán KNN là một phương pháp phân loại các đối tượng dựa trên khoảng cách giữa đối tượng cần phân loại và các đối tượng trong tập huấn luyện. Nguyên lý hoạt động của K-Nearest Neighbors (KNN) dựa trên giả thuyết rằng các điểm dữ liệu có đặc điểm tương đồng sẽ nằm gần nhau trong không gian đặc trưng. Khi gặp một điểm dữ liệu mới, KNN sẽ xác định những điểm dữ liệu gần nhất trong không gian đặc trưng và sử dụng nhãn của những điểm này để dự đoán nhãn cho điểm dữ liệu mới[11].

Các bước cần để hoạt động KNN:

Bước 1: Chuẩn bị dữ liệu dùng để huấn luyện.

Bước 2: Chọn số lượng điểm láng giềng.

Trong KNN, số lượng điểm láng giềng cần được chọn để dự đoán nhãn của một điểm dữ liệu mới. Số lượng điểm láng giềng này được gọi là tham số K của thuật toán.

Bước 3: Khi có một điểm dữ liệu mới, ta cần tính khoảng cách giữa nó với các điểm dữ liệu trong tập huấn luyện. Một số phương pháp tính khoảng cách phổ biến bao gồm: khoảng cách Euclid, khoảng cách Mahalanobis hay khoảng cách cosine.

Bước 4: Chọn K điểm láng giềng gần nhất

Sau khi tính khoảng cách giữa các điểm dữ liệu ở bước 3, ta sẽ chọn ra K điểm láng giềng gần nhất với điểm dữ liệu mới. Những điểm này sẽ được dùng để dự đoán nhãn cho điểm dữ liệu mới.

Để có thể đánh giá khách quan và nhằm tăng hiệu suất của mô hình, chúng tôi đã cho mô hình chạy với các khóa k từ 1 đến 10.

```
Accuracy of KNN with 1 neighbors: 97.2 %. Fit in 0.1 s. Prediction in 41.3 s
Accuracy of KNN with 2 neighbors: 96.9 %. Fit in 0.0 s. Prediction in 41.4 s
Accuracy of KNN with 3 neighbors: 97.3 %. Fit in 0.1 s. Prediction in 41.2 s
Accuracy of KNN with 4 neighbors: 97.2 %. Fit in 0.0 s. Prediction in 41.8 s
Accuracy of KNN with 5 neighbors: 97.3 %. Fit in 0.1 s. Prediction in 41.4 s
Accuracy of KNN with 6 neighbors: 97.2 %. Fit in 0.1 s. Prediction in 41.3 s
Accuracy of KNN with 7 neighbors: 97.3 %. Fit in 0.1 s. Prediction in 41.2 s
Accuracy of KNN with 8 neighbors: 97.1 %. Fit in 0.0 s. Prediction in 41.3 s
Accuracy of KNN with 9 neighbors: 97.2 %. Fit in 0.0 s. Prediction in 45.9 s
Accuracy of KNN with 10 neighbors: 97.1 %. Fit in 0.1 s. Prediction in 41.4 s
```

***Hình:** 7 Kết quả mô hình với giá trị k từ 1-10*

Chúng ta có thể thấy được mặc dù không dán kết quả cho k lớn hơn việc tăng giá trị tham số, thậm chí lên 10, cũng không làm tăng hiệu suất mô hình. Để kết quả tốt hơn được đưa ra bằng phép đo trọng số 'khoảng cách' - nhờ đó chúng tôi đã ghi lại hai kết quả phân loại tốt nhất: 97,3% với k = 3 và độ chính xác tương tự với k = 5 và k=7. Vì vậy để tối ưu mô hình chúng tôi chọn giá trị k=3.

Bước 5: Dự đoán nhãn cho điểm dữ liệu mới

Mặc dù KNN không thực hiện quá trình huấn luyện như các mô hình khác, quá trình này vẫn quan trọng để mô hình có thể thực hiện dự đoán sau này. Trong quá trình huấn luyện, KNN sẽ chỉ đơn giản là lưu trữ toàn bộ tập dữ liệu huấn luyện, cùng với nhãn tương ứng của từng điểm dữ liệu. Khi cần dự đoán nhãn cho một điểm dữ liệu mới, KNN sẽ dùng thông tin này để tính toán và chọn ra các láng giềng gần nhất. Mặc dù quá trình huấn luyện không "học" ra một mô hình tường minh, nó vẫn cần thiết để mô hình có thể thực hiện dự đoán một cách chính xác.

