IMPLEMENTATION OF DEEP LEARNING NEURAL NETWORK LENET5 ON STM32 MICROCONTROLLER FOR IMAGE RECOGNITION

Huynh Viet Thang
University of Science and Technology – University of Danang

ARTICLE INFO

Received: 16/5/2021
Revised: 02/8/2021
Published: 09/8/2021

KEYWORDS

Deep learning
Edge computing
STM32 microcontroller
MNIST
Internet of Things

ABSTRACT

The advent of smart mobile devices, along with the explosion of Internet-based applications and services, has led to the birth of a new computing paradigm – edge computing. Along with the current expanding trend of artificial intelligence applications, deploying artificial intelligence and deep learning applications on edge computing platforms is a prominent trend. This paper will investigate the ability to execute deep learning models using the convolutional neural network LeNet5 for deep learning problems implemented on low-power microcontrollers based on ARM architecture. We present the process of designing and implementing the handwritten digit recognition problem on the STM32 development board. We use Google Colab and Python language to train the convolutional neural network model, then map the trained model to execute on the STM32F411 microcontroller development board with the use of X-Cube-AI tool. The experimental results show that the implementation on the microcontroller achieves nearly the same performance as that on the general purpose computers.

NGHIÊN CỨU TRIỂN KHAI MẠNG HỌC SÂU LENET5 TRÊN VI ĐIỀU KHIỂN STM32 ỨNG DỤNG TRONG NHẬN DẠNG HÌNH ẢNH

Huỳnh Việt Thắng
Trường Đại học Bách Khoa – ĐH Đà Nẵng

THÔNG TIN BÀI BÁO

Ngày nhận bài: 16/5/2021
Ngày hoàn thiện: 02/8/2021
Ngày đăng: 09/8/2021

TỪ KHÓA

Học sâu
Điện toán biên
Vi điều khiển STM32
MNIST
Internet vạn vật

TÓM TẮT

Sự ra đời của các thiết bị di động thông minh, cùng với sự bùng nổ của các ứng dụng và dịch vụ trên nền tảng Internet dẫn đến sự ra đời của mô hình tính toán mới – điện toán biên. Cùng với xu hướng ứng dụng trí tuệ nhân tạo đang rộng mở hiện nay, triển khai các ứng dụng trí tuệ nhân tạo và học sâu trên nền tảng điện toán biên là một xu hướng nổi bật. Bài báo này sẽ khảo sát khả năng thực thi mô hình học sâu sử dụng mạng nơ-ron tích chập LeNet5 cho các bài toán học sâu được triển khai trên các vi điều khiển công suất thấp dựa trên kiến trúc ARM. Chúng tôi trình bày quá trình thiết kế và thực thi bài toán nhận dạng hình ảnh là chữ số viết tay trên board phát triển STM32. Chúng tôi sử dụng Google Colab và ngôn ngữ Python để huấn luyện mô hình mạng nơ-ron tích chập, sau đó ánh xạ mô hình đã huấn luyện lên thực thi trên board phát triển vi điều khiển STM32F411 với công cụ X-Cube-AI. Kết quả đánh giá thực tế trên phần cứng cho thấy việc thực thi trên vi điều khiển đạt hiệu năng gần tương đương với thực thi trên máy tính đa mục đích.

DOI: https://doi.org/10.34238/tnu-jst.4497

Email: thanghv@dut.udn.vn

1. Giới thiệu

1.1. Internet vạn vật và điện toán biên

Ngày nay, sự bùng nổ của các thiết bị di động cùng các cảm biến thông minh đã mở rộng phạm vi và các ứng dụng dựa trên nền tảng Internet. Internet vạn vật (Internet of Things - IoT) đề cập đến mạng lưới các đối tượng và thiết bị được dùng để thu thập và trao đổi dữ liệu, với số lượng các thiết bị IoT này ngày càng gia tăng. Báo cáo của Cisco [1] dự đoán sẽ có khoảng 15 tỷ thiết bị IoT được kết nối vào năm 2023, ước tính chiếm đến 50% tổng số lượng thiết bị kết nối Internet. Cisco cũng ước tính rằng, gần 850 Zettabyte (ZB) dữ liệu sẽ được tạo ra mỗi năm bên ngoài các dịch vụ đám mây, trong khi lưu lượng trung tâm dữ liệu toàn cầu chỉ khoảng 20,6 ZB [2]. Điều này cho thấy các nguồn dữ liệu cho tính toán dữ liệu lớn đang trải qua một quá trình chuyển dịch: chuyển từ các trung tâm dữ liệu đám mây quy mô lớn sang một loạt các thiết bị được đặt ở biên của mạng, còn gọi là các thiết bị biên hay *Edge Device*. Lượng dữ liệu khổng lồ được tạo ra bởi các đối tượng này đã thúc đẩy sự ra đời của các giải pháp mới để lưu trữ và phân tích dữ liệu. Trong bối cảnh đó, khái niệm điện toán biên (hay tính toán biên) – *Edge Computing* – được đề xuất như một mô hình máy tính mới với việc tính toán chủ yếu được thực hiện trên các thiết bị đặt tại biên của mạng, hay là trên các edge device.

