BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

LUẬN VĂN

NGÀNH KHOA HỌC MÁY TÍNH

Đề tài

XÂY DỰNG CÔNG CỤ TRỰC QUAN HÓA MÔ HÌNH MÁY HỌC 3D CỦA CÁC MÔ HÌNH MÁY HỌC PHỔ BIẾN

Sinh viên thực hiện : Huỳnh Trương Minh Quang

Mã số : B1510210

Khóa : 44

Cần Thơ, 8/2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

LUẬN VĂN

NGÀNH KHOA HỌC MÁY TÍNH

Đề tài

XÂY DỰNG CÔNG CỤ TRỰC QUAN HÓA MÔ HÌNH MÁY HỌC 3D CỦA CÁC MÔ HÌNH MÁY HỌC PHỔ BIẾN

Giáo viên hướng dẫn:

Th.S.Phạm Nguyên Hoàng

Sinh viên thực hiện:

Huỳnh Trương Minh Quang

Mã số: B1510210

Khóa: 44

Cần Thơ, 8/2022

NHẬN XÉT CỦA GIẢNG VIÊN

Cần Thơ, ngày tháng năm

(GVHD ký và ghi rõ họ tên)

LỜI CẢM ƠN

Kính dâng:

Cha, mẹ đã luôn bên con, chăm sóc dạy dỗ cho con khôn lớn, luôn giúp con vượt qua những khó khăn và áp lực.

Xin tỏ lòng biết ơn sâu sắc đến: Ths. Phạm Nguyên Hoàng đã tận tình hướng dẫn dạy bảo và tạo điều kiện thuận lợi cho em hoàn thành tốt luận văn này.

Chân thành cảm ơn các quý Thầy Cô trường Đại Học Cần Thơ đã tạo điều kiện cho em học tập tốt trong những năm vừa qua.

Cần Thơ, ngày 28 tháng 8 năm 2022

Người viết

Huỳnh Trương Minh Quang

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

ABSTRACT

Machine learning is broadly used and important in the present and the future. One of the machine learning process is neural network which are demonstrated tremendous performance in prediction and classification. Therefore, study and understand neural network are essential for newcomers and other programmers.

Visualization tools are always considered of top importance to learning and researching. Photo in 2D or model in 3D will make it easier for us to visualize than read concepts. Especially, In machine learning aspect, neural networks are too big and have heavily concepts.

Accordingly, the popular 3D modeling tool application is Unity, with the ability to create 3D models, games, movies, and more accessible to new users. Designers will easily focus on developing inside the model and still keep the visualization of the model. The modeling side will be handled quickly and powerfully with the Tensorflow library and the language used in the machine learning field is python. Thereby the application will easily reach machine learning researchers, without having to spend extra effort researching how to use other languages or libraries.

The application has popular machine learning models, allowing users to choose and observe. Once selected, the app creates a three-dimensional visual model, which can then be moved freely to observe the model's changes through individual layers.

TÓM TẮT

Để hỗ trợ học tập và nghiên cứu, công cụ trực quan hóa luôn được xem là quan trọng hàng đầu. Nghe nhìn sẽ giúp chúng ta dễ hình dung hơn là đọc hiểu, hơn nữa là ở các mô hình máy học phức tập.

Theo đó ứng dụng công cụ tạo mô hình 3 chiều phổ biến là unity, với khả năng tạo ra các mô hình 3 chiều, trò chơi, phim ảnh, và hơn nữa còn dễ tiếp cận cho người mới dùng, khiến cho người thiết kế mô hình quan sát sẽ dễ dàng tập trung vào phát triển bên trong mô hình và vẫn giữ được sự trực quan cho mô hình. Phía mô hình sẽ được xử lý một cách nhanh chóng và mạnh mẽ với thư viện Tensorflow và ngôn ngữ dùng nhiều trong lĩnh vực máy học là python. Qua đó ứng dụng sẽ dễ dàng tiếp cận người nghiên cứu máy học, không phải tốn thêm công sức nghiên cứu cách sử dụng ngôn ngữ hay thư viện khác.

Ứng dụng đã có những mô hình máy học phổ biến, giúp người dùng có thể chọn lựa mà quan sát. Sau khi chọn, ứng dụng sẽ tạo mô hình trực quan ba chiều, sau đó có thể di chuyển camera tự do để quan sát sự thay đổi của mô hình qua từng lớp riêng biệt.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Máy học đã và đang là một khái niệm rộng rãi, được nhiều người biết đến ví dụ như xe tự hành, hệ thống gợi ý của các trang web lớn như Netflix và Amazon, các trợ lý ảo giúp tương tác qua giọng nói như Bixby, Siri, Google Assistant. Ngoài ra, các ứng dụng của máy học đang được lan rộng và gia tăng trên hầu hết các lĩnh vực bao gồm nông nghiệp, công nghiệp, năng lượng, thương mại điện tử, phát hiện và chẩn đoán lỗi trên các loại máy móc, chăm sóc sức khỏe, và cả giáo dục [2].

Trong lĩnh vực nghiên cứu máy học, mạng nơ ron nhân tạo (ANN) là mô hình xử lý dữ liệu dựa trên cách các hệ thống thần kinh sinh học xử lý dữ liệu, như cách các nơ ron trong bộ não động vật có vú hoạt động. Theo Roza Dastres và Mohsen Soori, mạng thần kinh nhân tạo có thể sẽ là hy vọng duy nhất hoặc nó là cách tốt nhất để tạo nên một cổ máy có trí tuệ [1].

Từ các công việc liên quan đến dự đoán và phân loại các kiểu dữ liệu khác nhau trong đó có dữ liệu hình ảnh, văn bản, giọng nói, và video, kiến trúc mạng nơ ron đã chứng minh được hiệu suất to lớn của mình [3]. Trong đó công việc với kiểu dữ liệu hình ảnh có một kiểu mạng nơ ron đặc biệt tương thích là mạng nơ ron tích chập. Mạng nơ ron tích chập (CNN) tương tự như mạng nơ ron nhân tạo truyền thống, bao gồm các nơ ron giúp tự tối ưu thông qua quá trình học. Điểm khác biệt đáng chú ý duy nhất giữa mạng nơ ron tích chập và mạng nơ ron nhân tạo truyền thống là mạng nơ ron tích chập chủ yếu được sử dụng trong lĩnh vực nhận dạng mẫu trong hình ảnh. Điều này cho phép mã hóa các tính năng dành riêng cho hình ảnh vào kiến trúc, làm cho mạng phù hợp hơn cho các tác vụ tập trung vào hình ảnh, đồng thời giảm thêm các tham số cần thiết để thiết lập mô hình. Một trong những hạn chế lớn nhất của các dạng mạng nơ ron nhân tạo truyền thống là chúng có xu hướng gặp khó khăn với độ phức tạp trong tính toán cần thiết để tính toán dữ liệu hình ảnh [4].

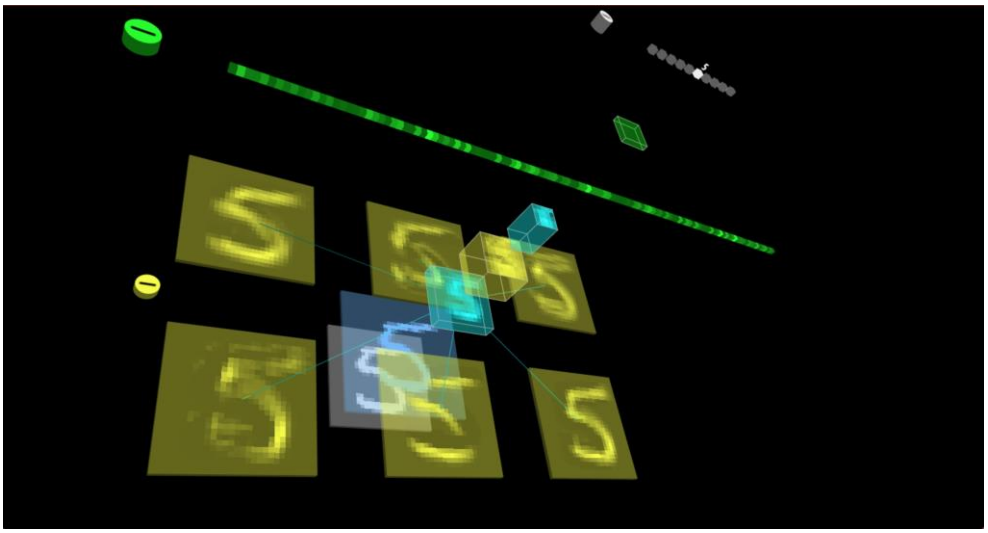
Các hệ thống máy học mạng nơ ron gần như được xem là một "chức năng hộp đen" do kích thước và sự phức tạp của nó. Tuy nhiên, với mong muốn tiếp cận học

tập, nguyên cứu và sử dụng các hệ thống máy học nơ ron trong các hệ thống phần mềm hàng ngày càng lớn. Việc trực quan hóa hệ thống nơ ron là rất cần thiết. Việc này sẽ nhắm đến các nhóm nhà khoa học máy tính, những người cần hiểu những hiểu biết cơ bản về mạng nơ ron mà không cần phải nắm rõ chi tiết và toàn bộ khái niệm. Mọi người khi đó sẽ bắt đầu tiếp cận với các mô hình máy học nơ ron mà không bị choáng ngợp bởi các thông tin được hiển thị [3].

