Transport and Communications Science Journal, Vol 77, Issue 05 (6/2026), 700-714

Transport and Communications Science Journal

APPLYING ARTIFICIAL INTELLIGENCE TO THE SELF-DRIVING ELECTRIC VEHICLE CONTROL SYSTEM AT NHA TRANG UNIVERSITY

Vu Thang Long, Tran Dang Khoi, Nguyen Van Thuan*

Nha Trang University, No 2 Nguyen Dinh Chieu Street, Khanh Hoa, Vietnam

ARTICLE INFO

TYPE: Research Article
Received: 22/10/2025
Revised: 26/3/2026
Accepted: 25/5/2026
Published online: 15/6/2026
https://doi.org/10.47869/tcsj.77.5.7

* *Corresponding author*
Email: thuannv@ntu.edu.vn; Tel: 0399007896

Abstract. The application of Artificial Intelligence (AI) in the development of autonomous vehicles is a highly promising research direction that has garnered widespread attention from the scientific community. In this study, a Convolutional Neural Network (CNN) was employed to address the road detection task for a three-wheeled electric vehicle model, utilizing image data collected from a camera for training. The steering angle determined by the neural network is executed through the control of a stepper motor. Consequently, the trained CNN achieved a high accuracy of up to 94%. Experimental validation demonstrated that the system operated effectively in controlling the self-driving vehicle on the designed test road. However, the research team also recognizes that the system still has limitations and requires further improvements to enhance stability and precision under various operating conditions. In conclusion, the successful integration of the Convolutional Neural Network and the electronic steering control circuit has laid a crucial foundation for subsequent research and development in the field of autonomous vehicles, particularly at Nha Trang University and in Vietnam as a whole.

Keywords: Convolutional neural network, CNN, autonomous vehicle, road recognition, control circuit.

Tạp chí Khoa học Giao thông vận tải

ỨNG DỤNG TRÍ TUỆ NHÂN TẠO VÀO HỆ THỐNG ĐIỀU KHIỂN XE ĐIỆN TỰ LÁI TẠI TRƯỜNG ĐẠI HỌC NHA TRANG

Vũ Thăng Long, Trần Đăng Khôi, Nguyễn Văn Thuần*

Trường Đại học Nha Trang, số 02 đường Nguyễn Đình Chiểu, Khánh Hòa, Việt Nam

THÔNG TIN BÀI BÁO

CHUYÊN MỤC: Công trình khoa học
Ngày nhận bài: 22/10/2025
Ngày nhận bài sửa: 26/3/2026
Ngày chấp nhận đăng: 25/5/2026
Ngày xuất bản Online: 15/06/2026
https://doi.org/10.47869/tcsj.77.5.7

** Tác giả liên hệ*
Email: thuannv@ntu.edu.vn; Tel: 0399007896

Tóm tắt. Việc ứng dụng trí tuệ nhân tạo để phát triển xe tự hành là một hướng nghiên cứu đầy tiềm năng và đang thu hút sự quan tâm rộng rãi của cộng đồng khoa học. Trong nghiên cứu này, mạng nơ-ron tích chập được ứng dụng để giải quyết bài toán nhận dạng đường cho mô hình xe điện ba bánh, sử dụng dữ liệu ảnh đầu vào được thu thập từ camera để huấn luyện. Giá trị góc lái mà mạng nơ-ron quyết định sẽ được thực thi thông qua sự điều khiển của một động cơ bước. Kết quả là mạng nơ-ron tích chập sau quá trình huấn luyện đã đạt được độ chính xác cao, lên tới 94%. Quá trình thực nghiệm đã chứng minh hệ thống hoạt động tốt trong việc điều khiển xe tự lái trên đường thử nghiệm được thiết kế. Tuy nhiên, nhóm nghiên cứu cũng nhận ra rằng hệ thống còn nhiều hạn chế, cần được cải tiến để tăng cường tính ổn định và độ chính xác trong các điều kiện vận hành khác nhau. Tóm lại, việc kết hợp mạng nơ-ron tích chập và mạch điện điều khiển góc lái đã đặt nền tảng quan trọng cho những nghiên cứu và phát triển tiếp theo trong lĩnh vực xe tự hành tại Đại học Nha Trang nói riêng và Việt Nam nói chung.

Từ khóa: mạng nơ-ron tích chập, CNN, xe tự hành, nhận dạng đường, mạch điều khiển.

1. ĐẶT VẤN ĐỀ

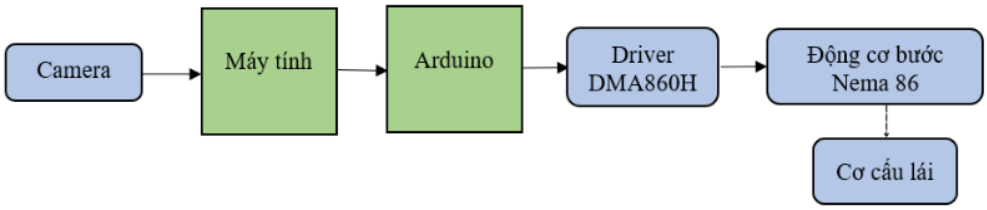
Lịch sử phát triển ngành ô tô đã chứng kiến nhiều sự thay đổi đáng kể để định hình nên một chiếc ô tô hiện đại. Ngày nay ô tô phát triển theo hướng ngày càng thông minh, đặc biệt là khả năng tự hành mà không có sự can thiệp của con người. Các công nghệ như radar, laser, cảm biến LIDAR hoặc siêu âm đã được nghiên cứu và ứng dụng trên ô tô từ sớm. Tuy nhiên, những công nghệ này chỉ hỗ trợ người lái chứ không thể thay thế hoàn toàn con người. Trí tuệ nhân tạo đã tạo ra một cuộc cách mạng trong nghiên cứu về xe tự hành, dựa trên nền tảng trí tuệ nhân tạo các công nghệ như xử lý ảnh, thị giác máy tính, thị giác nhân tạo ngày càng phát triển. Công nghệ này giải quyết bài toán khó mà các công nghệ trước chưa xử lý được như nhận diện biển báo, tín hiệu đèn giao thông, nhận diện làn đường, người đi bộ… Mặc dù có nhiều lợi thế như vậy nhưng phương pháp dựa trên thị giác thiếu tính ổn định khi điều kiện ánh sáng thay đổi [1].

Mạng nơ-ron tích chập (Convolutional Neural Network), được giới thiệu vào những năm 1990 đã trở thành kiến trúc học sâu phổ biến nhờ hiệu quả vượt trội với các bài toán liên quan đến hình ảnh. CNN cũng đã được áp dụng thành công trong các hệ thống nhận dạng vạch kẻ đường, xử lý ngôn ngữ tự nhiên và nhiều lĩnh vực khác. Trước khi CNN được áp dụng rộng rãi, hầu hết các nhiệm vụ nhận dạng mẫu đều được thực hiện thông qua một giai đoạn đầu gồm trích xuất đặc trưng thủ công và sau đó là phân loại. Đột phá của CNN nằm ở chỗ các đặc trưng được học tự động từ các ví dụ huấn luyện. CNN đặc biệt hiệu quả trong các nhiệm vụ nhận dạng hình ảnh vì phép tích chập có thể khai thác bản chất hai chiều của hình ảnh. Ngoài ra, bằng cách sử dụng các bộ lọc tích chập để quét toàn bộ hình ảnh, số lượng tham số cần học tương đối ít so với tổng số phép toán [2].