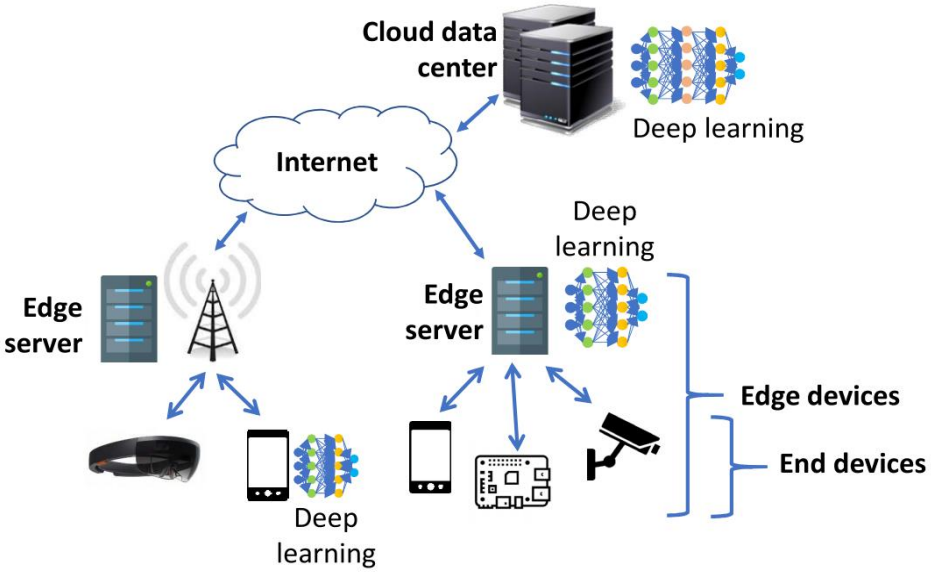
Điện toán biên giúp giải quyết một số thách thức quan trọng liên quan đến các ứng dụng IoT. Ví dụ, trong các ứng dụng và dịch vụ chăm sóc sức khỏe, trong điều khiển xe tự hành, việc phản hồi nhanh là bắt buộc. Với yêu cầu phản hồi nhanh, thay vì gửi dữ liệu lên tính toán ở các máy chủ tại đám mây (cloud), việc tính toán và lưu trữ có thể được thực hiện trực tiếp tại các thiết bị biên đặt ngay gần ứng dụng, do vậy điện toán biên giúp giảm thiểu thời gian truyền dữ liệu và cho phép đáp ứng với độ trễ thấp [2], [3]. Ứng dụng điện toán biên trong những trường hợp như vậy cũng giúp giảm chi phí truyền dữ liệu, tính toán và lưu trữ. Ngoài ra, xử lý dữ liệu ở biên sẽ bảo vệ tính riêng tư của người dùng vì không cần tải dữ liệu lên máy chủ.

Bên cạnh những ưu điểm, việc triển khai mô hình điện toán biên cũng gặp không ít thách thức. Thiết bị biên là một thiết bị điện tử cung cấp kết nối với các nhà cung cấp dịch vụ và các thiết bị cạnh khác; thông thường, các thiết bị như vậy có tài nguyên (tốc độ tính toán, bộ nhớ, các tài nguyên phần cứng khác) hạn chế [3]. Vì các thiết bị biên bị hạn chế về tài nguyên phần cứng, nhiệm vụ chạy các thuật toán, phương pháp và ứng dụng trên các thiết bị biên được coi là một thách thức đáng kể.

1.2. Học sâu trên nền điện toán biên

Là một lớp con của các kỹ thuật học máy, học sâu (Deep Learning – viết tắt DL) là một lĩnh vực nghiên cứu và ứng dụng rất nổi bật hiện nay, đã và đang mang lại những thành quả đáng kể trong nhiều lĩnh vực bao gồm xử lý hình ảnh [4], nhận dạng giọng nói [5] và điều khiển xe tự hành [6]. Với sự ra đời của mô hình điện toán biên, việc triển khai các ứng dụng học sâu kết hợp điện toán biên đang là một xu hướng nổi bật hiện nay. Các camera, micrô và nhiều thiết bị cảm biến đều được đặt ở biên của mạng, cùng với sự ra đời của rất nhiều thiết bị điện toán biên với hiệu năng cao gần đây đã và đang mở ra nhiều cơ hội tuyệt vời để thực thi các thuật toán học sâu ở biên của mạng. Hình 1 minh họa các trường hợp triển khai các ứng dụng học sâu, trong đó mô hình học sâu có thể được triển khai trên các trung tâm dữ liệu ở đám mây (cloud data center), hoặc ở máy chủ biên (edge server), hoặc ở thiết bị điện toán biên (edge device) [3].

Trên thế giới đã có các nghiên cứu mô hình học sâu trên các thiết bị điện toán biên cũng như phát triển các thiết bị điện toán biên cho việc thực thi các thuật toán học sâu [2], [3]. Những giải pháp phần cứng bao gồm: *i)* sử dụng các máy tính nhúng đơn board (single board computer); *ii)* sử dụng các thiết bị tính toán hiệu năng cao như các board phát triển FPGA hay các card đồ họa GPU; *iii)* sử dụng các chip vi điều khiển với công suất thấp và hiệu năng tính toán ở mức chấp nhận được phù hợp với các yêu cầu ứng dụng cụ thể.