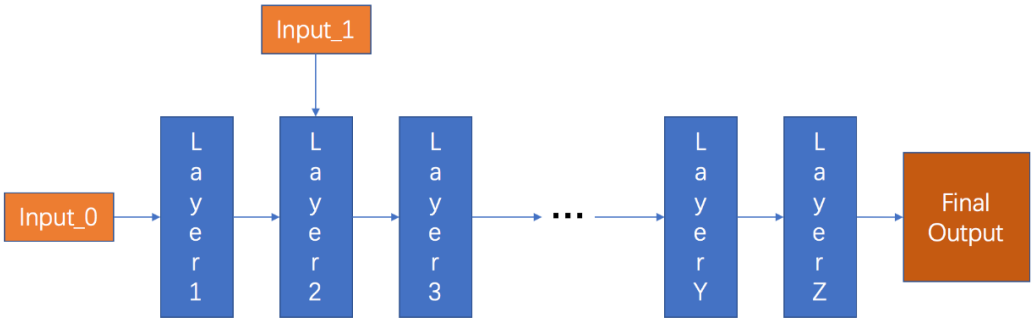
Vì vậy, dù bản thân của quá trình diễn giải, giải thích tính minh bạch của mạng nơ ron là một lĩnh vực nghiên cứu [3], nhưng công cụ vẫn nhắm đến tái tạo mô hình máy học sang mô hình ba chiều sẽ mang lại lợi ích giúp những cá nhân không phải là chuyên gia có được hiểu biết cơ bản về chức năng của mạng nơ-ron, góp một phần vào sự phát triển của máy học trong tương lai.

2. Lịch sử giải quyết vấn đề

Ứng dụng Tensorspace được tạo ra và đóng góp bởi đội ngũ tensorspaceteam, đây là ứng dụng trực quan hóa mô hình 3D mạng nơ tron được xây dựng bằng TensorFlow.js, Three.js and Tween.js. Tensorspace cung cấp bộ API có thể dùng để tạo mạng học sâu, hiển thị các mô hình đã được luyện trên trình duyệt [5].



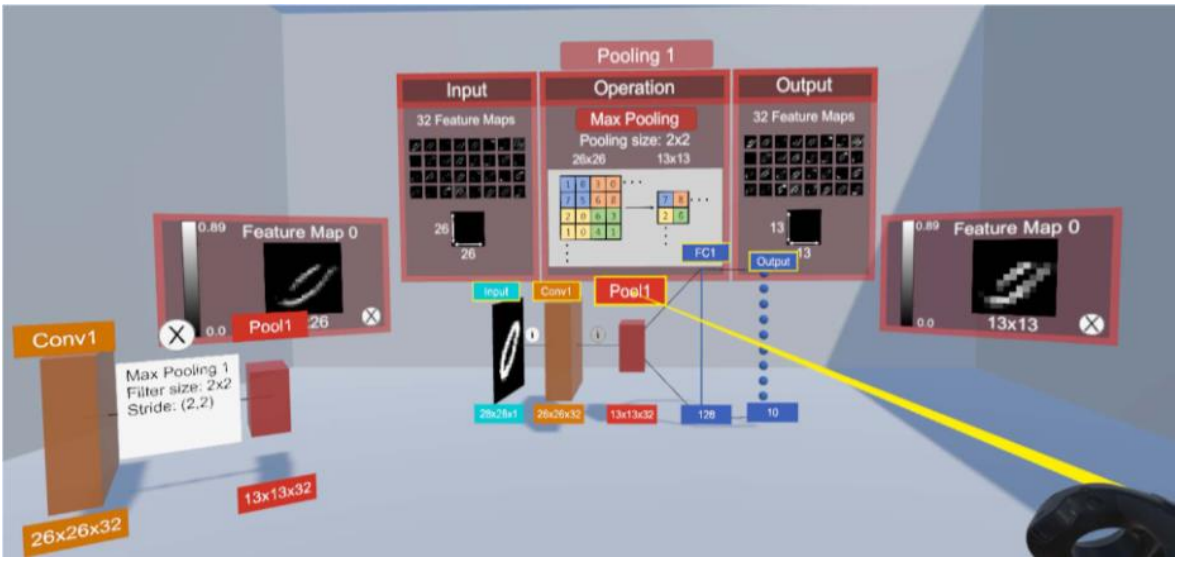
Hình 1. Mô hình LeNet trong ứng dụng Tensorspace [5]



Hình 2. Mô hình đã qua huấn luyện với một đầu ra duy nhất [5]

Ở Tensorspace, mô hình được đào tạo sẽ sử dụng dữ liệu đầu vào từ người dùng, sau đó tính toán giữa các lớp/tensor khác nhau và cuối cùng trả về kết quả đầu ra có ý nghĩa có thể được sử dụng để đánh giá thêm [5].

Theo nghiên cứu của Nadine Meissler, Annika Wohlan, et al. (2019), mô hình mạng nơ ron tích chập (CNN) đã được trực quan hóa với công nghệ thực tế ảo (VR) [3].



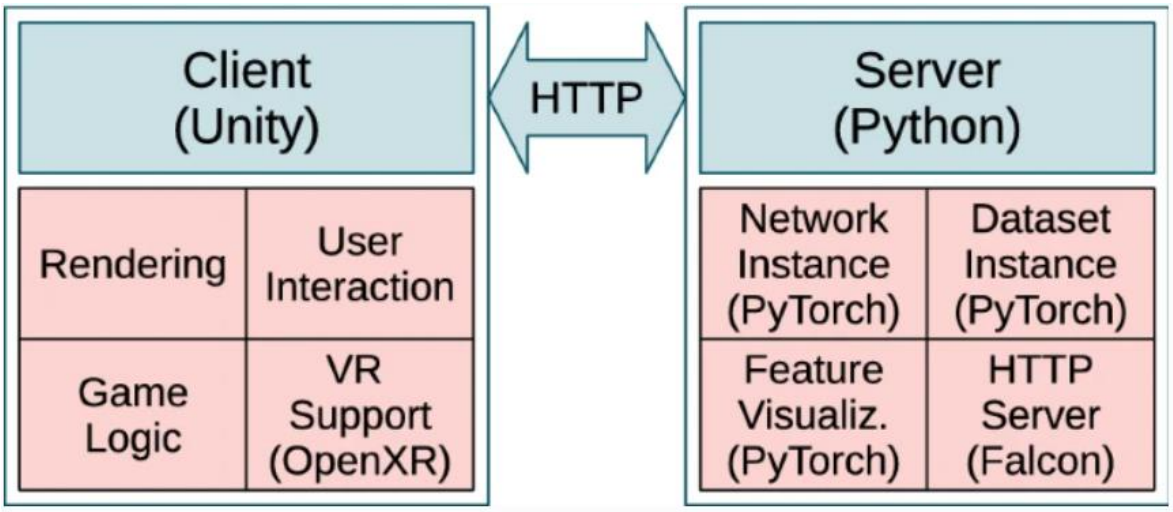
Hình 3. Trực quan hóa mạng nơ ron tích chập với thực tế ảo [3]

Nghiên cứu này đã thiết kế ứng dụng dựa trên nền tảng Unity và SteamVR. Mạng học sâu được định nghĩa sử dụng Keras, một lớp cao cấp chạy trên Tensorflow. Mô hình được thiết kế bằng ngôn ngữ Python. Sau đó, Bản đồ tính năng (feature maps) của từng lớp chập và từng lớp tổng hợp được hiển thị dưới dạng hình ảnh 2D với MATPLOTLIB bằng cách sử dụng bảng màu xám [3].

Toàn bộ phần thiết kế và triển khai mô hình 3D sẽ được thực hiện với Unity cùng với ngôn ngữ C#. Với kết quả thu được sau nghiên cứu, mọi người khi sử dụng đã không quá choáng ngợp bởi các thông tin được hiển thị, dù họ là người mới tiếp xúc với mạng nơ ron. Kiến thức về mạng nơ ron của các người tham gia đã cải thiện và có lợi ích lớn với quá trình học mạng nơ ron tích chập [3].

Christoph Linse cùng với Hammam Alshazly và Thomas Martinetz (2022), cũng đã thực hiện nghiên cứu trực quan hóa các mô hình máy học nơ ron với thực tế ảo (VR), với tên ứng dụng là DeepVisionVR [6].