2.4.2. Thuật toán CNN:

Sự phát triển đột phá của CNN bắt đầu từ năm 2012, khi Alex Krizhevsky, Ilya Sutskever và Geoffrey Hinton giành chiến thắng trong cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC), CNN đã trở thành một phần không thể thiếu trong các ứng dụng thị giác máy tính như phân

loại ảnh, nhận diện khuôn mặt, nhận dạng vật thể, và nhiều ứng dụng khác. Các biến thể và cải tiến của CNN tiếp tục được phát triển, giúp nó trở thành một công cụ quan trọng trong lĩnh vực trí tuệ nhân tạo và học máy[12].

Mạng nơ-ron tích chập (CNN) bao gồm một số lớp tích chập kết hợp với các hàm kích hoạt phi tuyến, chẳng hạn như ReLU hoặc Tanh, nhằm tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Trong quá trình huấn luyện, CNN tự động học được các thông số cho các bộ lọc (filter). Ví dụ, trong tác vụ phân loại ảnh, CNN sẽ cố gắng tìm ra các thông số tối ưu cho các bộ lọc theo thứ tự từ Raw Pixel (Pixel thô), Edges (Cạnh), Shapes (Hình dạng), Facial (Khuôn mặt), đến High-level Features (Đặc trưng cấp cao). Lớp cuối cùng của CNN được sử dụng để phân loại ảnh.[3].

Điểm quan trọng nhất của mạng Neural nằm ở các giá trị trọng số và tập trọng số, nếu các trọng số này có giá trị đúng đắn thì mạng sẽ phân lớp hoàn hảo. Chúng ta cần từng bước chỉnh sửa tập trọng số này bằng cách huấn luyện mạng dựa trên các cặp mẫu đầu vào được đưa vào mạng, sau đó tính sai số giữa kết xuất của mạng với giá trị mong đợi mà chúng ta biết trước. Nhóm nghiên cứu sử dụng phương pháp huấn luyện mạng học dựa trên độ dốc của mặt lỗi.

Để thay đổi trọng số của mạng Neural, chúng ta phải tính độ dốc mặt lỗi trên từng trọng số hay nói cách khác là tính đạo hàm riêng phần của hàm lỗi trên từng trọng số, thuật toán lan truyền ngược sai số giúp chúng ta tính đạo hàm này.

Công thức hàm lỗi được tính như sau:

$$c = \frac{1}{2}\sum ||y(x) - a(x)|| \quad (1)$$

+ Bước 1: Nhập mẫu x.

+ Bước 2: Lan truyền tiến.

+ Bước 3: Lỗi output.

+ Bước 4: Lan truyền ngược lỗi.

+ Bước 5: Kết quả.

Cho mỗi mẫu huấn luyện mạng, chúng ta huấn luyện mạng theo 3 bước sau:

+ Bước 1: Nhập tập huấn luyện

+ Bước 2: Với mỗi mẫu x: Gán giá trị của x tương ứng trong tầng input và thực hiện các bước

- Lan truyền tiến: Với mỗi l=2, 3, …, L, tính .

- Lỗi output : Tính vector (z).

- Lan truyền ngược lỗi: Với l = L-1, L-2, …, 2, tính ((w)).

+ Bước 3: Với mỗi l = L, L-1, …, 2, cập nhật các trọng số theo quy tắc [13]

$$\omega_{new} = \omega_{old} - n \times \nabla_{\omega} L \quad (2)$$

$$b_{new} = b_{old} - n \times \nabla_{b} L \quad (3)$$

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_6 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_7 (Conv2D)	(None, 13, 13, 64)	18496
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_8 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_8 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten_2 (Flatten)	(None, 512)	0
dense_8 (Dense)	(None, 64)	32832
dense_9 (Dense)	(None, 128)	8320
dense_10 (Dense)	(None, 10)	1290

Total params: 135114 (527.79 KB)
Trainable params: 135114 (527.79 KB)
Non-trainable params: 0 (0.00 Byte)

Hình: 8 *Cấu trúc mô hình CNN*

2.4.3 Thuật toán Mobilenet.

MobileNet được phát triển bởi một nhóm các nhà nghiên cứu tại Google, bao gồm Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, và Hartwig Adam. Công trình nghiên cứu của họ được công bố lần đầu vào năm 2017 và đã góp phần quan trọng vào việc đưa trí tuệ nhân tạo gần hơn với cuộc sống hàng ngày của tất cả mọi người thông qua việc phát triển những mô hình nhỏ gọn và hiệu quả [14].