Hình 1. *Các mô hình triển khai ứng dụng học sâu* [2]

1.3. Đóng góp khoa học của bài báo

Trong bài báo này, chúng tôi sẽ khảo sát khả năng triển khai thuật toán học sâu trên dòng chip vi điều khiển ARM. Chúng tôi sẽ thực thi bài toán nhận dạng hình ảnh là chữ số viết tay [7] với tập dữ liệu MNIST [8] trên board phát triển STM32F411 [9] của hãng ST. Vi điều khiển STM32F411 (32-bit) có kiến trúc dựa trên lõi ARM Cortex-M và hiện đang được sử dụng phổ biến trong các ứng dụng IoT, y tế, chăm sóc sức khỏe và nhiều ứng dụng khác.

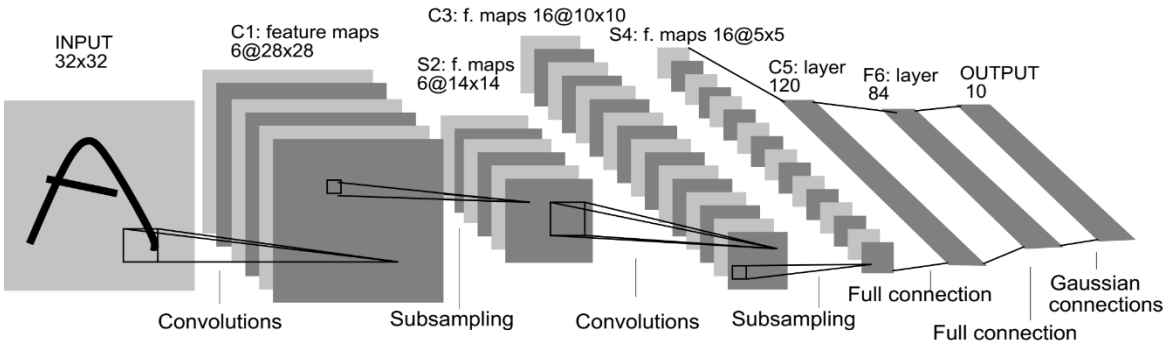
Phần còn lại của bài báo được tổ chức như sau. Phần 2 trình bày mô hình thực hiện hệ thống học sâu trên thiết bị điện toán biên cùng các cơ sở lý thuyết liên quan. Phần 3 trình bày chi tiết quá trình thực hiện bài toán học sâu với tập dữ liệu chữ số viết tay MNIST trên vi điều khiển ARM Cortex-M STM32F411 của hãng ST. Phần 4 là kết quả thực thi và đánh giá mô hình. Cuối cùng, kết luận và những hướng nghiên cứu tiếp theo sẽ được trình bày ở Phần 5.

2. Cơ sở lý thuyết

2.1. Mạng nơ-ron tích chập cho học sâu

Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một loại mạng học sâu thường được sử dụng cho các ứng dụng xử lý hình ảnh và thị giác máy tính. Kiến trúc mạng nơ-ron tích chập thường gồm nhiều lớp (layer), trong đó có ba lớp phổ biến là lớp tích chập (convolution layer), lớp lấy mẫu xuống (subsampling layer) và lớp liên kết hoàn toàn (fully-connected layer). Với kiến trúc như vậy, mạng nơ-ron tích chập thực hiện kết hợp giữa hai chức năng trích chọn thuộc tính và phân loại trong cùng một kiến trúc thực thi, do đó mang lại hiệu suất nhận dạng rất tốt. Hình 2 minh họa một kiến trúc mạng nơ-ron tích chập tiêu biểu – mạng LeNet5 [7] được ứng dụng trong bài toán nhận dạng chữ số viết tay.

Lớp tích chập thực hiện tính tích chập hai chiều (2D convolution) giữa dữ liệu đầu vào và ma trận lọc. Ma trận lọc thường được gọi là kernel và thông thường kernel là một ma trận có kích thước 3x3 hoặc 5x5 phần tử. Sau khi tính tích chập, một hàm kích hoạt phi tuyến được áp dụng cho kết quả chập để tạo ra bản đồ đặc trưng (feature map). Hàm kích hoạt ReLu thường được sử dụng phổ biến trong mạng nơ-ron tích chập. Có nhiều loại kernel được sử dụng trong mỗi lớp của mạng nơ-ron tích chập để tạo ra nhiều bản đồ đặc trưng nhằm trích xuất các loại đặc trưng khác nhau từ dữ liệu đầu vào. Các bản đồ đặc trưng này sau đó được sử dụng để thực hiện phân loại với các lớp mạng nơ-ron liên kết hoàn toàn.