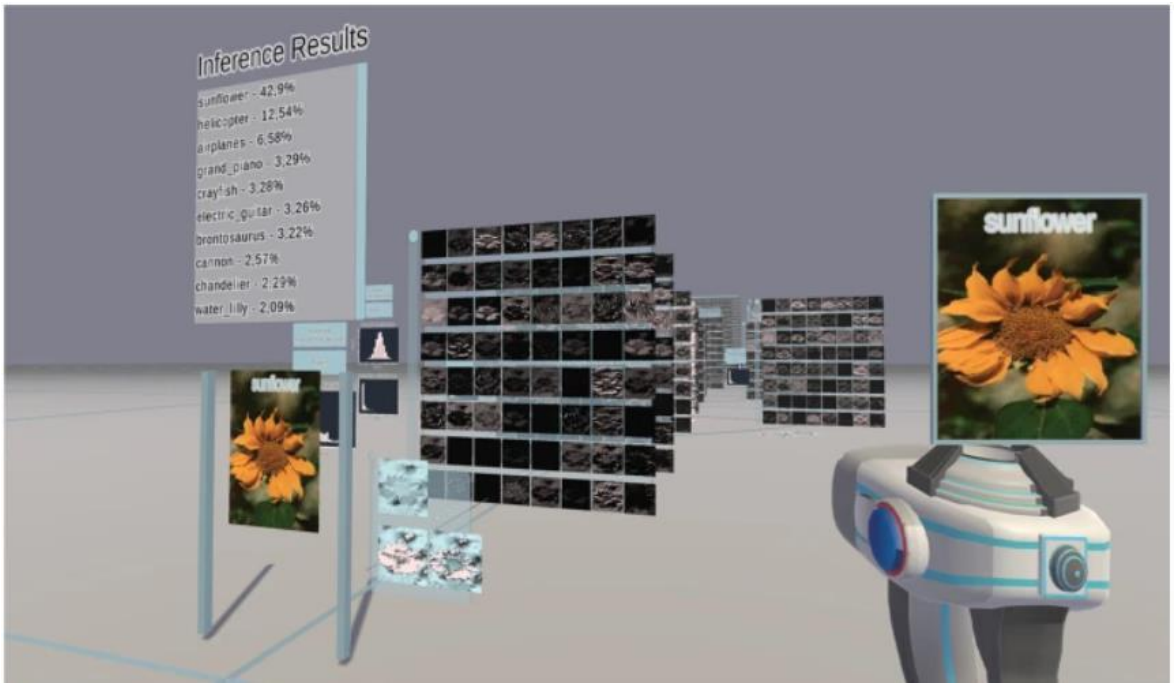
Với cấu trúc ở hình 4, được tạo nên từ mô đun Python để kết nối với Pytorch framework. Với sự hổ trợ tương tác với các thành phần trong môi trường 3D, cùng với khả năng linh hoạt và triển khai trên nhiều nền tảng. Nên game engine Unity được sử dụng để có thể trực quan hóa mô hình tích chập trong môi trường 3D. OpenXR plugin được sử dụng cho việc tích hợp VR vào ứng dụng [6].



Hình 4. Cấu trúc của ứng dụng DeepVisionVR [6]

DeepVisionVR triển khai cấu trúc client-server, hiển thị 3D với Unity (client) và xử lý với Pytorch (server). Theo đó, client sẽ trao đổi với server qua giao thức HTTP, cấu trúc mạng và feature maps sẽ được server gửi đến client. Các tính toán của chương trình sẽ được chia tải ra ở hai máy khác nhau, cấu trúc mạng cũng có thể dễ dàng thay đổi ở mã lệnh Python. Server sẽ được hỗ trợ với GPU và CUDA, trong khi đó client sẽ sử dụng GPU cùng với VR Headset để có thể để phân bổ mức sử dụng bộ nhớ và

tải tính toán. Giao tiếp sẽ thiết lập với web server Flask, và giao tiếp với client qua REST API [6].



Hình 5. CovidResnet trong môi trường 3D bằng ứng dụng DeepVisionVR [6]

Với cách thức tiếp cận Unity đơn giản không đòi khỏi kỹ năng và kinh nghiệm với engine này quá nhiều. Tận dụng các Unity prefabs và để Unity xử lý feature maps như các textures và sẽ được tải vào prefabs trước đó [6].

3. Mục tiêu đề tài

Nghiên cứu kiến trúc các mô hình máy học nơ ron tích chập. Xây dựng lại các kiến trúc nơ ron phổ biến sau đó trực quan hóa các kiến trúc mạng nơ ron trong không gian ba chiều. Có thể tương tác quan sát một cách tự do để xem sự thay đổi của các ảnh qua các lớp.

4. Đối tượng và phạm vi nghiên cứu

Trong khuôn khổ đề tài, các mô hình tuần tự được xây dựng sẵn, hoặc được người dùng chọn từ thư mục sao lưu. Các mô hình sẽ được trực quan hóa với công nghệ Unity3D. Các mô hình có thể hiển thị tốt là các mô hình máy học nơ ron tích chập được dựng bởi thư viện Keras.

Đối tượng nghiên cứu của đề tài là các kiến trúc mạng nơ ron nhân tạo tích chập bao gồm Lenet5, AlexNet. Và các phương pháp mô hình hóa chúng.

5. Phương pháp nghiên cứu

- Lý thuyết: đọc các tài liệu về máy học, về unity, trao đổi với thầy
- Thực hành: thử nghiệm và phát triển ứng dụng trên unity, thử nghiệm kết nối socket, và các tính năng khác

6. Kết quả đạt được

Sau quá trình nghiên cứu và cài đặt, công cụ giúp trực quan hóa các mô hình máy học nơ ron nhân tạo đã được hoàn thành. Ứng dụng với đầu vào đọc được các mô hình máy học được dựng sẵn hoặc người dùng xây dựng thông qua thư viện Keras. Sau đó, ứng dụng có thể chiết xuất thông tin và hiển thị cấu trúc của mô hình nơ ron trực quan trong không gian ba chiều. Người dùng có thể quan sát sự thay đổi qua từng lớp nơ ron, theo dõi thông tin của các lớp nơ ron cùng với khả năng quan sát mô hình một cách tự do.

7. Bố cục luận văn

Phần giới thiệu

Giới thiệu tổng quát về đề tài.

Phần nội dung

Chương 1 : Mô tả chi tiết bài toán, các cơ sở lý thuyết đã áp dụng.

Chương 2 : Thiết kế, cài đặt giải thuật, biểu diễn cơ sở dữ liệu, trình bày các bước xây dựng hệ thống bằng phương pháp lọc cộng tác.

Chương 3 : Kiểm thử hệ thống và đánh giá độ chính xác, tốc độ của hệ thống.

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển hệ thống.

PHẦN NỘI DUNG

CHƯƠNG 1

MÔ TẢ BÀI TOÁN

1. Mô tả chi tiết bài toán

Với mục tiêu dựng lại các mô hình máy học nơ ron tích chập phổ biến trong không gian ba chiều cùng với khả năng tương tác với các lớp nơ ron. Bài toán sẽ nhận vào là tệp tin mô hình máy học được dựng sẵn hoặc người dùng xây dựng thông qua thư viện Keras. Sau đó, kết quả sẽ là mô hình máy học nơ ron cùng với các hình ảnh được chiết xuất sau khi đi qua từng lớp nơ ron sẽ hiển thị theo thứ tự từng lớp trên không gian ba chiều.

Sử dụng mô hình client-server để phân biệt quá trình xử lý mô hình máy học và quá trình trực quan hóa cũng như giao diện người dùng. Server được viết với ngôn ngữ Python, ngôn ngữ phổ biến trong máy học, có những thư viện hổ trợ như numpy, tensorflow, cv2 giúp dễ dàng xử lý với các mô hình máy học, mạng nơ ron.

Client được xử lý và cài đặt bằng game engine Unity với mục đích tận dụng khả năng tương tác và chiết xuất mô hình 3D mà không phải thao tác sâu với 3D graphics API. Server và Client sẽ liên kết với nhau thông qua liên kết socket, với giao thức TCP/IP, sau khi liên kết sẽ có thể truyền tải dữ liệu qua lại từ hai phía.

Sau khi đã thực hiện thành công liên kết, ứng dụng sẽ cho phép người dùng chọn mô hình từ thư mục lưu trữ hoặc cấu trúc nơ ron mạng học sâu phổ biến có sẵn tại giao diện người dùng. Các lựa chọn sẽ được gửi đến phía server xử lý. Server sẽ đọc các mô hình máy học Keras với định dạng SavedModel, hoặc các cấu trúc nơ ron mạng học sâu đã được cài đặt.