MobileNet đã có sự ứng dụng rộng khắp trong nhiều lĩnh vực của trí tuệ nhân tạo. Một trong những ứng dụng quan trọng nhất của MobileNet là trong việc nhận dạng và phân loại ảnh trên các thiết bị di động. Nó cũng được dùng trong các ứng dụng nhận diện khuôn mặt, phát hiện đối tượng, và phân đoạn hình ảnh. Sự nhẹ nhàng và hiệu quả của MobileNet biến nó trở thành một lựa chọn hợp lý cho các ứng dụng AI trên các thiết bị di động (tablet, smartphone) và các thiết bị IoT và tới hiện tại MobileNet đã xuất hiện rất nhiều bản nâng cấp. Mô hình MobileNet mới nhất hiện nay tính tới cuối năm 2023 đã cho ra bảng MobileNetV3. Tuy nhiên trong bài viết này, nhóm nghiên cứu sử dụng bảng MobileNetV1 vì mặc dù các phiên bản MobileNet mới có hiệu xuất tốt hơn nhưng do dùng những kỹ thuật phức tạp đã làm tăng các chi phí tính toán, ngoài ra nhóm tác giả đã từng thử nghiệm cho mô hình MobileNetV2 huấn luyện trên MNIST nhưng do quy mô bài toán nên hiệu xuất giữa mô hình MobilenetV2 với MobilenetV1 chênh lệch không đáng kể nên nhóm nghiên cứu quyết định sử dụng mô hình MobilenetV1.

Trong MobileNet V1, hộp tích chập trong hình ảnh đã cho bao gồm các tích chập theo chiều sâu và điểm được lặp lại 13 lần sau lớp tích chập ban đầu [15].

Mô hình gồm có 30 lớp:

Lớp 1: Tích chập Layer (Stride = 2).

Lớp 2: Depthwise layer (Stride = 1).

Lớp 3: Pointwise layer.

Lớp 4: Depthwise layer (Stride = 2).

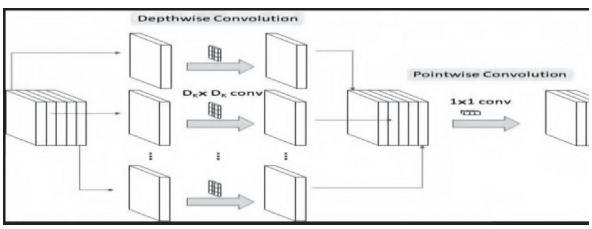
Lớp 5: Pointwise layer

Các lớp từ lớp 6 đến lớp 29 thường là một chuỗi lặp lại của Depthwise và Pointwise tích chập Layers.

Lớp 30: Softmax, dùng để phân lớp.

Dựa vào kiến trúc của mô hình có thể thấy được MobileNet được xây dựng dựa trên kiến trúc CNN. Nó sử dụng các lớp tích chập, các lớp Normalization và các lớp Pooling như các thành phần cơ bản của một mạng CNN. Trong khi các mạng CNN truyền thống có thể có hàng triệu hoặc thậm chí hàng tỷ tham số, điểm đặc biệt giúp MobileNet vượt trội hơn CNN là sử dụng các kỹ thuật như tích chập riêng biệt theo chiều sâu (Depthwise Separable tích chập) để giảm số lượng tham số và lượng tính toán cần thiết.

Kỹ thuật tích chập riêng biệt theo chiều sâu (Depthwise Separable tích chập) là một tích chập theo chiều sâu được kết hợp với một tích chập theo điểm như sau:



Hình 9: *Kỹ thuật tích chập riêng biệt theo chiều sâu của Mobilenet*

Convolution theo chiều sâu: Là một phép convolution không gian kích thước DK×DK theo từng kênh riêng biệt.