Hình 2. *Kiến trúc mạng nơ-ron tích chập LeNet* [7]

Lớp lấy mẫu xuống thực hiện việc giảm kích thước không gian của các bản đồ đặc trưng từ lớp tích chập trước đó. Việc lấy mẫu xuống nhằm trích xuất các đặc điểm nổi trội từ bản đồ đặc trưng nhằm đảm bảo tính bất biến không gian của các thuộc tính được trích chọn đối với phép toán xoay và dịch chuyển của các điểm ảnh trong tập dữ liệu đầu vào, do đó giúp nâng cao hiệu suất của quá trình huấn luyện và nhận dạng mẫu. Ngoài ra, lớp lấy mẫu xuống cũng hữu ích để giảm độ phức tạp tính toán của mạng nơ-ron tích chập. Có hai toán tử lấy mẫu xuống thường được dùng: lấy mẫu tối đa (max pooling) và lấy mẫu trung bình (average pooling), trong đó lấy mẫu tối đa thường được ưa chuộng hơn. Kích thước của mặt nạ lấy mẫu thông thường là 2x2. Trong một mạng nơ-ron tích chập, các lớp mạng gồm lớp tích chập và lớp lấy mẫu xuống được ghép nối tầng và xen kẽ nhau. Ví dụ, mạng trong Hình 2 có hai lớp tích chập và hai lớp lấy mẫu xuống để thực hiện việc trích xuất đặc trưng cho dữ liệu đầu vào.

Khi việc trích xuất đặc trưng được thực hiện xong, đầu ra của lớp lấy mẫu xuống được làm phẳng thành một vectơ giá trị duy nhất và được đưa vào các lớp nơ-ron kết nối đầy đủ. Các lớp nơ-ron kết nối đầy đủ tiếp đó thực hiện nhiệm vụ phân loại cho hình ảnh đầu vào. Trong ví dụ ở Hình 2, có ba lớp nơ-ron kết nối đầy đủ được sử dụng.

Để thuận tiện trong trình bày, trong bài báo này các lớp trong mạng nơ-ron tích chập được ký hiệu như sau: Lớp tích chập ký hiệu "**Conv**", lớp lấy mẫu tối đa ký hiệu "**MaxPooling**", lớp nơ-ron liên kết hoàn toàn ký hiệu "**Dense**", lớp làm phẳng ký hiệu "**Flatten**".

2.2. Vi điều khiển ARM Cortex-M STM32 và công cụ X-Cube-AI

Dòng vi điều khiển STM32 là một dòng vi điều khiển 32-bit có kiến trúc dựa trên lõi ARM Cortex do hãng STMicroelectronics sản xuất hướng đến các ứng dụng IoT và hệ thống nhúng hiệu năng cao có khả năng cung cấp sự cân bằng tốt nhất giữa mức năng lượng tiêu thụ khi hoạt động và hiệu suất xử lý. Vi điều khiển STM32F411VET6 [7] có dung lượng bộ nhớ chương trình (Flash) là 512 KB và dung lượng bộ nhớ dữ liệu (SRAM) là 128 KB, hoạt động với thạch anh ngoài ở tần số 8 MHz, và thông qua hệ thống phân phối xung clock bên trong chip cho phép lõi vi điều khiển có thể hoạt động ở tần số xung clock tối đa 100 MHz. Trên thị trường, STM32F411VET6 được tích hợp trên board phát triển EDiscovery của STMicroelectronics cho phép người dùng chạy được các ứng dụng điều khiển nhúng thông thường cũng như phát triển các ứng dụng xử lý tín hiệu số như xử lý hình ảnh, âm thanh và video.

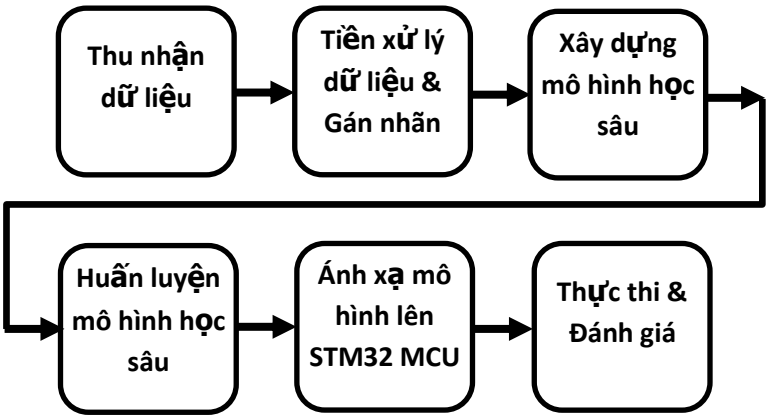
Gần đây, STMicroelectronics đã phát triển các công cụ cho phép thực thi các mô hình mạng nơ-ron nhân tạo trên các dòng chip vi điều khiển STM32. Một trong số đó là gói thư viện mở rộng X-Cube-AI chuyên cho triển khai các mô hình mạng nơ-ron nhân tạo và học máy. Công cụ X-Cube-AI [10] là gói mở rộng của bộ công cụ STM32 CubeMX cho phép người dùng ánh xạ và chạy các mô hình mạng nơ-ron nhân tạo đã được huấn luyện trước (pre-trained) trên các chip STM32 dòng F3 và F4 trở lên. Công cụ X-Cube-AI hỗ trợ tạo thư viện được tối ưu hóa để thực thi các mô hình mạng nơ-ron nhân tạo đã được huấn luyện trước, đồng thời hỗ trợ ánh xạ mô

hình mạng nơ-ron nhân tạo từ nhiều thư viện phát triển mô hình mạng nơ-ron nhân tạo và học sâu phổ biến hiện nay như Keras, TensorFlow Lite, ONNX, PyTorch, Caffe.