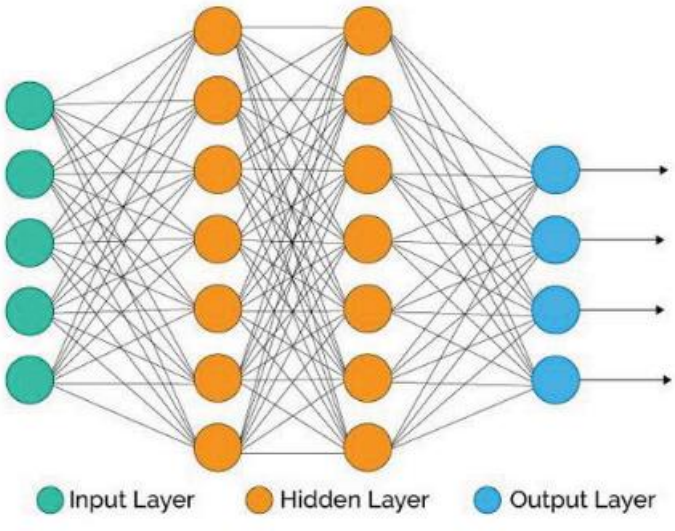
Sau khi đã chọn, mô hình sẽ được chạy với một tệp ảnh mẫu có sẵn tại server. Sau đó, các thông tin cần thiết của mô hình cùng với các bản đồ tính năng (feature maps) của từng lớp tích chập và từng lớp tổng hợp được dưới dạng hình ảnh 2D sẽ được gửi về phía client Unity để tạo thành các textures, các textures sẽ được lần lượt đưa vào các prefabs tương ứng với các lớp. Do trong môi trường 3D các textures 2D phải trở thành các sprites, và được chiết xuất thành mô hình 3D với SpriteRenderer. Các prefabs trước đó chính là các GameObject chứa SpriteRenderer được định dạng sẵn.

Nhằm mục đích thuận tiện cho người dùng, mô hình cần phải tích hợp khả năng tương tác như xoay, di chuyển. Cùng với đó là ứng với từng lớp khi nhấp chuột chọn các feature maps sẽ được chiết xuất, với lớp tương tứng đang được tương tác sẽ được thể hiện ở một bản đồ lớp nhỏ ở góc dưới màn hình. Thông tin lớp cũng sẽ được hiển thị dưới góc màn hình giúp người dùng có thể dễ dàng hình dung và nắm bắt khi quan sát ảnh của các lớp.

2. Vấn đề và giải pháp liên quan đến bài toán

2.1 Các mô hình máy học

Mạng thần kinh nhân tạo được thiết kế giống như bộ não con người, với các nút nơ ron thần kinh được kết nối với nhau theo kiểu giống như các mạng lưới. Tế bào thần kinh nơ ron là hàng tỷ tế bào tạo nên bộ não con người. Mỗi tế bào thần kinh được tạo thành từ một tế bào xử lý thông tin bằng cách đưa nó vào (input) và đi ra (output) từ bộ não. Ý tưởng chính của các mạng như vậy được lấy cảm hứng từ cách thức hoạt động của hệ thống thần kinh sinh học, để xử lý dữ liệu và thông tin nhằm học hỏi và tạo ra kiến thức. Yếu tố chính của ý tưởng này là tạo ra các cấu trúc mới cho hệ thống xử lý thông tin. Kiến trúc mạng nơ-ron nhân tạo được thể hiện trong hình 6 [1].



Hình 6. Cấu trúc mạng nơ ron nhân tạo [1]

Các tế bào thần kinh được liên kết với nhau chặt chẽ và được tổ chức thành các lớp. Lớp đầu vào nhận dữ liệu, trong khi lớp đầu ra tạo ra kết quả cuối cùng [1].

2.2 Mạng nơ ron tích chập

Thị giác máy tính (Computer Vision), một trong những ngành trí tuệ nhân tạo phát triển nhanh nhất, ngày càng trở nên quan trọng trong xã hội của chúng ta do ứng dụng rộng rãi của nó trong các lĩnh vực khác nhau như chăm sóc sức khỏe và y học (thuật toán có thể chẩn đoán bệnh bằng hình ảnh y tế), robot dựa trên thị giác, xe ô tô tự lái (có thể nhìn thấy và lái xe an toàn). Mạng nơ ron tích chập là kiến trúc lấy cảm hứng từ sinh học và đại diện cho cốt lõi của các thuật toán học sâu trong thị giác máy tính [10].

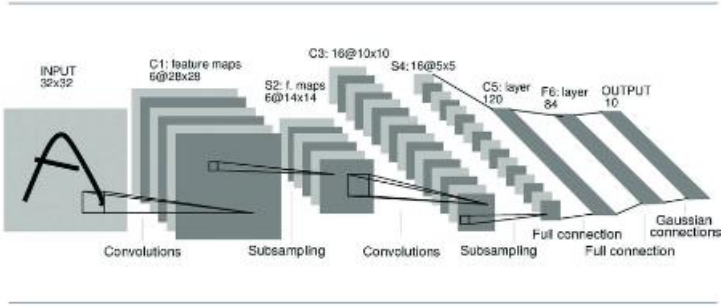
Mạng nơ ron tích chập (CNNs) là một dạng đặc biệt của mạng nơ ron nhân tạo (ANNs), đã được chứng minh là có hiệu suất cao trên các tác vụ liên quan đến thị giác khác nhau, bao gồm phân loại hình ảnh, phân đoạn hình ảnh, truy xuất hình ảnh, phát hiện đối tượng, chú thích hình ảnh, nhận dạng khuôn mặt, ước tính tư thế, nhận dạng biển báo giao thông, xử lý giọng nói, chuyển kiểu nơ-ron, v.v. Mạng nơ-ron tích chập đã trở thành một lĩnh vực được quan tâm nhanh chóng trong những năm gần đây [10].

2.3 Các cấu trúc mạng nơ ron phổ biến

Các kiến trúc CNN phổ biến nhất xếp chồng một vài lớp tích chập lại với nhau, nó theo sau là lớp tổng hợp (pooling layer), sau đó kiểu mẫu này lặp lại và thêm vào cuối các lớp đã được kết nối đầy đủ (Full Connected Layer). Các kiến trúc CNN cổ điển là LeNet-5, AlexNet, VGGNet [10] và GoogleNet là kiến trúc CNN hiện đại.

LeNet-5 là CNN đầu tiên, nó được đề xuất bởi Yann LeCun và nhóm của ông tại Bell Labs vào năm 1998, kiến trúc này được thể hiện trong Hình 7. Mạng này được dành cho nhận dạng chữ số và hệ thống được sử dụng để phát hiện chữ ký viết tay trong séc và nó đã được triển khai thương mại thành công cho mục đích này. Nó chỉ được sáng tác trên một vài lớp và một vài bộ lọc, do những hạn chế của máy tính vào thời điểm đó. Như được hiển thị trong Hình 7, kiến trúc có hai lớp tích chập, hai lớp tổng hợp trung bình, hai lớp được kết nối đầy đủ và một lớp đầu ra với kết nối Gaussian. LeNet-5 có 60.000 thông số. Với chức năng kích hoạt, chức năng kích hoạt

tanh được sử dụng. Hàm tiếp tuyến hypebol (Tanh) là một biến thể nhỏ của hàm sigmoid có tâm là 0 [10].



Hình 7. Cấu trúc Lenet-5 [10].

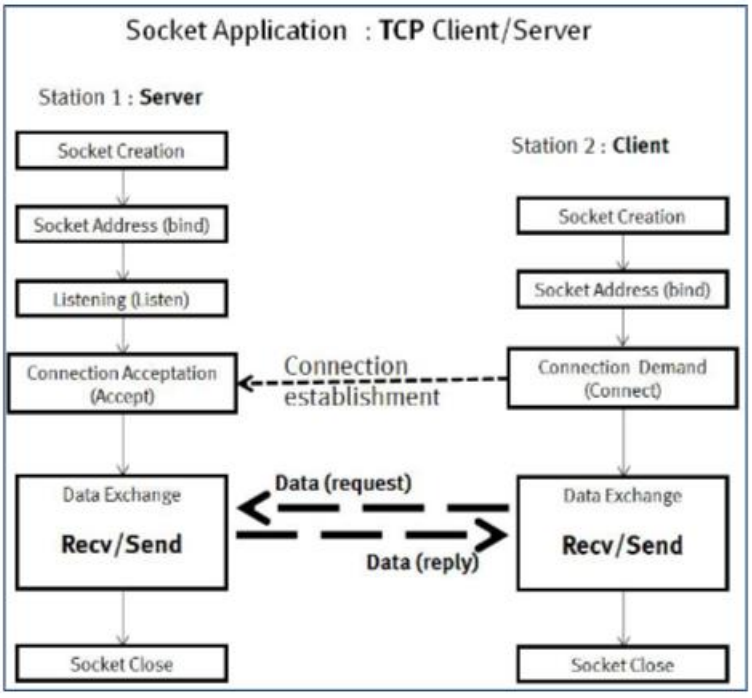
Kiến trúc AlexNet là công trình đầu tiên phổ biến Mạng kết hợp trong Thị giác máy tính, có tỷ lệ lỗi top 5 là 15,4% so với 26,2% cho mạng thấp nhất tiếp theo. AlexNet tuân theo mô hình của kiến trúc LeNet-5, nhưng nó sâu hơn, lớn hơn và nổi bật với các lớp tích chập xếp chồng lên nhau. Hàm kích hoạt tanh, được sử dụng trong LeNet-5, được thay thế bằng hàm ReLU, như hàm mất mát, hàm mất mát entropy chéo được sử dụng. AlexNet đã sử dụng một tập huấn luyện lớn hơn nhiều. LeNet-5 được đào tạo trên tập dữ liệu MNIST với 50.000 hình ảnh và 10 danh mục, AlexNet đã sử dụng một tập hợp con của tập dữ liệu ImageNet với tập huấn luyện chứa một triệu hình ảnh màu và 1000 danh mục [10].