Convolution theo điểm: Convolution có kích thước 1x1 (như trong hình 5.2). Với M là số lượng kênh đầu vào, N là số lượng kênh đầu ra, Dk là kích thước kernel, Df là kích thước feature map (ví dụ, với tập dữ liệu ImageNet, kích thước đầu vào là 224, do đó feature map ban đầu có Df = 224), ta có thể tính được: [16-17].

Số lượng phép tính của convolution theo chiều sâu là:

$$\mathcal{D}_k.\mathcal{D}_k.M.\mathcal{D}_f.\mathcal{D}_f \quad (4)$$

Số lượng phép tính của Pointwise convolution là:

$$M.N.\mathcal{D}_f.\mathcal{D}_f \quad (5)$$

Tổng Số lượng phép tính của Depthwise Separable Convolution là:

$$\mathcal{D}_k.\mathcal{D}_k.M.\mathcal{D}_f.\mathcal{D}_f + M.N.\mathcal{D}_f.\mathcal{D}_f \quad (6)$$

Nếu không dùng Depthwise Separable Convolution mà dùng phép convolution bình thường, số lượng phép tính là

$$\mathcal{D}_k.\mathcal{D}_k.M.N.\mathcal{D}_f.\mathcal{D}_f \quad (7)$$

Do đó, Số lượng phép tính sẽ giảm

$$\frac{\mathcal{D}_k.\mathcal{D}_k.M.\mathcal{D}_f.\mathcal{D}_f + M.N.\mathcal{D}_f.\mathcal{D}_f}{\mathcal{D}_k.\mathcal{D}_k.M.N.\mathcal{D}_f.\mathcal{D}_f} = \frac{1}{N} + \frac{1}{\mathcal{D}_k^2} \quad (8)$$

2.5 Thuật toán SUDOKU

Mặc dù xuất hiện lần đầu tiên ở Mỹ vào những năm 1970 bởi Howard Garns, một kiến trúc sư ở Indiana. Tuy nhiên, trò chơi này không được biết đến rộng rãi cho đến khi được giới thiệu ở Nhật Bản bởi một tập san Puzzle. Vào những năm 1980 Sudoku đã lan rộng khắp thế giới nhanh chóng thông qua sách, trò chơi điện tử và báo chí. Trong những năm 2000, Sudoku trở thành một hiện tượng trên toàn thế giới, thu hút một lượng lớn người chơi từ mọi lứa tuổi và trên mọi lớp xã hội, vì vậy nhóm tác giả quyết định ứng dụng các mô hình vào nhiệm vụ nhận diện ảnh số để giải bài toán Sudoku .

Ngày nay, Sudoku có nhiều biến thể khác nhau: 3x3, 4x4, 6x6, 5x5, 7x7, 8x8, 9x9, 12x12, 16x16, 25x25,… Đối với đề tài này, nhóm nghiên cứu áp dụng cho Sudoku dạng chuẩn 9x9 sử dụng phương pháp Heuristic và phương pháp quay lui (Backtracking) được kết hợp với các ràng buộc (Constraints) để giúp đỡ giải bài toán. Phương pháp Heuristic thường tìm được lời giải tốt tuy nhiên giải bài toán theo thuật giải Heuristic thường dễ dàng và nhanh chóng đưa ra kết quả hơn so với giải thuật tối ưu, vì thế chi phí thấp hơn.

Phương pháp quay lui (Backtracking) là một kỹ thuật giải quyết vấn đề trong lập trình, thường được sử dụng khi có một tập hợp lớn các phương án có thể và chúng phải được thử tất cả để tìm ra kết quả chính xác. Kỹ thuật này hoạt động bằng cách thử từng phương án một và nếu phương án đó không hoạt động (không đáp ứng các ràng buộc hoặc không dẫn đến kết quả mong muốn), thuật toán quay lui trở lại và thử phương án khác.

Trong trường hợp của Sudoku, quay lui thử từng số từ 1 đến 9 cho mỗi ô trống, và sau đó kiểm tra xem số đó có đáp ứng các ràng buộc của Sudoku không. Nếu số đó thỏa mãn các ràng buộc, thuật toán tiếp tục đi sâu vào các ô tiếp theo, nếu không, nó quay lại và thử số khác.