3. Thực hiện mô hình học sâu trên vi điều khiển ARM Cortex-M STM32F411

3.1. Các bước thực hiện

Hình 3 minh họa các bước tiêu biểu khi triển khai một ứng dụng nhận dạng và/hoặc học sâu lên vi điều khiển STM32 MCU [10].



Hình 3. *Các bước triển khai một ứng dụng nhận dạng/ học sâu trên vi điều khiển STM32 MCU*

- **Bước 1. Thu nhận dữ liệu**. Trong bước này, các dữ liệu liên quan của bài toán hay ứng dụng cần triển khai sẽ được thu nhận với số lượng mẫu đủ lớn, làm cơ sở để thực hiện việc mô hình hóa hay phân loại dựa trên dữ liệu ở các bước tiếp theo. Để thu nhận dữ liệu có thể sử dụng các cảm biến trên hoặc gần đối tượng được theo dõi và thực hiện việc ghi lại trạng thái và những thay đổi của đối tượng theo thời gian. Các thông số vật lý có thể bao gồm gia tốc, nhiệt độ, âm thanh và hình ảnh tùy thuộc vào ứng dụng.
- **Bước 2**. **Tiền xử lý dữ liệu và Gán nhãn**. Tiền xử lý dữ liệu có thể bao gồm quá trình lọc nhiễu, làm sạch dữ liệu thô. Tiếp đó, dữ liệu được gán nhãn (label) để phục vụ cho việc huấn luyện sử dụng các mô hình học sâu có giám sát ở các bước tiếp theo.
- **Bước 3. Xây dựng mô hình học sâu**. Dựa vào mục tiêu của ứng dụng và tập dữ liệu đã thu nhận được, người thiết kế cần lựa chọn và xây dựng một mô hình học sâu phù hợp để có thể học và suy luận với kết quả tốt nhất từ tập dữ liệu đã có nhằm đáp ứng mục tiêu bài toán. Công việc này có thể bao gồm quyết định kiểu mạng nơ-ron nhân tạo cần dùng (ví dụ mạng nơ-ron lan truyền thẳng hay mạng nơ-ron tích chập), kiến trúc mạng (số lớp, số nơ-ron trong mỗi lớp, hàm kích hoạt cần dùng), phương pháp huấn luyện mạng.
- **Bước 4. Huấn luyện mô hình học sâu**. Quá trình huấn luyện thường được tiến hành trên máy tính hoặc sử dụng các dịch vụ học máy với năng lực xử lý lớn và tài nguyên phần cứng (bộ nhớ, tốc độ) không hạn chế. Kết quả của quá trình huấn luyện là một mô hình mạng nơ-ron nhân tạo đã được huấn luyện (pre-trained model) và được lưu ở định dạng phù hợp với công cụ phát triển của STM32.
- **Bước 5. Ánh xạ mô hình lên vi điều khiển STM32**. Sử dụng công cụ X-Cube-AI để chuyển đổi mô hình mạng nơ-ron đã được huấn luyện trước đó thành mã C tối ưu hóa về độ phức tạp tính toán và yêu cầu bộ nhớ, để thực thi mô hình mạng nơ-ron nhân tạo đã huấn luyện trên vi điều khiển STM32.
- **Bước 6. Thực thi và Đánh giá**. Quá trình thực thi và đánh giá mạng nơ-ron đã thiết kế trên vi điều khiển STM32 có thể được thực hiện dễ dàng bằng cách sử dụng các hàm hoặc tham khảo các ví dụ sẵn có được tạo ra và tích hợp sẵn bởi bộ công cụ X-Cube-AI.

3.2. Huấn luyện mạng nơ-ron tích chập cho nhận dạng chữ số viết tay với MNIST

Trong nghiên cứu này, chúng tôi thực hiện việc ánh xạ mô hình mạng nơ-ron tích chập LeNet thực thi bài toán nhận dạng chữ số viết tay trên tập dữ liệu MNIST lên board phát triển EDiscovery STM32F411VET6. Mô hình mạng nơ-ron tích chập LeNet được huấn luyện trước bằng công cụ Google Colab [11] sử dụng thư viện Keras, sau đó mô hình mạng đã huấn luyện sẽ được ánh xạ, thực thi và đánh giá lên vi điều khiển STM32F411VET6 sử dụng công cụ X-Cube-AI trong môi trường phát triển tích hợp STM32-CubeIDE.

Tập dữ liệu MNIST được sử dụng để huấn luyện và kiểm tra mạng nơ-ron tích chập. Tập dữ liệu MNIST gồm một tập huấn luyện 60.000 mẫu và một tập kiểm tra 10.000 mẫu. Có 10 loại chữ số viết tay khác nhau từ 0 đến 9 trong tập dữ liệu MNIST. Mỗi chữ số được chuẩn hóa và căn giữa trong các bức ảnh đa mức xám có kích thước 28x28 điểm ảnh.

Bảng 1. *Kiến trúc mạng nơ-ron tích chập ứng dụng cho nhận dạng chữ số viết tay với tập MNIST, triển khai trên vi điều khiển STM32F411VET6*

TT	Lớp	Số feature map	Kích thước đầu ra	Kích thước kernel	Hàm kích hoạt	Số lượng tham số
0	Ảnh đầu vào	1	28x28	-	-	-
1	Conv1	6	28x28	5x5	ReLu	156
2	MaxPooling1	6	14x14	2x2	-	-
3	Conv2	16	14x14	5x5	ReLu	2416
4	MaxPooling2	16	7x7	2x2	-	-
5	Flatten	-	784	-	-	-
6	Dense1	-	120	-	ReLu	94200
7	Dense2	-	84	-	ReLu	10164
8	Dense3	-	10	-	Softmax	850

Tổng số tham số: 107786.