Tại Việt Nam, việc nghiên cứu chế tạo xe tự hành sử dụng mô hình CNN cũng đang được quan tâm. Nguyễn Việt Bách và Phạm Xuân Tùng đã thực hiện nghiên cứu so sánh các phương pháp phát hiện làn đường, tập trung cụ thể vào kỹ thuật xử lý ảnh truyền thống và mạng nơ-ron tích chập. Kết quả cho thấy mạng nơ-ron tích chập đã thể hiện hiệu suất vượt trội so với các kỹ thuật xử lý ảnh truyền thống về độ chính xác trong phát hiện làn đường. Các mô hình học sâu thường cho thấy độ chính xác và tốc độ xử lý tốt hơn mà không cần đến các bước tiền xử lý phức tạp có thể làm mất nhiều thông tin quan trọng từ hình ảnh [3]. Hà Thị Kim Duyên và các cộng sự đã nghiên cứu và phát triển hệ thống xe tự hành ứng dụng trí tuệ nhân tạo [4]. Trong nghiên cứu này, tác giả đã giải quyết hai nhiệm vụ quan trọng của xe tự hành là nhân dạng vạch kẻ đường và đọc biển báo giao thông. Mô hình CNN được sử dụng cho nhiệm vụ nhận dạng vạch kẻ đường, trong khi thuật toán Adaboost Cascaded cho nhiệm vụ nhận dạng biển báo, hệ thống được lập trình nhúng trên nền tảng phần cứng xử lý hiệu năng cao chuyên dụng cho trí tuệ nhân tạo là TX2 Jetson và hệ điều hành lập trình cho robot ROS. Bên cạnh việc ứng dụng học sâu để xử lý ảnh giúp xác định làn đường. Việc tích hợp camera và LiDAR trong hệ thống xe tự hành có ý nghĩa khoa học quan trọng vì giúp cải thiện độ chính xác trong việc nhận diện các đối tượng trong điều kiện môi trường phức tạp [5].

Bài viết này trình bày kết quả nghiên cứu về hệ thống nhận dạng đường và điều khiển góc lái trên mô hình ô tô ba bánh tại Bộ môn Kỹ thuật ô tô, trường Đại học Nha Trang. Hệ thống phần cứng bao gồm máy tính nhúng NVIDIA Jetson Nano Developer Kit, camera Depth Intel Realsense D435i, vi điều khiển Arduino Uno R3. Ngoài ra còn sử dụng động cơ bước Nema 86 để điều khiển cơ cấu lái. Dữ liệu huấn luyện được thu thập bằng cách lái xe thủ công, đồng thời ghi lại các hình ảnh cùng với góc lái được đồng bộ hóa theo thời gian thể hiện ở Hình 1. Sau đó mạng nơ-ron được huấn luyện ngoại tuyến trên máy tính xách tay trang bị Core-i5, RAM 8GB, ROM 256GB. Cuối cùng, mạng nơ-ron đã được huấn luyện sẽ được sao chép trở lại vào máy

tính nhúng. Kết quả là máy tính nhúng sẽ nhận một khung hình ảnh từ camera và đưa ra giá trị góc lái để điều khiển động cơ bước.



Hình 1. Sơ đồ khối hệ thống điều khiển cơ cấu lái sử dụng xử lý ảnh và Arduino.

2. CÁC THIẾT BỊ PHẦN CỨNG

2.1. Máy tính nhúng NVIDIA Jetson Nano Developer Kit

NVIDIA Jetson Nano là một máy tính bảng đơn hiệu suất cao, chi phí thấp được thiết kế cho các ứng dụng AI và người máy. Được xây dựng dựa trên hệ thống trên chip (SoC) NVIDIA Jetson Nano, tích hợp GPU và CPU mạnh mẽ trong một hệ số dạng nhỏ gọn. Điều này làm cho nó trở thành một nền tảng lý tưởng để phát triển và triển khai các ứng dụng AI, như thị giác máy tính, xử lý ngôn ngữ tự nhiên và người máy. Jetson Nano Developer Kit được trang bị bộ nhớ LPDDR4 4 GB và hỗ trợ nhiều loại giao diện, bao gồm cổng Gigabit Ethernet, USB 3.0 và cổng camera MIPI-CSI. Nó cũng có tiêu đề GPIO 40 chân, cung cấp quyền truy cập vào một loạt thiết bị ngoại vi, chẳng hạn như cảm biến và bộ truyền động.

2.2. Camera Depth Intel Realsense D435i

Intel RealSense D435i là camera độ sâu được thiết kế để sử dụng trong các hệ thống tự trị, chẳng hạn như rô-bốt và xe tự hành, cũng như trong các ứng dụng như thực tế tăng cường và quét 3D. Máy ảnh này được trang bị hai cảm biến hình ảnh, cảm biến độ sâu Stereo và đơn vị đo lường quán tính (IMU), cung cấp khả năng theo dõi chuyển động 6 bậc tự do (6DoF). Một trong những tính năng chính của RealSense D435i là khả năng cung cấp thông tin độ sâu chính xác và đáng tin cậy, ngay cả trong những môi trường đầy thách thức, như điều kiện ánh sáng yếu hoặc có bề mặt phản chiếu. Điều này làm cho máy ảnh rất phù hợp để sử dụng trong các hệ thống tự hành, nơi cần có thông tin độ sâu để điều hướng và tránh chướng ngại vật.

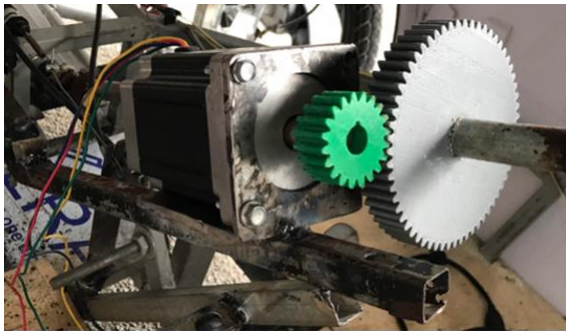
2.3. Arduino UNO R3

Arduino Uno là một vi điều khiển đơn bo mạch dựa trên chip ATmega328. Nó có 14 chân đầu vào/đầu ra kỹ thuật số (trong đó 6 chân có thể sử dụng làm đầu ra PWM), 6 đầu vào analog và một bộ dao động tinh thể 16 MHz. Bo mạch này tích hợp đầy đủ các thành phần cần thiết để hỗ trợ vi điều khiển, cho phép kết nối với nhiều bo mạch mở rộng khác nhau.