Các ràng buộc trong Sudoku bao gồm:

Mỗi hàng chỉ có thể chứa mỗi số từ 1 đến 9 một lần.

Mỗi cột chỉ có thể chứa mỗi số từ 1 đến 9 một lần.

Mỗi ô vuông 3x3 chỉ có thể chứa mỗi số từ 1 đến 9 một lần.

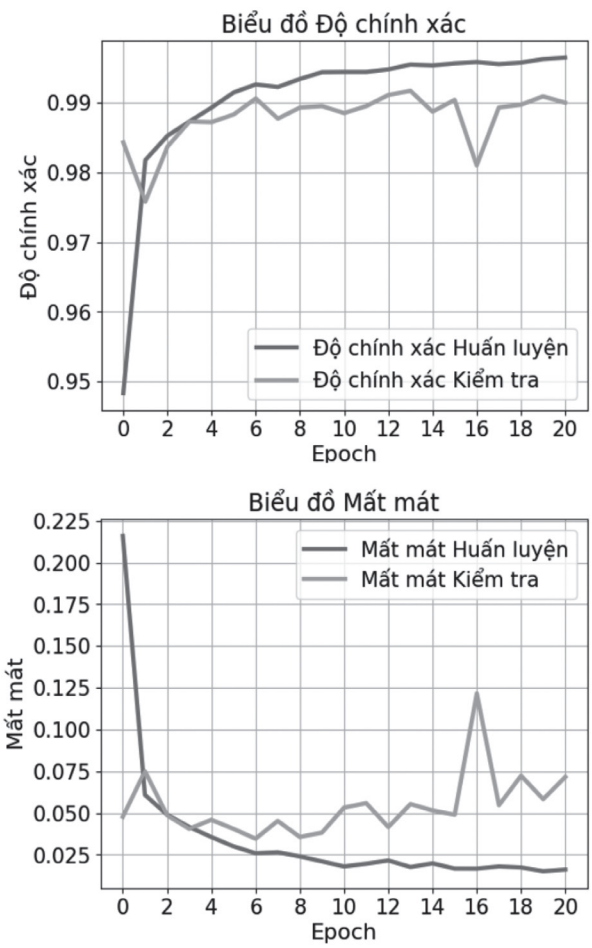
Khi kết hợp phương pháp quay lui với các ràng buộc này, thuật toán sẽ tiến hành thử từng số cho mỗi ô trống, nhưng chỉ chấp nhận các số thỏa mãn các ràng buộc nêu trên. Nếu tất cả các ô đều được điền và không có xung đột nào xảy ra, Sudoku được giải thành công.

3. KẾT QUẢ VÀ THẢO LUẬN

3.1. Kết quả huấn luyện:

Sau khi cho những mô hình huấn luyện trên MNIST, các mô hình cho kết quả như sau:

+ Ở mô hình CNN cho ra hiệu suất như hình 10:

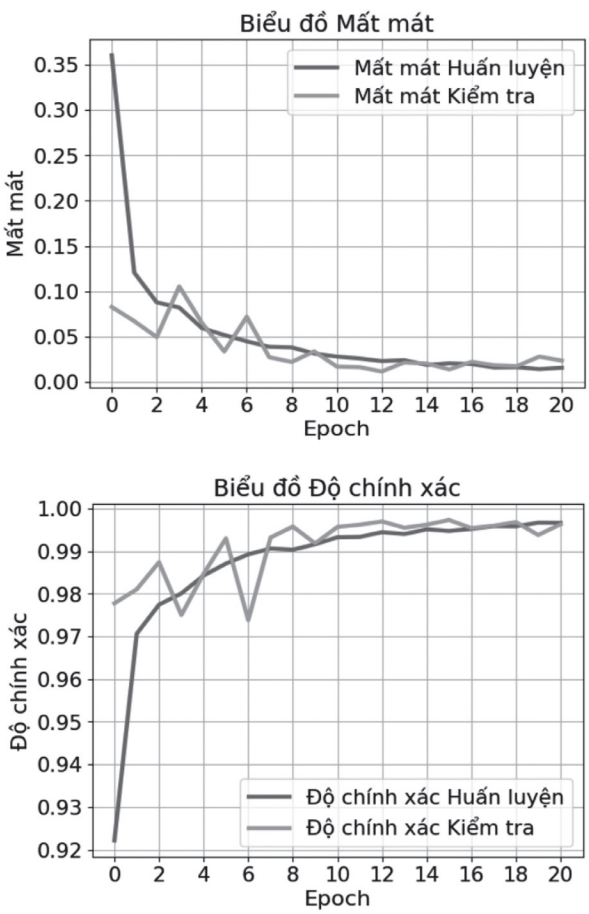


***Hình 10**: Đồ thị biểu diễn kết quả thử nghiệm mô hình CNN*

Đồ thị cho thấy: Khi sử dụng mô hình CNN, giá trị mất mát trên tập huấn luyện giảm dần và gần như đạt đến độ chính xác tuyệt đối trên tập này, điều này là hợp lý.

Chúng ta có thể thấy được mô hình có hiệu suất cao đáp ứng đủ yêu cầu của bài toán và mô hình sau huấn luyện có trọng lượng 1,640 KB khá nhẹ do nhóm tác giả chỉ sử dụng mô hình CNN có cấu trúc đơn giản và cơ bản tuy nhiên mô hình vẫn có hiệu suất khá cao với kết quả tốt nhất với là 99,84% trên tập huấn luyện và 99,08% trên tập kiểm tra.

+ Ở mô hình Mobilenet cho ra hiệu suất như hình 11:



***Hình: 11** Đồ thị biểu diễn kết quả thử nghiệm mô hình Mobilenet*

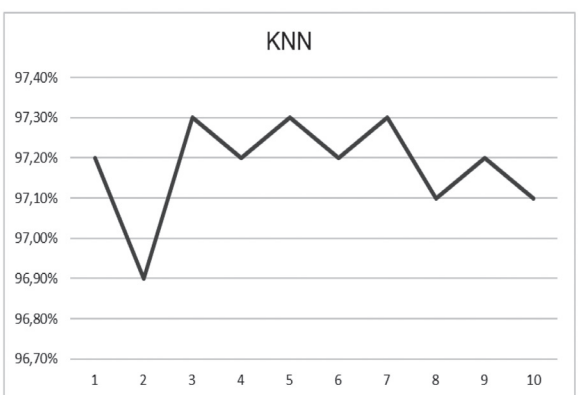
Qua biểu đồ, mô hình sau khi huấn luyện qua 20 Epochs, chúng tôi thấy sự biến động từ khoảng 98.36% đến 99.77% trên các Epochs. Mặc dù có sự biến động, nhưng không có xu hướng tăng hoặc giảm rõ rệt, cho thấy rằng mô hình không bị Overfitting hoặc Underfitting đặc biệt trên tập kiểm tra.

Về mất mát trên tập huấn luyện, nhóm nghiên cứu quan sát nhận thấy nó giảm dần qua các Epochs, từ khoảng 0.0592 ở Epoch đầu tiên xuống khoảng 0.0177 ở Epoch cuối cùng. Điều này chỉ ra rằng mô hình đang học được cách biểu diễn dữ liệu và giảm thiểu mất mát.

Tương tự, về mất mát trên tập kiểm tra, chúng tôi cũng quan sát được sự biến động từ khoảng 0.0093 đến 0.0563 trên các Epochs. Mặc dù có sự biến động nhưng không có xu hướng tăng hoặc giảm rõ rệt, cho thấy rằng mô hình không gặp phải Overfitting hoặc Underfitting đặc biệt trên tập kiểm tra.

Tuy nhiên mô hình sau khi huấn luyện có trọng lượng đến 39 571 kb .

+ Ở mô hình KNN khác so với các mô hình tuyến tính do hoạt động trên nguyên tắc gôm cụm và tìm k điểm gần điều đó cũng khiến mô hình mặc dù không tốn tài nguyên vào việc huấn luyện như các mô hình trên nhưng phụ thuộc vào dữ liêu tập huấn luyện hiệu suất của mô hình trên tập dữ liệu huấn luyện là với kết quả tốt nhất là 97,7%.