Mạng VGG được giới thiệu vào năm 2014 bởi Karen Simonyan và Andrew Zisserman. Vào thời điểm đó, nó được coi là một mạng lưới rất sâu. Đóng góp chính của nó là chỉ ra rằng độ sâu của mạng là thành phần quan trọng để đạt được độ chính xác nhận dạng hoặc phân loại tốt hơn trong CNN. VGGNet sử dụng các bộ lọc 3x3, có hai bộ lọc 3x3 liên tiếp mang lại trường tiếp nhận hiệu quả là 5x5 và ba bộ lọc 3x3 mang lại trường tiếp nhận gồm các bộ lọc 7x7. Số lượng bộ lọc trong kiến trúc tăng gấp đôi sau mỗi hoạt động tổng hợp tối đa [10].

2.4 Kết nối và trao đổi giữa server và client

Giao thức điều khiển truyền tải (TCP) là một trong những giao thức cốt lõi của bộ giao thức Internet (IP). TCP cung cấp khả năng truyền dữ liệu đáng tin cậy, có thứ tự và được kiểm tra lỗi giữa hai ứng dụng đang chạy trên một thiết bị được kết nối

với mạng cục bộ, Intranet hoặc Internet. Một lượng lớn các giao thức tiêu chuẩn có thể được gói gọn trong TCP: HTTP, HTTPS, SMTP, POP3, IMAP, SSH, FTP, Telnet... Và giao thức công nghiệp phổ biến nhất Modbus TCP. Các kết nối TCP là song công hoàn toàn và End-to-End. Song công hoàn toàn có nghĩa là lưu lượng có thể đi theo cả hai hướng cùng một lúc. End-to-End có nghĩa là mỗi kết nối có chính xác hai điểm cuối. Vì vậy, không giống như giao thức UDP, TCP không hỗ trợ phát đa hướng và phát sóng. Vì TCP cho phép truyền dữ liệu đáng tin cậy giữa hai đối tác nên nó thường được sử dụng cho các tác vụ cụ thể như cấu hình thiết bị từ xa (truyền tham số), trao đổi dữ liệu giữa robot và tầm nhìn hệ thống, quản lý RFID, truyền tải (upload/download) chương trình [11].



Hình 8. Mối quan hệ giữa client và server trong phiên TCP [11]

Phiên TCP có thể được chia thành ba giai đoạn [11]:

- Giai đoạn thiết lập kết nối: kết nối giữa 2 thiết bị (Client/Server) phải được thiết lập đúng trước khi bắt đầu truyền dữ liệu
- Giai đoạn truyền dữ liệu: quá trình trao đổi dữ liệu giữa Client và Server được kết nối

- Giai đoạn kết thúc kết nối: khi kết thúc và đóng kết nối, đồng thời tất cả các tài nguyên được phân bổ được giải phóng

Từ quan điểm của Máy chủ TCP, điểm bắt đầu cho kết nối là trạng thái Nghe trong đó Máy chủ chờ yêu cầu kết nối từ bất kỳ Máy khách TCP từ xa nào. Nhưng trước khi nghe, trước tiên Máy chủ phải tạo ổ cắm và liên kết với (để mở) một cổng giao tiếp. Khi Máy chủ ở trạng thái Nghe, Máy khách có thể bắt đầu kết nối (chức năng Connect()). Nếu yêu cầu của Máy khách được Máy chủ chấp nhận, kết nối được thiết lập và chuyển trực tiếp sang Giai đoạn truyền dữ liệu. Trong giai đoạn này, cả Client và Server đều có thể giao tiếp. Hàm Recv() dùng để nhận dữ liệu và hàm Send() dùng để gửi dữ liệu. Về mặt lý thuyết, thứ tự gọi hàm Send() và Recv() không không thành vấn đề. Bạn có thể gọi cái này trước cái kia hoặc ngược lại. Tuy nhiên, đối với cùng một quá trình, trình tự Recv() / Send() có thể bị chặn và ứng dụng sẽ không thể gửi gì đó nếu trước đó, nó không nhận được gì [11].

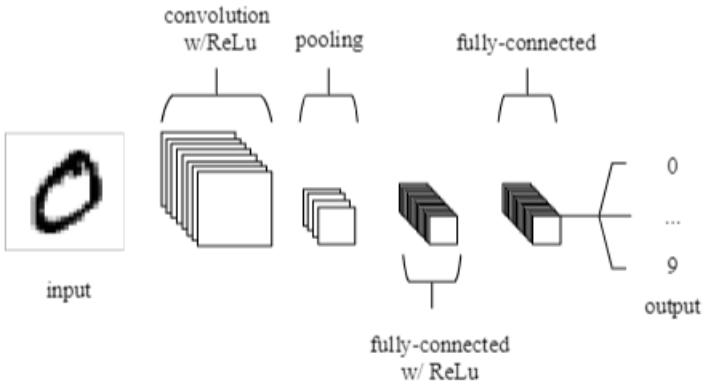
2.5 Mô hình hóa các lớp của mạng nơ ron

Mô hình hóa các lớp mạng nơ ron là giúp hiển thị các lớp của mạng nơ ron, từ ba lớp chính là lớp đầu vào, lớp ẩn và lớp đầu ra ở hình 6. Lớp đầu vào được kết nối với lớp ẩn, lớp ẩn được kết nối với lớp đầu ra. Phép tính cuối cùng sẽ tạo ra một giá trị ở lớp đầu ra [12].

Trong đó lớp ẩn có chức năng tăng khả năng của mạng tăng toàn bộ sức mạnh tính toán của mạng, do đó nó sẽ dễ dàng thực hiện công việc của mình hơn. Với mạng nơ ron nhân tạo không có các lớp ẩn, khả năng của nó sẽ rất hạn chế. Các lớp ẩn có thể bao gồm một hoặc nhiều lớp. Mỗi lớp có một số tế bào thần kinh (nơ ron). Cho đến nay, không có quy tắc dứt khoát nào về số lượng lớp ẩn và số lượng tế bào thần kinh tốt nhất trong lớp ẩn trong ANN. Người dùng vẫn gặp khó khăn trong việc xác định số lượng lớp ẩn và số lượng tế bào thần kinh lý tưởng trong lớp ẩn của hệ thống ANN [12].

Trong mạng nơ ron tích chập (CNN), bao gồm ba loại lớp. Đây là các lớp tích chập, lớp tổng hợp và lớp được kết nối đầy đủ. Khi các lớp này được xếp chồng lên

nhau, một kiến trúc CNN đã được hình thành [4]. Kiến trúc CNN đơn giản hóa cho phân loại MNIST được minh họa trong hình 9.



Hình 9. Cấu trúc CNN đơn giản, gồm 5 lớp [4]

Chức năng cơ bản của ví dụ CNN ở trên có thể được chia thành bốn phần chính [4]

1. Như được tìm thấy trong các dạng ANN khác, lớp đầu vào sẽ chứa các giá trị pixel của hình ảnh.
2. Lớp tích chập sẽ xác định đầu ra của các nơ-ron được kết nối với các vùng cục bộ của đầu vào thông qua tính toán tích vô hướng giữa trọng số của chúng và vùng được kết nối với khối lượng đầu vào. Đơn vị tuyến tính được chỉnh lưu (ReLu) nhằm mục đích áp dụng chức năng kích hoạt 'theo từng phần tử', chẳng hạn như sigmoid cho đầu ra của kích hoạt do lớp trước đó tạo ra.
3. Sau đó, lớp tổng hợp sẽ chỉ thực hiện lấy mẫu xuống dọc theo chiều không gian của đầu vào nhất định, tiếp tục giảm số lượng pa-rameter trong lần kích hoạt đó.
4. Sau đó, các lớp được kết nối đầy đủ sẽ thực hiện các nhiệm vụ giống như trong các ANN tiêu chuẩn và cố gắng tạo ra điểm số của lớp từ các lần kích hoạt, được sử dụng để phân loại. Cũng có ý kiến cho rằng ReLu có thể được sử dụng giữa các lớp này để cải thiện hiệu suất.

Thông qua phương pháp chuyển đổi đơn giản này, các CNN có thể chuyển đổi từng lớp đầu vào ban đầu bằng cách sử dụng các kỹ thuật lấy mẫu tích chập và lấy mẫu xuống để tạo ra điểm số lớp cho mục đích phân loại và hồi quy [4].