3. NGHIÊN CỨU THIẾT KẾ, CHẾ TẠO MẠCH ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC

3.1. Thiết kế, chế tạo phần gá động cơ

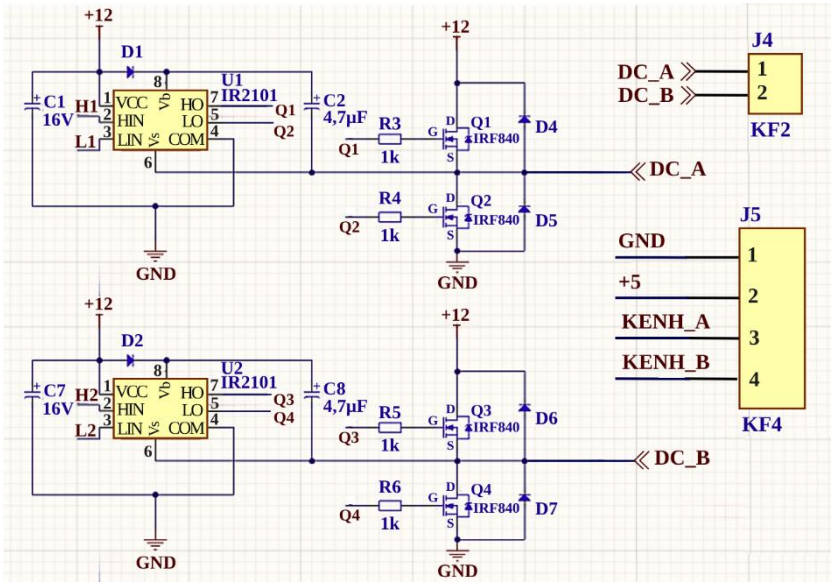
Thiết kế bộ truyền bánh răng có tỉ số truyền 1:4, với bánh răng nhỏ là 15 răng và bánh răng lớn là 60 răng, bước răng 2mm, được thiết kế và in 3D từ nhựa PLA. Sau khi nghiên cứu nhiều loại động cơ, nhóm quyết định chọn động cơ bước Nema 86 với khả năng quay góc chính xác dựa vào xung được cấp. Hệ thống điều khiển cơ cấu lái thể hiện ở Hình 2.



Hình 2. Cơ cấu lái sử dụng động cơ bước Nema 86.

3.2. Thiết kế, chế tạo mạch điều khiển động cơ DC

Động cơ DC được điều khiển bằng một mạch công suất hoạt động dựa vào nguyên lý của mạch cầu H để điều khiển chiều quay và tốc độ. Hình 3 sơ đồ mạch điện điều khiển động cơ DC.



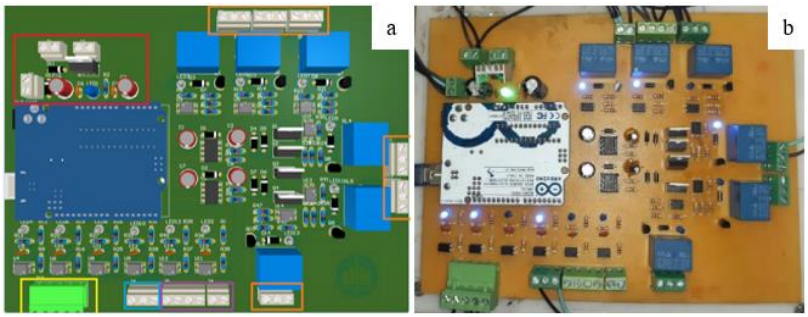
Hình 3. Sơ đồ mạch điều khiển động cơ dùng 2 cầu H.

Dựa vào IC IR2101 chuyên dùng để điều khiển và kích MOSFET một cách độc lập, dễ dàng điều khiển đóng ngắt và đảo chiều động cơ. Thông tin các chân kết nối giữa Arduino và IC IR2101 được trình bày tại Bảng 1.

Bảng 1. Chân kết nối giữa Arduino và IC IR2101.

Arduino	IR2101	MOSFET	Lưu ý
7	H1	Q1	Chỉ được điều khiển kích theo cặp H1(PWM) và L2(Digital) hoặc H2(PWM) và L1 (Digital)
6	L1	Q2	
5	H2	Q3	
4	L2	Q4	

Trong nghiên cứu này nhóm sử dụng bộ chuyển đổi điện áp từ 12V DC xuống 5V để cấp nguồn và đảm bảo an toàn cho Arduino cũng như cấp nguồn cho các thiết bị. Mạch tổng thể điều khiển hệ thống được thiết kế trên phần mềm trước khi chế tạo thực tế. Hình 4 trình bày mạch điều khiển hệ thống dẫn hướng.

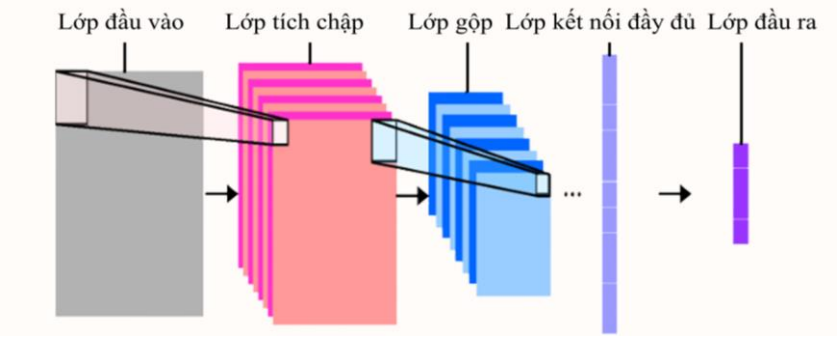


Hình 4. Mạch thiết kế trên phần mềm Altium Designer (a) và mạch điều khiển thực tế (b).

4. NGHIÊN CỨU XÂY DỰNG PHẦN MỀM NHẬN DẠNG ĐƯỜNG

4.1. Kiến trúc mạng nơ-ron tích chập (Convolution Neural Network (CNN))

Nhóm sử dụng mô hình mạng nơ-ron tích chập (CNN) [6] vì đây là một mô hình học sâu rất phổ biến để sử dụng cho các bài toán cần phân loại với độ chính xác cao. Kiến trúc cơ bản của mạng nơ-ron tích chập được minh họa tại Hình 5, bao gồm các lớp tích chập (Convolution layer), lớp lọc (Pooling layer) và cuối cùng là lớp kết nối đầy đủ (Fully connected layer).



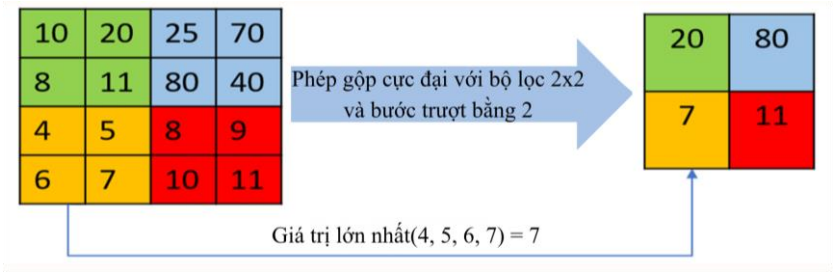
Hình 5. Kiến trúc cơ bản của mạng nơ-ron tích chập [7].

Lớp tích chập (CNN) sẽ thực hiện phép tính tích chập bằng cách nhân giá trị điểm ảnh của ảnh đầu vào với giá trị tương ứng của bộ lọc (kernel), rồi cộng chúng lại và lấy giá trị tuyệt đối của các điểm ảnh để cho tạo thành ảnh đầu ra (output). Quá trình này sẽ lặp lại khi bộ lọc dịch chuyển trình tự từ trái sang phải và từ trên xuống dưới trên ảnh đầu vào (input). Kích thước của ảnh đầu ra (output size) sẽ phụ thuộc và kích thước của bộ lọc (kernel) và bước nhảy (stride) trong quá trình bộ lọc dịch chuyển trên ảnh đầu vào. Việc tính toán áp dụng cho mỗi chiều của ảnh (chiều cao và chiều rộng).