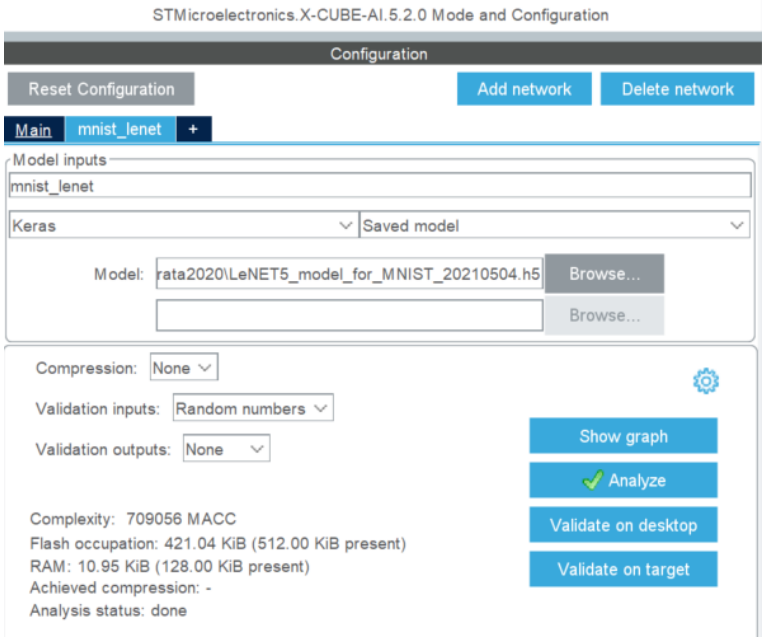
Chúng tôi lựa chọn sử dụng mô hình mạng nơ-ron tích chập LeNet-5, chi tiết mô hình mạng sử dụng được cho trong Bảng 1, với kiến trúc mạng gồm 2 lớp tích chập (Conv1 và Conv2) ghép xen kẽ với 2 lớp lấy mẫu xuống với phương thức lấy mẫu tối đa (MaxPooling1 và MaxPooling2), sau đó được chuyển thành vector nhờ lớp Flatten và đưa đến 3 lớp nơ-ron kết nối đầy đủ (Dense1, Dense2, Dense3) gồm có 120, 84 và 10 nơ-ron tương ứng trong từng lớp để thực hiện việc nhận dạng. Hàm kích hoạt ReLu được sử dụng trong các lớp tích chập và các lớp nơ-ron kết nối đầy đủ, riêng lớp cuối cùng (Dense3) chúng tôi sử dụng hàm kích hoạt Softmax để thực hiện phân loại. Các lớp tích chập sử dụng mặt nạ 5x5, các lớp lấy mẫu sử dụng cửa sổ lấy mẫu kích thước 2x2. Tổng số tham số của mạng là 107786 tham số.

Để huấn luyện mạng, chúng tôi sử dụng Google Colab [11] với chương trình huấn luyện mạng nơ-ron tích chập được phát triển trên nền tảng Python, sử dụng thư viện Keras. Sau 20 vòng huấn luyện (epoch), mạng cho độ chính xác (Accurcy) là 99,77% đối với tập dữ liệu huấn luyện và 99,05% đối với tập dữ liệu kiểm tra. Các kết quả về độ chính xác này rất tốt và có thể so sánh với độ chính xác của các mô hình học máy liên quan [7], [8], [12] cho các ứng dụng nhận dạng chữ số viết tay với cùng tập dữ liệu MNIST. Tất cả các tham số của mạng nơ-ron tích chập đã huấn luyện được lưu lại ở định dạng .h5 phù hợp để sử dụng trong việc ánh xạ mô hình này lên vi điều khiển STM32F411VET6.

3.3. Ánh xạ mô hình mạng nơ-ron tích chập trên vi điều khiển STM32F411VET6

Hình 4 minh họa việc cài đặt các thông số đầu vào sử dụng bộ công cụ X-Cube-AI (phiên bản 5.2.0) trong môi trường phát triển tích hợp STM32-CubeIDE (phiên bản 1.5.0) để thực hiện việc ánh xạ mạng nơ-ron tích chập đã huấn luyện lên vi điều khiển STM32. Để tối ưu hóa dung lượng bộ nhớ sử dụng, X-Cube-AI hỗ trợ nén dữ liệu với các hệ số nén là 4 và 8. Chúng tôi không sử

dụng chế độ nén dữ liệu, vì vậy dung lượng bộ nhớ sử dụng tương ứng cho mô hình mạng nơ-ron đã huấn luyện là: 421,04 KB (82,23%) cho bộ nhớ chương trình (Flash) và 10,95 KB (8,55%) cho bộ nhớ dữ liệu (RAM) (xem Hình 4).