3. Công cụ và thư viện hỗ trợ

3.1 Ngôn ngữ lập trình C#

C# là một ngôn ngữ lập trình hướng đối tượng hiện đại, được tạo và phát triển bởi Microsoft cùng với nền tảng .NET. Cùng là ngôn ngữ cấp cao như C++ hay Java, tất cả các chương trình của C# đều có tính hướng đối tượng. Các chương trình C# hầu hết thường chạy trên nền tảng Windows, nhưng vẫn được hỗ trợ các nền tảng khác như Android, IOS, MacOS [7].

Ngày nay ngôn ngữ C# được sử dụng bởi hàng triệu lập trình viên, là một ngôn ngữ dễ học khi so với các đàn anh là C hay C++. Tuy vậy, C# vẫn được dùng trong rất nhiều công ty lớn khiến C# trở thành một trong những ngôn ngữ lập trình phổ biến nhất hiện nay [7].

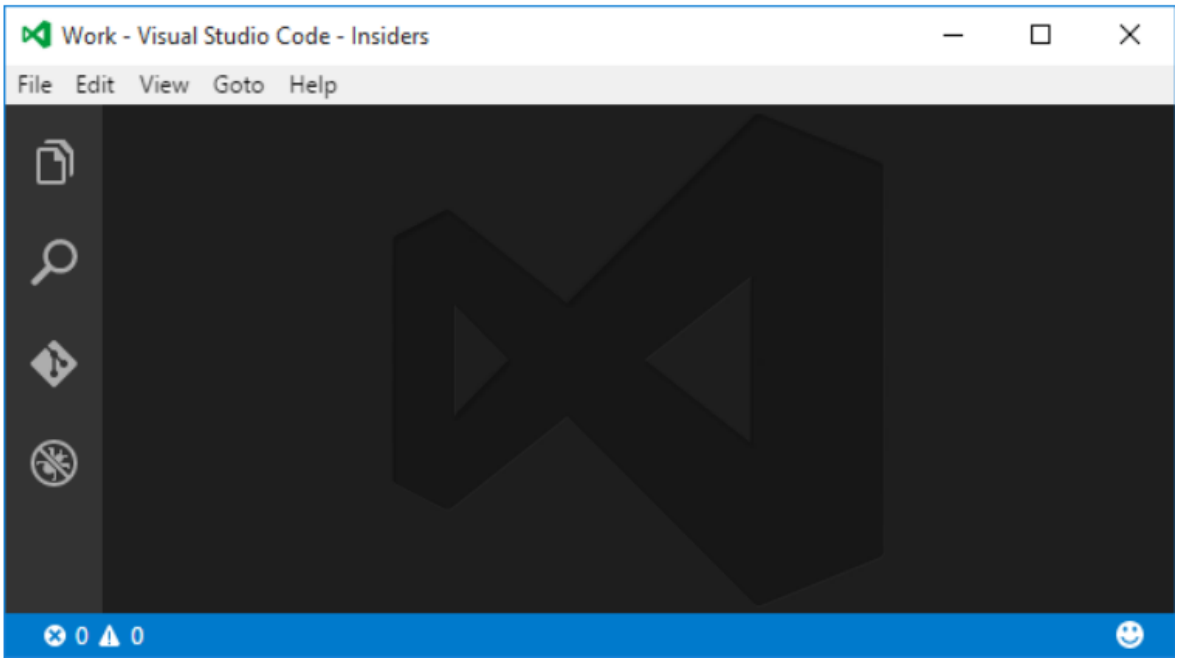
3.2 Ngôn ngữ lập trình python

Năm 1991, ngôn ngữ Python được phát triển bởi Guido van Rossum. Python là ngôn ngữ lập trình hướng đối tượng, thông dịch và tương tác. Nó cung cấp các cấu trúc dữ liệu cấp cao như danh sách (list), bộ dữ liệu (tuples), bộ (sets), từ điển (dictionaries) [8].

Python là ngôn ngữ với cú pháp đơn giản, dễ học, dễ tiếp cận và là một trong những ngôn ngữ mạnh mẽ nhất hiện nay. Hầu như có nhiều lợi thế nhất trong các ngôn ngữ lập trình hiện nay khi sở hữu rất nhiều thư viện giúp giảm số mã lệnh xuống còn một phần ba, vì thế mà Python đã đạt được đỉnh cao nhất về máy học [8].

3.3 Visual Studio Code

Visual Studio Code (VS Code) là trình chỉnh sửa mã nguồn mở miễn phí để phát triển và gỡ lỗi các ứng dụng web và đám mây, có sẵn miễn phí trên Linux, OS X và Windows. VS Code hỗ trợ hơn 30 ngôn ngữ lập trình, đánh dấu và cơ sở dữ liệu khác nhau, một số trong số đó là JavaScript, C#, C++, PHP, Java, HTML, R, CSS, SQL, Markdown, TypeScript, Less, Sass, JSON, XML và Python [9].



Hình 10. Giao diện Visual Studio Code Insider [9]

3.4 Unity

Là một môi trường lập trình và thiết kế trò chơi được công ty Unity Technologies có trụ sở tại San Francisco, California phát triển. Công cụ Unity cho phép người dùng tạo ra những trò chơi trong không gian hai chiều, ba chiều hoặc thực tế ảo. Thế mạnh của Unity nằm ở việc đây là một engine cho phép người dùng lựa chọn giữa việc kéo thả và điều chỉnh thông số của các vật thể, Cùng với đó là Unity Script giúp người dung sẽ dụng ngôn ngữ lập trình C# và các thư viện của riêng Unity cho phép người dùng tạo những event, thiết lập hành vi cho vật thể. Ngoài ra, Unity còn hỗ trợ tạo ra ứng dụng đa nền tảng, giúp người dùng có thể dễ dàng xuất dự án thành các tập tình thực thi trên các nền tảng máy tính khác nhau [13].

3.5 Tensorflow

TensorFlow là một hệ thống máy học hoạt động ở quy mô lớn và trong môi trường không đồng nhất. TensorFlow sử dụng biểu đồ luồng dữ liệu để thể hiện tính toán, trạng thái được chia sẻ và các hoạt động làm thay đổi trạng thái đó. Nó ánh xạ các nút của biểu đồ luồng dữ liệu trên nhiều máy trong một cụm và trong một máy trên nhiều thiết bị tính toán, bao gồm CPU đa lõi, GPU đa năng. TensorFlow cho phép các nhà phát triển thử nghiệm các thuật toán đào tạo và tối ưu hóa mới. TensorFlow hỗ trợ nhiều ứng dụng khác nhau, đặc biệt hỗ trợ mạnh mẽ cho đào tạo và suy luận trên các mạng lưới nơ ron học sâu [14].

3.6 OpenCV

OpenCV được bắt đầu tại Intel vào năm 1999 bởi Gary Bradsky, và bản phát hành đầu tiên ra mắt vào năm 2000. Vadim Pisarevsky cùng với Gary Bradsky quản lý nhóm OpenCV phần mềm tiếng Nga của Intel. Năm 2005, OpenCV đã được sử dụng trên Stanley, chiếc xe đã giành chiến thắng trong cuộc thi DARPA Grand Challenge năm 2005. Sau đó, sự phát triển tích cực của nó tiếp tục dưới sự hỗ trợ của Willow Garage với Gary Bradsky và Vadim Pisarevsky lãnh đạo dự án. OpenCV hiện hỗ trợ vô số thuật toán liên quan đến Thị giác máy tính và Học máy và đang mở rộng từng ngày [15].

OpenCV hỗ trợ nhiều ngôn ngữ lập trình như C++, Python, Java, v.v. và có sẵn trên các nền tảng khác nhau bao gồm Windows, Linux, OS X, Android và iOS. Các giao diện cho hoạt động GPU tốc độ cao dựa trên CUDA và OpenCL cũng đang được phát triển tích cực [15].

OpenCV-Python là API Python cho OpenCV, kết hợp những phẩm chất tốt nhất của OpenCV C++ API và ngôn ngữ Python. OpenCV-Python là một thư viện các liên kết Python được thiết kế để giải quyết các vấn đề về thị giác máy tính [15].