Xác định kích thước ảnh đầu ra (output size) theo công thức (1):

$$Output\ size = \left(\frac{Input\ size - kernel}{Stride}\right) + 1 \quad (1)$$

Lớp lọc Pooling layer nằm giữa các lớp tích chập, sau khi lớp tích chập đã trích xuất các đặc trưng thì pooling layer sẽ làm giảm kích thước ảnh để giảm số lượng tham số tính toán nhưng vẫn giữ các đặc trưng quan trọng [9]. Có hai loại pooling là tính giá trị trung bình (Average pooling) và lấy giá trị cực đại (Max pooling) như được minh họa tại Hình 6. Phép kết hợp cực đại thường được sử dụng để tối ưu hóa các đặc trưng của ảnh, trong nghiên cứu này nhóm sử dụng Max pooling layer. Ngoài ra, pooling layer còn giúp giảm hiện tượng quá khớp [8].



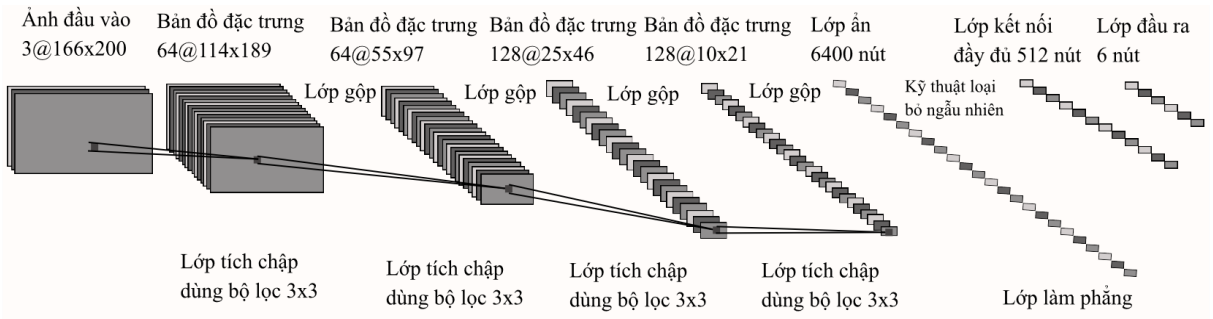
Hình 6. Hình minh họa quá trình Max pooling [10].

Lớp kết nối đầy đủ (fully connected layer) là lớp cuối cùng của một mạng nơ-ron tích chập. Trước khi vào lớp kết nối đầy đủ, mạng nơ-ron tích chập sẽ phải có một lớp làm phẳng (flatten layer). Ma trận từ lớp pooling layer thường là ma trận 2D hoặc 3D, lớp làm phẳng sẽ chuyển đổi ma trận này thành một vector 1D. Vector này sẽ được sử dụng làm đầu vào cho lớp kết nối đầy đủ để thực hiện phân loại hoặc dự đoán. Đặc điểm của lớp này là mỗi nút trong lớp được kết nối trực tiếp với mọi nút ở cả lớp trước và lớp sau. Do đó, đây là lớp có số lượng tham số nhiều nhất trong mạng nơ-ron tích chập và thường mất nhiều thời gian để huấn luyện. Nhược điểm chính của lớp kết nối đầy xuất phát từ số lượng tham số quá lớn, đòi hỏi tính toán phức tạp trong quá trình huấn luyện. Vì vậy, cần phải cố gắng giảm số lượng nút và kết nối. Một giải pháp được sử dụng là kỹ thuật dropout. Đây là kỹ thuật loại bỏ ngẫu nhiên một số nút trong quá trình huấn luyện để giảm quá khớp và tăng khả năng tổng quát hóa.

Hàm kích hoạt (Activation function) đóng vai trò thiết yếu trong các lớp của một mạng nơ-ron tích chập. Trong đó, hàm rectified linear unit (ReLU) được đánh giá là hàm kích hoạt phổ biến nhất trong quá trình trích xuất đặc trưng bằng mạng nơ-ron tích chập. Hàm ReLU mặc dù đơn giản nhưng hiệu quả, loại bỏ các giá trị âm và giữ nguyên các giá trị dương, giúp tăng tốc độ huấn luyện. Về mặt toán học, biểu diễn của hàm ReLU được trình bày như sau:

$$f(x)\,ReLu = max(0,x) \tag{2}$$

Kiến trúc mạng nơ-ron tích chập đề xuất được sử dụng trong nghiên cứu minh họa ở Hình 7 bao gồm 12 lớp, trong đó có 4 lớp convolutional layer, 4 lớp pooling layer, 2 lớp full connected layer và flatten layer để chuyển tensor 3D thành vector 1D. Ngoài ra, để tăng độ ổn định của kiến trúc và tránh hiện tượng quá khớp (overfitting), một lớp dropout được sử dụng với tỷ lệ 0,5. Ảnh đầu vào có kích thước 116×200×3 (cao × rộng × số kênh). Các lớp tích chập được thiết kế để thực hiện việc trích xuất các đặc trưng và được lựa chọn một cách thực nghiệm thông qua nhiều thí nghiệm với các cấu hình khác nhau. Các lớp tích chập sử dụng bộ lọc 3×3 và bước nhảy (stride) là 1×1. Số lượng bộ lọc tương ứng của từng lớp là 64, 64, 128, 128, nhằm nâng cao khả năng học sau của mạng nơ-ron.



Hình 7. Kiến trúc mạng nơ-ron sử dụng trong nghiên cứu.

Sau các lớp tích chập, đầu ra được làm phẳng (flatten) và sau đó đi qua một hai lớp kết nối đầy đủ có kích thước lần lượt là 512 và 3. Tất cả các lớp tích chập và lớp dense đều được trang bị hàm kích hoạt ReLu nhằm cải thiện khả năng hội tụ. Từ các vector đặc trưng, áp dụng softmax để tính toán xác suất góc xoay vô lăng. Toàn bộ mạng có khoảng 3.538.011 tham số.

Số lượng tham số được xác định sau mỗi convolutional layers được xác định như sau:

$$Parameters = \left(f_w \times f_h \times C_{in} + 1 \right) \times C_{out} \quad (3)$$

Trong công thức (3), f_{w} và f_h là kích thước kernel (3×3), số 1 là bias cho mỗi filter, C_{in} là số kênh input và C_{out} số bộ lọc.

Số lượng tham số sau lớp dense được xác định như sau:

$$Parameters = \left(N_{in} + 1 \right) \times N_{out} \quad (4)$$

Trong công thức (4), N_{in} và N_{out} là số node đầu vào và đầu ra, số 1 là bias cho từng node.

Các lớp Maxpooling, Flatten và Dropout không có tham số học được. Kết quả tính toán các tham số ở từng lớp được trình bày trong Bảng 2.

Bảng 2. Bảng tham số của mạng nơ-ron.