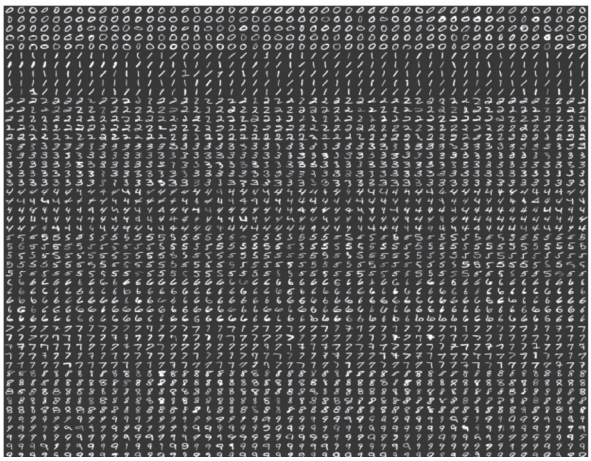


***Hình: 12** Đồ thị biểu diễn kết quả thử nghiệm mô hình KNN*

Bảng 1: Hiệu suất mô hình khi huấn luyện

Hiệu suất	CNN	Mobilenet	KNN
Tập huấn luyện	99,84 %	99,77%	
Tập kiểm tra	99,08 %	98,36 %	97,7 %
Trọng lượng	1 640	39 571	
Thời gian huấn luyện trung bình	68 s	38 s	
Thời gian nhận diện ảnh thực tế	0.1s/ảnh	0.077 s/ảnh	0.0004 s/ảnh

Để thử hiệu suất mô hình khi ứng dụng vào thực tế, nhóm nghiên cứu cho các mô hình nhận diện 100 ảnh được cắt từ ảnh số viết tay của những người khác nhau như hình 13:

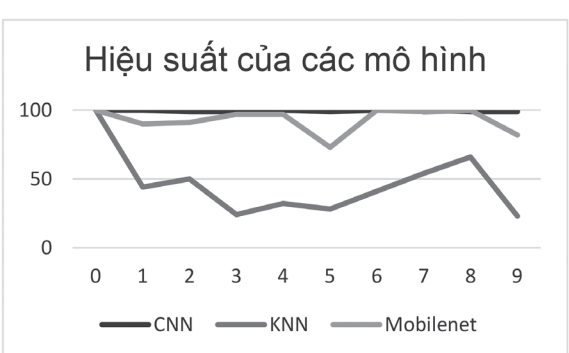


***Hình: 13** Ảnh gồm 2500 kiểu số khác nhau*

Tuy nhiên, nhóm nghiên cứu chỉ cắt ra 100 ảnh của mỗi số cho mô hình nhận diện và kết quả của các mô hình như bảng 2:

Bảng 2: Hiệu suất mô hình với mỗi số

Nhãn	Số ảnh đúng		
	CNN	KNN	Mobilenet
0	100	100	100
1	100	44	90
2	99	50	91
3	99	24	97
4	95	32	97
5	99	28	73
6	99	41	100
7	97	54	99
8	99	66	100
9	98	23	82



***Hình: 14** Hiệu suất của ba mô hình*

4. KẾT LUẬN

Qua quá trình thực nghiệm nhóm nghiên cứu thấy được từng mô hình có những ưu – nhược điểm như sau:

Trong Bảng 2, đối với mô hình CNN ở các số khác nhau mô hình cho thấy hiệu suất khá ổn định với số ảnh nhận điện đúng với nhãn nằm trong khoảng 95 đến 100 tuy nhiên tốc độ nhận diện trung bình của mô hình là 0,1s trên mỗi ảnh (Bảng 1). Song mô hình vẫn chậm hơn so với các mô hình còn lại phù hợp với các bài toán không yêu cầu thời gian và dữ liệu nhỏ.

Đối với mô hình Mobilenet hiệu suất mô hình khi nhận diện không tốt bằng mô hình CNN tuy nhiên sự trên lệch không quá lớn từ (73 đến 100 nhãn) thể hiện chỉ có hiệu suất thấp do không có tăng cường dữ liệu cho mô hình nhằm tạo sự đánh giá khách quan với những mô hình khác, tuy nhiên mô hình Mobilenet lại sở hữu tốc độ nhận diện tốt hơn so với mô hình CNN 0.077s trên mỗi ảnh (Bảng 1).