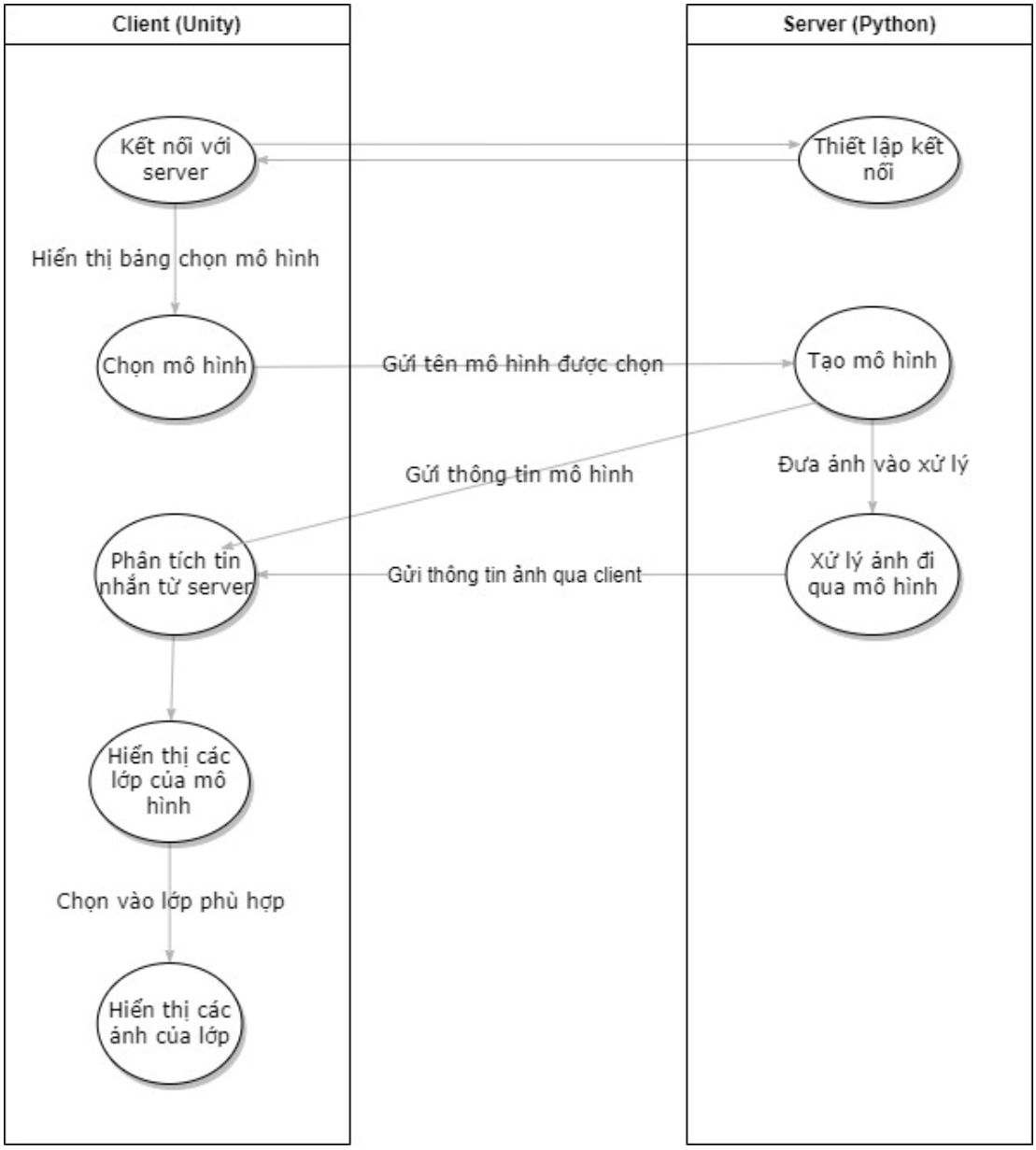
OpenCV-Python sử dụng Numpy, đây là thư viện được tối ưu hóa cao cho các phép toán số với cú pháp kiểu MATLAB. Tất cả các cấu trúc mảng OpenCV được chuyển đổi thành và từ các mảng Numpy. Điều này cũng giúp dễ dàng tích hợp với các thư viện khác sử dụng Numpy như SciPy và Matplotlib [15].

CHƯƠNG 2

THIẾT KẾ VÀ CÀI ĐẶT

1. Thiết kế hệ thống

Hệ thống sẽ được thiết kế với chi tiết sơ đồ thiết kế như hình 11



Hình 11. Sơ đồ các bước thiết kế hệ thống

Hệ thống cho phép người dùng chọn các mô hình máy học nơ ron được dựng sẵn hoặc người dùng có thể chọn từ thư mục mô hình máy học của mình. Mô hình sau

khi được chọn sẽ được đọc bởi thư viện keras và dựng thành mô hình máy học tại server.

Unity sẽ tạo ra các khối lập phương ứng với số lớp của mô hình. Người dùng có thể chọn khối lập phương phù hợp (lớp phù hợp) để có thể hiển thị các hình ảnh của lớp đó.

Hệ thống tạo ra kết nối giữa hai đối tượng là client và server. Client sử dụng công cụ Unity nhằm mục đích hỗ trợ quá trình trực quan hóa trong không gian ba chiều, server được viết nên bởi ngôn ngữ Python, dễ dàng trong quá trình xử lý các mô hình máy học. Kết nối với nhau qua kết nối socket với giao thức TCP. Sau đó người dùng có thể chọn mô hình phù hợp ở phía client. Thông tin đó sẽ được truyền qua server.

Với kiến trúc mô hình máy học đã được dựng nên, một hình ảnh được cài sẵn ở server sẽ được xử lý qua các lớp nơ ron của mô hình. Các hình ảnh được xử lý qua từng lớp của mô hình sẽ được lưu lại và được gửi qua phía Unity.

Các ảnh của từng lớp nơ ron sẽ được hiển thị trên nền ba chiều sau khi người dùng nhấp chuột trái vào khối lập phương ứng với mỗi lớp của mô hình.

2. Cài đặt giải thuật

2.1 Tạo mô hình máy học

Mô hình máy học được cài đặt sẵn cho chương trình được xây dựng thông qua thư viện Keras. Sử dụng cấu trúc mô hình tuần tự, là một ngắn xếp các lớp nơ ron trong mạng với mỗi lớp chỉ có duy nhất một tensor đầu vào và một tensor đầu ra. Với tensor là đối tượng hình học miêu tả quan hệ tuyến tính.

Các lớp bên trong mô hình tuần tự sẽ sử dụng các lớp như:

- Lớp tuần tự Conv2D (2D convolution layer): lớp tích chập 2D tạo ra một nhân được tích chập với đầu vào của lớp để tạo ra một tensor đầu ra.

- Lớp cốt lõi Dense layer: một trong các lớp cốt lõi (core layers), lớp nơ ron được kết nối một cách dày đặc (densely-connected)

- Lớp cốt lõi Activation layer: một trong các lớp cốt lõi (core layers), sử dụng để áp dụng chức năng kích hoạt cho tensor đầu ra

- Các lớp tổng hợp dữ liệu pooling layer như MaxPooling2D, AveragePooling2D: tổng hợp dữ liệu cho dữ liệu không gian hai chiều, giảm mẫu đầu vào dọc theo kích thước không gian của nó (chiều cao và chiều rộng) bằng cách lấy giá trị tối đa (Max) hoặc trung bình (Avarage) dựa theo loại pooling layer trên một cửa sổ đầu vào (có kích thước được xác định bởi pool_size) cho mỗi kênh của đầu vào.

- Lớp Dropout: Lớp Dropout đặt ngẫu nhiên các đơn vị đầu vào thành giá trị 0 với tần suất rate (từ 0 đến 1, tỷ lệ của các đơn vị ở tensor đầu vào) ở mỗi bước trong thời gian huấn luyện, điều này giúp ngăn chặn quá khớp với mô hình ban đầu (overfitting).

- Lớp BatchNormalization: Lớp chuẩn hóa đầu vào. BatchNormalization áp dụng một phép biến đổi duy trì đầu ra trung bình gần bằng 0 và độ lệch chuẩn đầu ra gần bằng 1.

- Lớp Flatten: làm phẳng đầu vào.

2.2 Tạo kết nối

Mô hình sử dụng giao thức TCP, giao tiếp qua lại giữa client và server. Server sử dụng thư viện socket của ngôn ngữ Python, với hai phương thức recv và send để lần lượt nhận và gửi dữ liệu sau khi đã kết nối với phía client.

Client được viết bằng ngôn ngữ C# trong Unity script, sử dụng giao diện Windows Socket (Winsock) được tích hợp trong không gian tên System.Net.Sockets, với thư viện NetworkStream hỗ trợ phương thức Read và Write giúp lần lượt đọc và gửi thông điệp đến phía server.

2.3 Xử lý gói tin

Sau khi đã thiết lập kết nối TCP giữa server và client, các gói tin được trao đổi qua lại cho đến khi kết thúc kết nối.

Tin nhắn dạng ký tự sẽ được mã hóa sang định dạng UTF-8 cho phép đồng nhất trong quá trình chuyển và nhận giữa hai phía. Trong đó phía server sử dụng phương thức encode để chuyển đổi dạng ký tự sang UTF-8. Phía client sử dụng phương thức Encoding trong không gian tên System.Text để chuyển đổi.

Tin nhắn sẽ được nhận dưới dạng bytes. Sẽ có nhiều cách xử lý sau khi nhận dược tin nhắn. Phía server sẽ giải mã tin nhắn sang dạng ký tự với phương thức decode.

Khi chuyển đổi thông tin từ mô hình máy học nơ ron từ phía server sang client. Ta sẽ sử dụng định dạng Json để có thể dễ dàng chiết xuất thông tin từ mô hình. Sử dụng phương thức to_json để chuyển đổi. Chiết xuất thông tin ở phía client với phương thức Parse được cung cấp bởi thư viện Newtonsoft.Json.Linq.