Hình 4. *Thiết lập thông số cho mô hình mạng nơ-ron tích chập sử dụng công cụ X-Cube-AI trong môi trường phát triển tích hợp STM32-CubeIDE*

Bảng 2. *Tài nguyên bộ nhớ sử dụng khi lựa chọn các chế độ nén dữ liệu khác nhau*

Chế độ	Bộ nhớ Flash (KB)	Bộ nhớ RAM (KB)	Hệ số nén thực tế đạt được
Không nén dữ liệu	421,04	10,95	1
Nén với hệ số 4	117,88	10,95	3,64
Nén với hệ số 8	65,15	10,95	6,48

Bảng 3. *Phân tích chi tiết độ phức tạp tính toán và tài nguyên sử dụng của các lớp trong mạng*

Lớp	Số lượng tham số	Tỉ lệ MACC thực thi (%)	Tỉ lệ sử dụng bộ nhớ Flash (%)
Conv1	156	17,9	0,1
Conv2	2416	67,2	2,2
Dense1	94200	13,3	87,5
Dense2	10164	1,4	9,4
Dense3	850	0,1	0,8

Tổng số phép toán nhân cộng tích lũy MACC: ***709056 MACC****. Dung lượng Flash sử dụng 421,04 KB. Dung lượng bộ nhớ RAM sử dụng 10,95 KB.*

Ngoài ra, chúng tôi cũng thử khảo sát sự thay đổi của dung lượng bộ nhớ yêu cầu cho mô hình mạng nơ-ron tích chập đã huấn luyện để ánh xạ lên vi điều khiển STM32F411 khi thay đổi các hệ số nén dữ liệu là 4 và 8, với kết quả dung lượng bộ nhớ sử dụng tương ứng cho các chế độ này được tóm tắt ở Bảng 2. Việc nén dữ liệu ở một khía cạnh sẽ giúp tiết kiệm bộ nhớ, ở khía cạnh khác sẽ làm suy giảm độ chính xác khi thực hiện nhận dạng của mạng, vì vậy người thiết kế cần cân nhắc trong từng trường hợp sử dụng cụ thể. Trong nghiên cứu này, chúng tôi không thực hiện chế độ nén dữ liệu khi thực thi trên board phát triển STM32F411.

Bảng 3 (được tạo ra bởi phần mềm X-Cube-AI) phân tích chi tiết độ phức tạp tính toán và tài nguyên bộ nhớ Flash đã sử dụng tương ứng cho mỗi lớp trong mạng đã triển khai. Tổng số phép toán nhân cộng tích lũy MACC (Multiply-ACCumulate operation) được thực hiện trong mạng là

709056 MACC, trong đó lớp Conv2 có số lượng MACC thực thi lớn nhất chiếm tỉ lệ 67,2%, tiếp theo là lớp Conv1 (tỉ lệ 17,9%) và lớp Dense1 (tỉ lệ 13,3%). Nếu xét đến tài nguyên bộ nhớ sử dụng thì lớp Dense1 có số lượng tham số nhiều nhất (94200 tham số), do vậy đã sử dụng hầu hết bộ nhớ của cả kiến trúc mạng (tỉ lệ 87,5%). Kết quả phân tích chi tiết độ phức tạp tính toán và tài nguyên sử dụng của mạng giúp người thiết kế có thể hiểu rõ hơn đặc điểm hoạt động của mạng và có những điều chỉnh phù hợp để cải thiện hiệu năng của mạng phù hợp mỗi ứng dụng cụ thể.

4. Thực thi và đánh giá

Trong phần này, chúng tôi trình bày kết quả thực thi bài toán nhận dạng chữ số viết tay sử dụng mạng nơ-ron tích chập trên board vi điều khiển EDiscovery STM32F411VET6. Quá trình thực thi được thiết lập chi tiết như sau: Chúng tôi phát triển chương trình C thực hiện việc nhận dạng chữ số viết tay sử dụng mô hình mạng nơ-ron đã được thiết lập trên vi điều khiển STM32F411VET6. Dữ liệu đầu vào để nhận dạng được trích xuất từ tập dữ liệu kiểm tra của tập dữ liệu MNIST. Để đo chính xác thời gian nhận dạng cho mỗi mẫu dữ liệu đầu vào của mạng, chúng tôi sử dụng bộ định thời TIM2 của vi điều khiển STM32F411VET6. Kết quả nhận dạng trên vi điều khiển, bao gồm chữ số được nhận dạng và thời gian nhận dạng cho mỗi mẫu của mạng nơ-ron, được chuyển đến hiển thị trên máy tính thông qua giao tiếp UART. Trên máy tính chúng tôi dùng phần mềm Hercules Terminal kết nối với vi điều khiển và hiển thị kết quả nhận dạng chữ số viết tay trên vi điều khiển.

Chúng tôi cũng so sánh kết quả thực thi mạng nơ-ron trên vi điều khiển STM32F411VET6 với kết quả thực thi cùng một mô hình mạng nơ-ron với cùng tập dữ liệu MNIST trên máy tính đa mục đích có cấu hình CPU Intel Core i5 7200U hoạt động ở tần số 2,5 GHz. Phiên bản mạng nơ-ron trên máy tính được phát triển trên phần mềm Matlab.