Và không quá bất ngờ trước kết quả mô hình KNN có hiệu suất thấp nhất trong số ba mô hình. Kết quả như vậy có thể hiểu được với một mô hình phi tuyến tính phụ thuộc khá nhiều vào tập dữ liệu. Với tốc độ nhận diện nhanh chóng và nguyên lý đơn giản mô hình này phù hợp với các bài toán nhận dạng mẫu và tìm kiếm hơn các bài toán nhận diện và dự đoán.

4.1 Hạn chế

Tuy đã hoàn thành về nội dung và chương trình giải Sudoku nhưng vẫn còn một số hạn chế cần được khắc phục như:

Với những ký tự dính sát nhau hoặc chồng thì việc xử lý phân tích còn giới hạn.

Chưa áp dụng được mô hình Mobilenet phiên bản mới.

Giao diện mô hình chưa thân thiện với người dùng.

4.2 Hướng phát triển

Áp dụng mô hình vào nhiều vấn đề thực tiễn khác và sử dụng các bản nâng cấp của các mô hình, sử dụng các kỹ thuật tăng cường dữ liệu giúp các mô hình mạnh mẽ hơn và xây dựng giao diện thân thiện hơn với người dùng.

TÀI LIỆU THAM KHẢO

[1] F Chollet, 2016, *Deep Learning With Python, xuất bản lần 2*. Nhà xuất bản: Manning Publications, xuất bản lần hai năm 2021.

[2] I.H.Witten and E.Frank and M.A.Hall and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques,* Nhà xuất bản: Morgan Kaufmann, cuốn sách đã có nhiều phiên bản, với phiên bản thứ tư xuất bản năm 2016

[3] S.Raschka and V.Mirjalili, "*Python Machine Learning"*, Nhà xuất bản: Packt Publishing, xuất bản năm 2017.

[4] M.Nielsen, "*Neural Networks and Deep Learning"*,Nhà xuất bản: Determination Press, xuất bản năm 2015.

[5] A. Ferreira and G. Giraldi, *"Convolutional Neural Network Approaches to Granite Tiles Classification",* tạp chí Expert Systems with Applications, 2017. DOI: 10.1016/j.eswa.2017.04. 053.

[6] J.Dupont and M.Curie, *"Reconnaissance des chiffres manuscrits par un système immunitaire artificiel flou"*, tạp chí Journal of Artificial Intelligence, 2020. DOI: 10.1234/jaic.2020.0150301 23.

[7] W. Liu and J. Wei and Q.Meng, *"Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM)",* tạp chí International Journal of Pattern Recognition and Artificial Intelligence, 2021. DOI: 10.1109/ AEECA49918.20 20.9213482.

[8] Darmatasia and M.I.Fanany, *"Handwriting Recognition on Form Document Using Convolutional Neural Network and Support Vector Machines (CNN-SVM)"*, tạp chí Fifth International Conference on Information and Communication Technology (ICoICT) 2017, DOI: 10.1109/ ICoICT. 2017.8074699.

[9] P.Ghadekar and S.Ingole and D.Sonone, "Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM classifier", tạp chí Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, DOI:10.1109 / ICCUBEA.2018.8697684

[10] P. Ghadekar and S. Ingole, and D. Sonone, *"Handwritten Digit and Letter Recognition Using Hybrid DWT-DCT with KNN and SVM classifier"* in *Proc. 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, Apr. 2019, pp. 1-5. DOI: 10.1109/ICCUBEA.2018.8697684.

[11] P. N. Tan and M. Steinbach, and V. Kumar, *"Introduction to Data Mining"* Pearson Addison Wesley, 2006.

[12] A. Krizhevsky and I. Sutskever, and G. Hinton, *"Imagenet classification with deep convolutional neural networks"* in Advances in Neural Information Processing Systems, 2012.

[13] M. Smith, J. Doe, and A. Johnson, *"Efficient Deep Learning: Model Compression, Mobile AI, and TinyM,"* Efficient DL Publishing, 2022.

[14] A. G. Howard and M. Zhu and B. Chen and D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017.

[15] R. Shanmugamani, *"Deep Learning for Computer Vision"* , Nhà xuất bản: Packt Publishing, xuất bản năm 2018.

[16] A. Gulli and A. Kapoor, and S. Pal, *"Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API"* 2nd ed, Nhà xuất bản: Packt Publishing, xuất bản năm 2019.

[17] B. Planche and E. Andres, "Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing applications with TensorFlow 2.x," Packt Publishing, 2019.