Các lớp	**Kích thước đầu ra**	**Số lượng tham số**
Input	(116, 200, 3)	0
Conv1 (3×3, 64)	(114, 198, 64)	1.792
MaxPool1 (2×2)	(57, 99, 64)	0
Conv2 (3×3, 64)	(55, 97, 64)	36.928
MaxPool2 (2×2)	(27, 48, 64)	0
Conv3 (3×3, 128)	(25, 46, 128)	73.856
MaxPool3 (2×2)	(12, 23, 128)	0
Conv4 (3×3, 128)	(10, 21, 128)	147.584
MaxPool4 (2×2)	(5, 10, 128)	0
Flatten	6.400	0
Dropout (0.5)	6.400	0
FC1 (Dense 512)	512	3.277.312
FC2 (Dense 6, softmax)	6	3.78.000
Tổng		**3.539.550**

Trong Bảng 2, số lượng tham số học được tại các lớp MaxPool, Flatten và Dropout đề bằng 0 do bản chất hoạt động của chúng không sử dụng các trọng số hay độ lệch cần tối ưu hóa. Cụ thể, lớp MaxPool hoạt động dựa trên phép toán cố định là trích xuất giá trị lớn nhất nhằm giảm kích thước không gian dữ liệu và gánh nặng tính toán; lớp Flatten chỉ đóng vai trò tái định dạng ma trận đặc trưng đa chiều thành vector một chiều để có thể truyền vào mạng nơ-ron kết nối đầy đủ; còn lớp Dropout là một kỹ thuật điều chuẩn hóa (regularization) thực hiện vô hiệu hóa ngẫu nhiên một tỷ lệ nơ-ron (dựa trên siêu tham số thiết lập sẵn) để ngăn chặn hiện tượng quá khớp (overfitting). Mặc dù không chứa tham số học được, các lớp này vẫn đóng vai trò thiết yếu giúp hệ thống giảm tải tính toán, giữ lại các đặc trưng ảnh quan trọng và tăng độ ổn định, giúp xe tự hành nhận dạng làn đường tốt hơn trong các điều kiện thực tế.

4.2. Thu thập dữ liệu hình ảnh đường thử nghiệm và tiền xử lý ảnh đầu vào

Nhóm sẽ thu hình ảnh đường để sử dụng làm dữ liệu huấn luyện và kiểm tra mô hình nhận dạng. Đường thử nghiệm có nền đen và đường bao màu trắng với làn đường rộng 2,4 m và đường viền rộng 5 cm. Đoạn đường thử nghiệm được thể hiện ở Hình 8. Dữ liệu được thu thập thông qua việc lái mô hình xe điện ba bánh thủ công và sử dụng camera, thể hiện ở Hình 8. Máy tính sẽ ghi lại hình ảnh và thông tin lái xe quanh làn đường với tốc độ 5-6 km/h.

Dữ liệu thu thập chứa hơn 9.000 hình ảnh. Độ phân giải gốc của hình ảnh là 480x640. Máy ảnh được cấu hình để chụp với tốc độ 10 khung hình/giây với thời gian phơi sáng 5000us để ngăn mờ do rung động đàn hồi khi xe chạy trên đường. Tập dữ liệu sẽ được chia thành 3 thư mục chứa hình ảnh bao gồm: Training, Validation và Testing. Mỗi thư mục sẽ chứa 6 thư mục con phân loại hình ảnh chứa góc đánh lái: 0, 30, 90, -30, -90, Stop. Tỉ lệ chứa hình ảnh sẽ là 80% Train và 20% Validation. Hình 9 thể hiện hình ảnh dữ liệu mẫu góc 30^o.

Hình 8. Đoạn đường thử nghiệm (a) và lái xe thủ công để thu thập dữ liệu huấn luyện (b).

Hình 9. Tập dữ liệu mẫu tại góc đánh lái 30^o.

Nhóm nghiên cứu sử dụng một đối tượng ImageDataGenerator thể hiện ở Hình 10 trong quá trình huấn luyện mạng nơ-ron tích chập (CNN) trên tập dữ liệu ảnh. ImageDataGenerator là một công cụ giúp tăng cường dữ liệu bằng cách tạo ra những phiên bản mới của các ảnh trong quá trình huấn luyện. Điều này giúp tăng độ chính xác của mô hình và giảm tình trạng overfitting. Các tham số được cấu hình cho ImageDataGenerator bao gồm, rescale = 1./255 giúp đưa giá trị của pixel ảnh từ [0, 255] về khoảng [0, 1] để tối ưu hóa quá trình huấn luyện, giúp mô hình học nhanh và hội tụ tốt hơn. Tham số rotation_range = 20 thực hiện xoay ảnh một góc ngẫu nhiên trong khoảng từ -20^o đến +20^o độ, giúp mô hình nhận diện tốt hơn khi đối tượng bị xoay. Tham số width_shift_range = 0,1 và height_shift_range= 0,1 là dịch chuyển ảnh theo chiều ngang và chiều dọc một khoảng tối đa 10% kích thước ảnh. Tham số shear_range = 0,2 làm cong ảnh một góc ngẫu nhiên trong khoảng từ -0,2 đến 0,2. Tham số zoom_range = 0,2, ảnh sẽ được phóng to hoặc thu nhỏ ảnh tối đa 20% so với ảnh gốc. Tham số fill_mode='nearest' giúp điền giá trị của pixel bị thiếu bằng giá trị gần nhất khi xoay, dịch chuyển hay zoom làm mất một phần ảnh.

```
TRAINING_DIR = "Dataset1/Training/"
training_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    fill_mode = 'nearest'
)
```

Hình 10. Đoạn mã làm giàu dữ liệu hình ảnh.

4.3. Huấn luyện mô hình

Huấn luyện mô hình theo kiến trúc mạng đã đề xuất trong Hình 10. Quá trình huấn luyện sử dụng thư viện Keras, đây là một thư viện học sâu phổ biến có thể được sử dụng để xây dựng các mô hình dự đoán góc lái trong xe tự hành. Keras giúp việc xác định và đào tạo các mô hình mạng nơ-ron phức tạp cho các tác vụ khác nhau trở nên đơn giản, bao gồm các vấn đề hồi quy như dự đoán góc lái. Hình 11 biểu diễn kiến trúc mạng nơ-ron dùng Keras/TensorFlow.

```
model = tf.keras.models.Sequential([
    # lưu ý hình dạng đầu vào là kích thước mong muốn của hình ảnh 116, 200 với 3 byte màu
    # Đây là tích chập đầu tiên
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(116, 200, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # conv2d 2
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    # conv2d 3
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    # Tconv2d 4
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    # Flatten kết quả
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),

    # 512 neuron hidden layter
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(6, activation='softmax') # 6 giá trị dự đoán
])

# tính toán và bắt đầu huấn luyện
model.summary()
```

Hình 11. Kiến trúc mô hình mạng Nơ-ron tích chập dùng Keras/TensorFlow.