Để so sánh, đánh giá năng lực của hệ thống nhận dạng chữ số viết tay sử dụng mạng nơ-ron tích chập chạy trên vi điều khiển STM32F411VET6, chúng tôi sử dụng các thông số là: thời gian nhận dạng và hiệu năng của mạng. Hiệu năng của mạng được tính bằng số lượng phép toán nhân cộng tích lũy MACC được thực thi trong mỗi giây (MACC/s), như trình bày trong công thức (1).

$$\text{Hiệu năng} = \text{Số phép toán MACC} / \text{Thời gian nhận dạng} \quad (1)$$

Bảng 4. *Kết quả thực thi và đánh giá hiệu năng vi điều khiển STM32F411VET6 trong bài toán nhận dạng chữ số viết tay với tập dữ liệu MNIST sử dụng mạng nơ-ron tích chập LeNet5*

Nền tảng phần cứng	Tần số (MHz)	Thời gian nhận dạng (ms)	Hiệu năng (MACC/s)
STM32F411VET6	100	70,66	$10,03 \times 10^6$
CPU Intel Core i5 7200U	2500	68,14	$10,41 \times 10^6$

Bảng 4 trình bày chi tiết kết quả thực thi và đánh giá. Vi điều khiển STM32F411VET6 thực thi mạng nơ-ron tích chập LeNet5 cho bài toán nhận dạng chữ số viết tay với thời gian nhanh gần tương đương với phiên bản phần mềm được thực thi trên máy tính (70,66 ms so với 68,14 ms).

Với thời gian nhận dạng của mạng nơ-ron đã đạt được (70,66 ms cho mỗi mẫu), chúng tôi ước lượng được số lượng mẫu chữ số viết tay mà vi điều khiển có thể nhận dạng trong mỗi giây là xấp xỉ 14 mẫu/giây. Tốc độ nhận dạng này có thể đáp ứng được yêu cầu của các ứng dụng nhận dạng mẫu thực tế triển khai trên các thiết bị điện toán biên công suất thấp trong các ứng dụng và hệ thống IoT. Trong tương lai, nếu thực hiện tối ưu hóa việc truyền dữ liệu giữa bộ nhớ và lõi xử lý của bộ vi điều khiển bằng cơ chế truy cập bộ nhớ trực tiếp (Direct Memory Access – DMA) và/hoặc tối ưu hóa kiến trúc mạng nơ-ron tích chập thì có thể cải thiện hơn nữa tốc độ nhận dạng của thiết bị.

5. Kết luận

Bài báo đã trình bày khả năng triển khai các thuật toán và ứng dụng học sâu trên nền tảng điện toán biên với thiết bị biên là vi điều khiển ARM Cortex-M STM32. Chúng tôi đã giới thiệu các

bước thực hiện mô hình học sâu trên vi điều khiển STM32 và đã triển khai thực thi và đánh giá mô hình mạng nơ-ron tích chập LeNet5 cho bài toán nhận dạng chữ số viết tay trên vi điều khiển STM32F411VET6. Kết quả thực thi cho thấy, mô hình mạng nơ-ron tích chập đã triển khai trên vi điều khiển hoàn toàn có thể đáp ứng yêu cầu của các ứng dụng nhận dạng trong thực tế. Chúng tôi tin rằng, kết quả của nghiên cứu này sẽ góp phần mở ra các khả năng thực thi các ứng dụng IoT và học máy với hiệu năng cao và công suất thấp trên các thiết bị điện toán biên.

Lời cảm ơn

Bài báo này được tài trợ bởi Quỹ Khoa học Công nghệ Murata và Trường Đại học Bách Khoa - ĐHĐN với đề tài có mã số T2020-02-04MSF.

TÀI LIỆU THAM KHẢO/ REFERENCES

[1] Cisco, "Cisco Annual Internet Report (2018–2023) White Paper", 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. [Accessed May 10, 2021].

[2] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869-904, 2020.

[3] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE 107*, no. 8, pp. 1655-1674, 2019.

[4] Y. Wei, W. Xia, M. Lin, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan, "HCP: A flexible CNN framework for multi-label image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 9, pp. 1901-1907, 2015.

[5] A. B. Nassif, I. Shahin, I. Attili, M. Azzeh, and K. Shaalan, "Speech recognition using deep neural networks: A systematic review," *IEEE Access*, vol. 7, pp. 19143-19165, 2019.

[6] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, and L. D. Jackel, "End to end learning for self-driving cars," 2016.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

[8] Y. LeCun and C. Cortes, "MNIST database", 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/. [Accessed May 10, 2021].

[9] ST Microelectronics, "STM32F411 Microcontroller – Product Overview and Documentations", 2020. [Online]. Available: https://www.st.com/en/microcontrollers-microprocessors/stm32f411.html. [Accessed May 10, 2021].

[10] STMicroelectronics, "STM32 solutions for Artificial Neural Networks", 2020. [Online]. Available: https://www.st.com/content/st_com/en/ecosystems/stm32-ann.html. [Accessed May 10, 2021].

[11] Google, "Google Colaboratory (Colab) Introduction", 2020. [Online]. Available: https://colab.research.google.com/notebooks/intro.ipynb. [Accessed May 10, 2021].

[12] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with MNIST and EMNIST," *Applied Sciences*, vol. 9, no. 15, 2019, Art. no. 3169.