Mô hình được biên dịch với hàm mất mát categorical crossentropy, bộ tối ưu hóa (optimizer) RMSprop và chỉ số đánh giá accuracy. Quá trình huấn luyện được thực hiện trên tập dữ liệu huấn luyện với 25 epoch, mỗi epoch gồm 20 bước lặp, đồng thời đánh giá hiệu suất trên tập kiểm định với 3 bước xác thực. Kết quả huấn luyện (bao gồm giá trị hàm mất mát và độ chính xác theo từng epoch) được lưu trữ bằng thư viện pickle dưới dạng tệp history_model_9.pkl. Sau khi huấn luyện hoàn tất, toàn bộ mô hình, bao gồm kiến trúc và trọng số, được lưu dưới định dạng HDF5 với tên model9.h5, cho phép sử dụng lại trong các bước dự đoán hoặc huấn luyện bổ sung. Hình 12 thể hiện quá trình huấn luyện mô hình.

```
import pickle

model.compile(loss='categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy']) #optimizer = 'adam'

history = model.fit(train_generator, epochs=25, steps_per_epoch=20, validation_data = validation_generator, verbose = 1, validation_steps=3)

# Save the history object to a file
with open('history_model_9.pkl', 'wb') as f:
    pickle.dump(history.history, f)

model.save("model9.h5")
```

Hình 12. Mã nguồn biên dịch và huấn luyện mô hình mạnh nơ-ron.

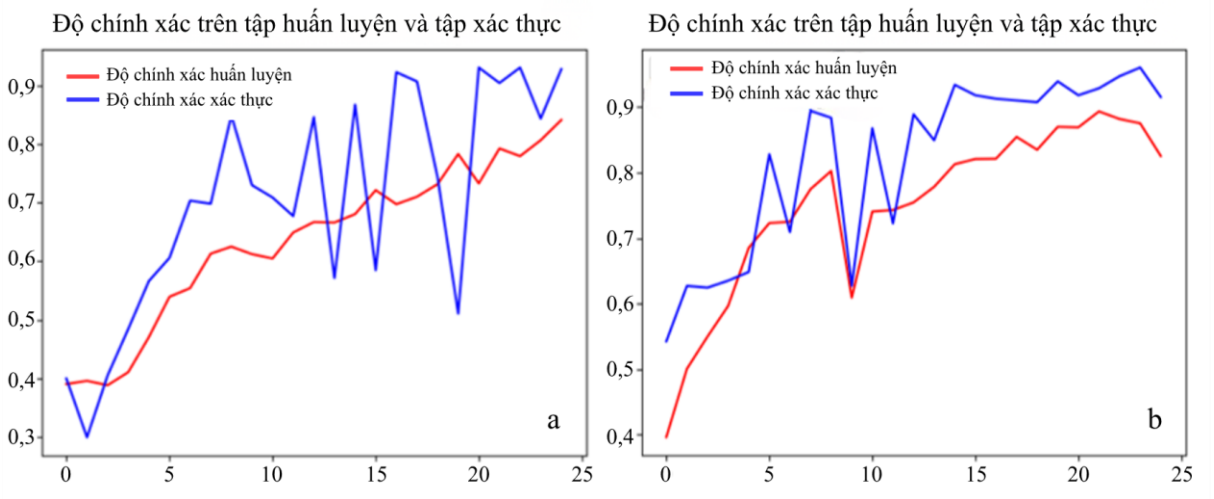
Kết quả từ Epoch 1 đến Epoch 25, phản ánh hai xu hướng tích cực. Giá trị loss và val_loss có xu hướng giảm với giá trị hàm mất mát (loss function) trên tập dữ liệu huấn luyện giảm từ 1,5002 xuống 0,3531, khi loss càng thấp, mô hình càng học tốt và dự đoán chính xác hơn [11]. Tương tự, giá trị hàm mất mát trên tập dữ liệu thực (validation data) cũng giảm dần từ 1,4627 xuống 0,1623, tập dữ liệu này được sử dụng để đánh giá hiệu suất của mô hình trên dữ liệu mà nó chưa từng thấy trong quá trình huấn luyện. Với giá trị val_loss thấp giúp mô hình giảm hiện tượng quá khớp. Trong khi đó, giá trị accuracy và val_accuracy có xu hướng tăng. Giá trị accuracy đánh giá độ chính xác của mô hình trên tập dữ liệu huấn luyện, tức là cho biết tỉ lệ dự đoán đúng của mô hình. Trong quá trình huấn luyện giá trị accuracy tăng từ 0,3115 lên 0,8766. Cuối cùng là giá trị đánh giá độ chính xác của mô hình trên tập dữ liệu xác thực val_accuracy, đây là chỉ số quan trọng để đánh giá khả năng tổng quát hóa của mô hình. Giá trị này tăng từ 0,5106 lên 0,9339. Qua sự thay đổi các giá trị kể trên cho thấy mô hình đang học rất hiệu quả. Cả hai chỉ số trên tập huấn luyện và tập xác thực đều cải thiện liên tục, chứng tỏ mô hình không chỉ học thuộc lòng dữ liệu huấn luyện mà còn có khả năng dự đoán tốt trên dữ liệu mới. Hình 13 thể hiện kết quả quá trình huấn luyện.

```
Epoch 9/25
20/20 [==============================] - 97s 5s/step - loss: 0.6603 - accuracy: 0.7877 - val_loss: 0.3735 - val_accuracy: 0.8677
Epoch 10/25
20/20 [==============================] - 97s 5s/step - loss: 0.5797 - accuracy: 0.8083 - val_loss: 0.4436 - val_accuracy: 0.8360
Epoch 11/25
20/20 [==============================] - 94s 5s/step - loss: 0.5625 - accuracy: 0.8128 - val_loss: 0.3613 - val_accuracy: 0.8651
Epoch 12/25
20/20 [==============================] - 98s 5s/step - loss: 0.5116 - accuracy: 0.8234 - val_loss: 0.3920 - val_accuracy: 0.9048
Epoch 13/25
...
Epoch 24/25
20/20 [==============================] - 95s 5s/step - loss: 0.3542 - accuracy: 0.8770 - val_loss: 0.1888 - val_accuracy: 0.9233
Epoch 25/25
20/20 [==============================] - 95s 5s/step - loss: 0.3531 - accuracy: 0.8766 - val_loss: 0.1623 - val_accuracy: 0.9339
```

Hình 13. Kết quả quá trình huấn luyện sau 25 epoch.

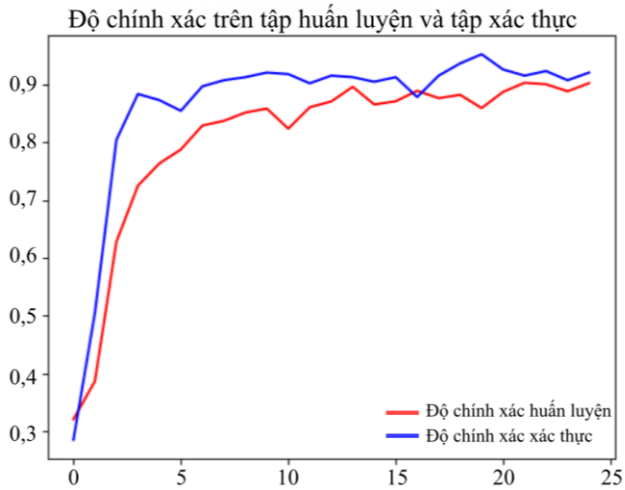
Kết quả quá trình huấn luyện có thể được quan sát trực quan dưới dạng đồ thị. Sau khi mô hình CNN được huấn luyện, toàn bộ lịch sử các chỉ số như accuracy và loss qua từng epoch sẽ được lưu lại. Dựa vào dữ liệu đó để vẽ biểu đồ so sánh training và validation. Nhờ đó có thể trực quan hóa quá trình học của mô hình và đánh giá xem mô hình có học tốt không hay bị quá khớp. Hình 14 thể hiện kết quả raining lần đầu và lần hai với phần trăm sai số lớn giữa dữ liệu huấn luyện và kiểm tra.

Trong kết quả training lần đầu giá trị training accuracy tăng dần và khá ổn định trong khi validation accuracy dao động mạnh, nhưng tổng thể vẫn có xu hướng tăng. Mô hình có học được, tuy nhiên, validation accuracy chưa ổn định. Ở lần training thứ hai validation accuracy ban đầu có dao động nhưng dần ổn định quanh 0,9, hai đường gần nhau, độ chính xác trên tập xác thực cao hơn tập huấn luyện. Đây là dấu hiệu mô hình đang học tốt và không bị overfitting nghiêm trọng, có cải thiện so với lần đầu training.



Hình 14. Kết quả quá trình huấn luyện lần đầu (a) và kết quả quá trình huấn luyện lần hai (b).

Sau hai lần huấn luyện và các lần sau đó, mô hình huấn luyện có tỉ lệ chính xác còn thấp, nhóm quyết định tăng thêm dữ liệu hình ảnh đầu vào với các điều kiện sáng và độ chói khác nhau. Sau khi tăng dữ liệu và chỉnh sửa một số thông số mô hình cũng như lớp tích chập thì cho đến kết quả chỉnh xác nhất. Kết quả quá trình huấn luyện trình bày tại Hình 15.



Hình 15. Đồ thị kết quả quá trình huấn luyện với độ chính xác lên đến 94%.

Đây là kết quả lý tưởng, mô hình học nhanh, đạt độ chính xác cao và không bị overfitting hoặc underfitting. Training và validation accuracy đều tăng và sớm đạt trên 0,9. Hai đường độ chính xác huấn luyện và kiểm tra cho thấy sự tương đồng về xu hướng tổng thể, dao động nhỏ, mô hình ổn định. Mô hình sau khi huấn luyện có thể triển khai thực tế.

4.4. Kết quả và thảo luận

Môi trường thử nghiệm được thiết lập trực tiếp tại tuyến đường nội bộ trong khuôn viên Đại học Nha Trang. Đường thử nghiệm là mặt đường nhựa thực tế, có các vạch kẻ giới hạn làn đường. Để đánh giá toàn diện khả năng tổng quát hóa của mô hình học sâu đề xuất, các thử nghiệm được tiến hành dưới điều kiện ánh sáng tự nhiên, bao gồm cả những đoạn đường có bóng râm của cây cối che khuất bề mặt. Không gian hành động của hệ thống được thiết kế dựa trên 5 kịch bản góc lái cốt lõi, tương ứng với 5 nhãn dữ liệu: đi thẳng (0°), rẽ trái nhẹ (-30°), rẽ

phải nhẹ (30°), rẽ trái gắt (-90°) và rẽ phải gắt (90°) và dừng lại. Các kịch bản này bao quát toàn bộ các tình huống chuyển hướng cơ bản của xe điện 3 bánh trên sa hình thực tế. Kết quả quá trình thử nghiệm thể hiện tại Bảng 3.

Bảng 3. Đánh giá khả năng điều khiển của xe trong các kịch bản thử nghiệm.

Nhãn góc lái	Tình huống trên đường thử nghiệm	Đánh giá trạng thái đáp ứng
0°	Xe di chuyển trên đoạn đường thẳng.	Xe giữ vô lăng ổn định, ít xảy ra hiện tượng rung lắc (overshoot), bám tốt tâm làn đường.
-30° / 30°	Xe tiến vào các đoạn đường cong nhẹ (trái/phải).	Hệ thống nhận diện kịp thời độ cong của vạch kẻ đường, động cơ bước bẻ lái, không bị lấn vạch.
-90° / 90°	Xe gặp các ngã rẽ vuông góc.	Mô hình phát hiện góc rẽ, hệ thống đánh hết lái để chuyển hướng an toàn và sau đó tự động trả lái khi hết cua.
Stop	Xe kết thúc vòng lặp sa hình	Hệ thống ngừng cấp xung cho động cơ di chuyển, xe dừng hẳn và duy trì trạng thái an toàn.

Qua quan sát thực tế ở cả 6 kịch bản, hệ thống điều khiển hoạt động trơn tru và đáp ứng tốt với các thay đổi của sa hình. Hiệu năng này hoàn toàn đồng nhất với kết quả huấn luyện đã phân tích ở Mục 4.3, khi mô hình đạt tỷ lệ dự đoán chính xác tổng thể (overall accuracy) lên tới 93,39% trên tập dữ liệu thực tế. Môi trường triển khai và tính ổn định của phần cứng hệ thống thể hiện ở Bảng 4.

Bảng 4. Đánh giá môi trường triển khai và tính ổn định phần cứng của hệ thống.

Thông số đánh giá	Kết quả thực nghiệm / Môi trường triển khai	Ý nghĩa đối với tính ổn định của xe
Nền tảng phần cứng huấn luyện	Core-i5, RAM 8GB, ROM 256GB	Đảm bảo năng lực tính toán để mô hình hội tụ.
Thời gian huấn luyện	40 phút	Mô hình tinh gọn, thời gian hội tụ nhanh.
Phần cứng suy luận trên xe	Máy tính nhúng NVIDIA Jetson Nano Developer Kit	Nền tảng nhúng thực thi trực tiếp điều khiển.
Tài nguyên phần cứng (CPU/RAM)	Hoạt động ổn định, không gây tràn bộ nhớ (Out of Memory).	Đảm bảo hệ thống không bị treo hoặc sụt nguồn đột ngột.

Bảng 5. So sánh kiến trúc, môi trường thử nghiệm và hiệu năng của mô hình đề xuất với các nghiên cứu liên quan.

Công trình nghiên cứu	Môi trường thử nghiệm / Dữ liệu	Kiến trúc sử dụng	Chỉ số hiệu năng (Loss / Accuracy)
Hoang T. N. và cộng sự. [12]	Mô phỏng 3D Gazebo-ROS2	VGG-19 (Học sâu chuyển giao - Deep Transfer Learning)	Độ chính xác (MPP) = 90,1%

Khan, M. S. và cộng sự. [13]	Dữ liệu tĩnh có sẵn (Sully Chen)	Mạng CNN hạng nhẹ kết hợp Học liên kết (Federated Learning)	Loss (MSE) = 0,0274
Karemore, Y. và cộng sự. [14]	Môi trường mô phỏng lái xe	Mạng NVIDIA CNN	Loss (MSE) = 0,0118
Nghiên cứu của chúng tôi	Thực tế (Đường chạy tại ĐH Nha Trang)	Mạng CNN đề xuất	Loss (val_loss) = 0,1623 Accuracy = 93,39%

So sánh với các nghiên cứu gần đây, mô hình CNN đề xuất thể hiện sự hiệu quả về tính ứng dụng thực tiễn. Các nghiên cứu khác đạt sai số rất thấp (0,0118 - 0,0274) nhờ sử dụng mạng nơ-ron đồ sộ (VGG-19, NVIDIA CNN), nhưng kết quả chỉ dừng ở môi trường giả lập hoặc dữ liệu tĩnh lý tưởng. Mạng CNN của nghiên cứu này được thiết kế tinh gọn, có chủ đích nhằm giảm độ trễ, đảm bảo khả năng xử lý mượt mà trên phần cứng nhúng thực tế của xe điện. Kiến trúc mạng CNN đề xuất được chủ đích thiết kế tinh gọn nhằm giảm thiểu độ trễ tính toán, đảm bảo khả năng suy luận (inference) mượt mà trên vi điều khiển nhúng của xe điện 3 bánh. Quá trình lấy dữ liệu và thử nghiệm được tiến hành trực tiếp trên đường chạy thực tế tại khuôn viên Đại học Nha Trang, nơi có sự biến động phức tạp về lóa sáng, bóng râm và độ rung lắc của camera. Trong điều kiện khắc nghiệt đó, mức sai số kiểm chứng (val_loss) 0,1623 là một kết quả hoàn toàn hợp lý và đáng tin cậy. Đặc biệt, hệ thống vẫn duy trì tỷ lệ nhận dạng bám làn đường thành công (Accuracy) ấn tượng đạt mức 93,39%. Kết quả này minh chứng rằng sự tối ưu hóa tinh gọn về mặt kiến trúc không làm suy giảm độ chính xác, mà ngược lại, đáp ứng xuất sắc yêu cầu điều khiển cơ khí thời gian thực trên một hệ thống xe tự lái thực tế .

5. KẾT LUẬN

Trong bài báo này, nhóm nghiên cứu đã ứng dụng trí tuệ nhân tạo vào việc nhận dạng làn đường và xử lý tín hiệu để điều khiển cho xe hoạt động một cách tự động và ổn định với độ chính xác của quá trình huấn luyện lên tới 94%. Việc lấy dữ liệu hình ảnh đường thực tế để làm dữ liệu huấn luyện là rất quan trọng, ta cần nhiều dữ liệu hình ảnh với các điều kiện ánh sáng khác nhau và nhiều góc lái tương ứng với hình ảnh khác nhau. Cuối cùng, nhóm đã hoàn thành công trình nghiên cứu với mô hình xe điện ba bánh tự hành có khả năng giữ làn đường và tự đánh lái. Hệ thống đã hoạt động khá tốt, nhưng bên cạnh đó vẫn cần thu thập thêm nhiều dữ liệu hình ảnh. Nếu không có tất cả những hình ảnh và góc lái, cùng với sự làm giàu dữ liệu, sẽ không thể xây dựng một mô hình mạnh mẽ. Vì vậy, cần tăng thêm dữ liệu hình ảnh để mô hình có thể tăng độ chính xác hơn nữa. Bên cạnh đó, một vấn đề được phát hiện trong đào tạo và kiểm tra mạng nơ-ron là độ trễ camera, do đó, bên cạnh việc cải tiến và cập nhật phần mềm thì phần cứng cũng cần phải được cải thiện để đáp ứng yêu cầu về độ chính xác và ổn định.

TÀI LIỆU THAM KHẢO

[1]. T.-D. Do, M.-T. Duong, Q.-V. Dang, M.-H. Le, Real-Time Self-Driving Car Navigation Using Deep Neural Network, 2018 4th International Conference on Green Technology and Sustainable Development (GTSD), (2018) 7-12. https://www.researchgate.net/publication/330487831

[2]. J. Yang, G. Yang, Modified Convolutional Neural Network Based on Dropout and the Stochastic Gradient Descent Optimizer, Algorithms, 11 (2018) 28. https://doi.org/10.3390/a11030028

[3]. N. V. Bach, P. X. Tung, Lane detection using Hough transformation and YOLOv8, Transport and Communications Science Journal, 75 (2024) 1659-1672. https://doi.org/10.47869/tcsj.75.4.15

[4]. Hà Thị Kim Duyên, Lê Mạnh Long, Nguyễn Đức Duy, Phan Sỹ Thuần, Nguyễn Ngọc Hải, Nguyễn Thị Tú Uyên, Ngô Mạnh Tiến, Nghiên cứu và phát triển hệ thống xe tự hành ứng dụng trí tuệ nhân tạo, Tạp chí Khoa học và Công nghệ, 57 (2021) 38-43.

[5]. T. T. H. T. Nguyen, T. T. Dao, T. B. Ngo, Real-Time Multi-Sensor Fusion for Object Detection and Localization in Self-Driving Cars: A CARLA Simulation, Transport and Communications Science Journal, 76 (2025) 64-78. https://doi.org/10.47869/tcsj.76.1.6

[6]. S. Albawi, T. A. Mohammed, S. Al-Zawi, Understanding of a Convolutional Neural Network, 2017 International Conference on Engineering and Technology, (2017) 264-269. http://doi:10.1109/ICEngTechnol.2017.8308186

[7]. C. Sharma, S. Bharathiraja, G. Anusooya, Self Driving Car using Deep Learning Technique, International Journal of Engineering Research & Technology, 9 (2020) 248-253. https://doi.org/10.17577/IJERTV9IS060247

[8]. P. Purwono, A. Ma'arif, W. Rahmaniar, H.I.K. Fathurrahman, A.Z.K. Frisky, Q.M. ul Haq, Understanding of Convolutional Neural Network (CNN): A Review, International Journal of Robotics and Control Systems, 2 (2022) 739-748. http://dx.doi.org/10.31763/ijrcs.v2i4.888

[9]. Nguyễn Hữu Quyết, Lê Văn Vũ, Trần Ngọc Hoà, Phân loại hư hỏng khung thép bằng mạng nơ ron tích chập một chiều và cơ chế chú ý kênh, Tạp chí Khoa học Giao thông vận tải, 75 (2024) 2333-2344. https://doi.org/10.47869/tcsj.75.9.8

[10]. A. Zafar, M. Aamir, N.M. Nawi, A. Arshad, S. Riaz, A. Alruban, A.K. Dutta, S. Almotairi, A Comparison of Pooling Methods for Convolutional Neural Networks, Applied Sciences, 12 (2022) 8643. https://doi.org/10.3390/app12178643

[11]. End-to-end Multi-Modal Multi-Task Vehicle Control for Self-Driving Cars with Visual Perception, https://arxiv.org/abs/1801.06734, truy cập ngày 07 tháng 9 năm 2025

[12]. H. T. Ngoc, P. P. Hong, A. N. Quoc, L. D. Quach, Steering Angle Prediction for Autonomous Vehicles Using Deep Transfer Learning, Journal of Advances in Information Technology, 15 (2024) 138–146. https://doi:10.12720/jait.15.1.138-146

[13]. M. S. Khan, A. R. Omer, A. Yousafzai, Steering Angle Prediction of Autonomous Vehicle Using Machine Learning, 2024 Horizons of Information Technology and Engineering (HITE), (2024) 1-6.

[14]. Y. Karemore, S. Dronkar, Prediction of Steering Angle in Autonomous Vehicles Using Deep Learning Approach, EPJ Web of Conferences, 328 (2025) 01034. https://doi.org/10.1051/epjconf/202532801034