Khi chuyển đổi thông tin từ các bảng đồ tính năng sau khi đã áp dụng ảnh đầu vào qua từng lớp của mô hình. Bảng đồ tính năng sẽ được thư viện cv2 cung cấp các phương thức xử lý hình ảnh tương thích, sau đó sử dụng phương thức imencode và tobytes nhằm chuyển đổi hình ảnh sang dạng bytes và chuyển qua client.

2.4 Xử lý đa luồng

Quá trình gửi hình ảnh từ server sang client và quá trình chiết xuất hình ảnh sẽ cần đến một khoảng thời gian tương đối. Cho nên chúng ta cần áp dụng xử lý đa luồng giúp ứng dụng không bị đứng trong quá trình đó, người dùng vẫn có thể tương tác bằng con trỏ chuột trên giao diện người dùng khi các quá trình đang chạy.

Ở C#, ta có thể sử dụng không gian tên System.Threading để có thể xử lý nhiều luồng một lúc. Nhưng Unity không cho phép các phương thức liên quan đến quá trình kết xuất đồ họa nằm ở luồng khác luồng chính. Ta sử dụng hàng đợi kết hợp với phương thức Update (phương thức sẽ được chạy tự động mỗi khung hình ở main thread trong Unity script).

2.5 Xử lý chuyển động camera

Sử dụng phép chiếu phối cảnh, camera được đặt mặc định tại vị trí 0, 0, 0.

Nhằm mục đích giúp người dùng có thể dễ dàng quan sát các hình ảnh của các lớp, camera sẽ được cài đặt cơ chế di chuyển theo mong muốn của người dùng.

Sử dụng các phương thức có sẵn của không gian tên UnityEngine.Input như GetMouseButton, GetKey, GetKeyDown để nhận sự kiện từ bàn phím và chuột do người dùng thao tác.

Sử dụng các phương thức của không gian tên UnityEngine.Camera để nhận vector từ trục các trục x, y, z của camera, như transform.forward để nhận vector trục z, transform.right để nhận vector trục x.

Thay đổi vị trí của camera bằng thay đổi position với phương thức Transform.position (đây là phương thức có sẵn trong thành phần của các gameObject - ở đây là camera)

Từ bàn phím khi nhập:

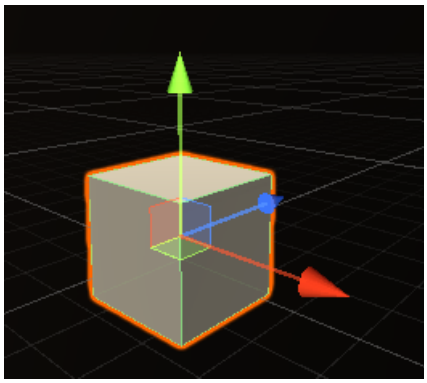
- Phím W: di chuyển camera đi thẳng
- Phím S: di chuyển camera đi lùi
- Phím A: di chuyển camera đi qua trái
- Phím D: di chuyển camera đi qua phải
- Phím R: trở lại vị trí ban đầu

Từ chuột máy tính:

- Nhấn giữ chuột phải + di chuột qua nửa bên trái màn hình: xoay camera qua trái
- Nhấn giữ chuột phải + di chuột qua nửa bên phải màn hình: xoay camera qua phải

2.6 Hiển thị các lớp và tập ảnh mỗi lớp

Trong không gian ba chiều của Unity, sử dụng hai mô hình có thể kết xuất đồ họa 3D là khối lập phương (hình 12) và SpriteRenderer (mô hình hỗ trợ kết xuất sprite trong môi trường ba chiều của Unity).



Hình 12. Khối lập phương trong Unity

Hai mô hình trên sẽ được định nghĩa sẵn và lưu lại dưới dạng Prefab. Ta có thể khởi tạo các prefab khi chương trình đang chạy với giao thức Instantiate, phương thức của các GameObject trong Unity.

Thông số của hai prefab được tạo sẵn gồm: prefab cho khối lập phương có thông số vị trí là vector (0, 0, 0), thông số quay là (0, 0, 0), thông số tỉ lệ là (100, 100, 33), prefab cho SpriteRenderer cube có thông số vị trí là vector (0, 0, 0), thông số quay là (0, 0, 0), thông số tỉ lệ là (1, 1, 1). Do SpriteRenderer sẽ được định nghĩa lại khi gán sprite vào trước khi chiết xuất lên màn hình, nên thông số của nó là ba vector mặc định.

Với Prefab có sẵn đó và số lượng layer của mô hình nơ ron máy học đã nhận được từ phía server, ta sẽ lần lượt tạo ra các GameObject cho từng layer, sử dụng các prefab khối lập phương. Lúc này các khối lập phương sẽ thay đổi vector vị trí tương ứng, khối cho lớp đầu tiên sẽ là (0, 0, 5), khối cho lớp tiếp theo là (0, 0, 160), các khối sẽ có khoảng cách là 150 đơn vị theo trục Z.

Đồng thời các tập ảnh của các lớp sẽ được tính toán vị trí và lưu vào một bảng đồ tọa độ ngay lúc này bằng cách tính chỉ số cạnh bằng căn bậc hai của tổng số hình trong một lớp, làm tròn số thành số nguyên gần nhất nhỏ hơn giá trị hiện tại. Cho khoảng cách của các ảnh sẽ là 30 đơn vị. Tập ảnh sẽ được kết xuất đúng với kích thước ảnh với một pixel (đơn vị một điểm ảnh) trên một unit (đơn vị nhỏ nhất của Unity). Với khoảng cách, kích thước ảnh, và số cạnh đã tính trước đó, ta tính vị trí của ảnh góc trên cùng bên tay trái. Do ta đang dùng vector vị trí (0, 0, 0) là góc. Cho nên ta chia số chỉ số cạnh cho hai, nhân nó với tổng của khoảng cách và kích thước của một hình, ta được tọa độ điểm cao nhất ở góc trên cùng bên tay trái. Cộng tiếp cho tổng của khoảng cách và kích thước một hình ta được tọa độ điểm góc trên cùng tay trái của hình tiếp theo, đến khi chỉ số hình chia hết cho chỉ số cạnh ta sẽ đổi chỉ số về chỉ số góc cao nhất trên cùng tay trái và đổi chỉ số tọa độ y (cộng thêm tổng khoảng cách và kích thước một hình).

Để dễ dàng cho quá trình quan sát, khi muốn tương tác với một lớp riêng biệt. Sự kiện nhấp chuột vào khối lập phương của lớp đó sẽ kết xuất tất cả bản đồ tính năng

(feature maps) hay hiện tại là các hình của lớp đó. Các hình sẽ được hiển thị lên trên của prefab spriterenderer đã được cài đặt trước đó. Các spriterenderer được khởi tạo bằng giao thức Instantiate và gán sprite là các bản đồ tính năng.

Các sprite này sau khi được chuyển đổi từ bytes sang texture sẽ được chuyển đổi thành sprite, do texture là dùng để hiển thị ảnh trong không gian hai chiều, lúc này ta chuyển thành sprite sẽ có thể kết xuất ảnh trong không gian ba chiều. Như đã nói ở trên sprite sau khi được gán vào spriterenderer vẫn sẽ giữ kích thước ban đầu với tỷ lệ một pixel trên một unit Unity. Khi các spriterenderer được tạo trở thành gameObject, gameObject biểu trưng cho lớp đó sẽ được tắt đi bằng cách chỉnh sửa thuộc tính enable thành false. Các spriterenderer sẽ được hiển thị dựa theo bảng đồ vector tọa độ đã được tính toán ở trên. Các spriterenderer sau khi được kết xuất sẽ được gán thuộc tính gồm có màu sắc là lần lượt có màu xanh nhạt (cyan), màu vàng, màu đỏ, màu xanh lá, và xanh dương. Với màu đỏ, xanh lá, xanh dương tượng trưng cho ba kênh màu của hình gốc. Màu vàng tượng trưng cho lớp tích chập, giúp phân biệt với các lớp khác, do đây là mô hình nơ ron tích chập nên lớp tích chập sẽ đóng vai trò quan trọng. Tiếp đến là màu xanh nhạt cho các lớp còn lại. Sử dụng màu vàng và xanh nhạt kết hợp với các ảnh đã được đảo ngược sẽ giúp dễ quan sát các điểm ảnh trong môi trường tối. Thêm vào đó, là thông số alpha cho màu, mỗi ảnh khi vừa được kết xuất sẽ mặc định có thông số alpha là 0.2 giúp ảnh mờ hơn, chống chồng chéo màu lên nhau khiến khó quan sát. Các ảnh được người dùng di chuột vào quan sát sẽ đổi thông số alpha trở thành 1 để giúp nổi bật các chi tiết ảnh hơn với nền và các ảnh xung quanh.

CHƯƠNG 3

KIỂM THỬ THỰC NGHIỆM

1. Kết quả kiểm tra

1.1 Giao diện

Giao diện chính của ứng dụng khi chưa kết nối với server như hình 13

Hình 13. Giao diện chính

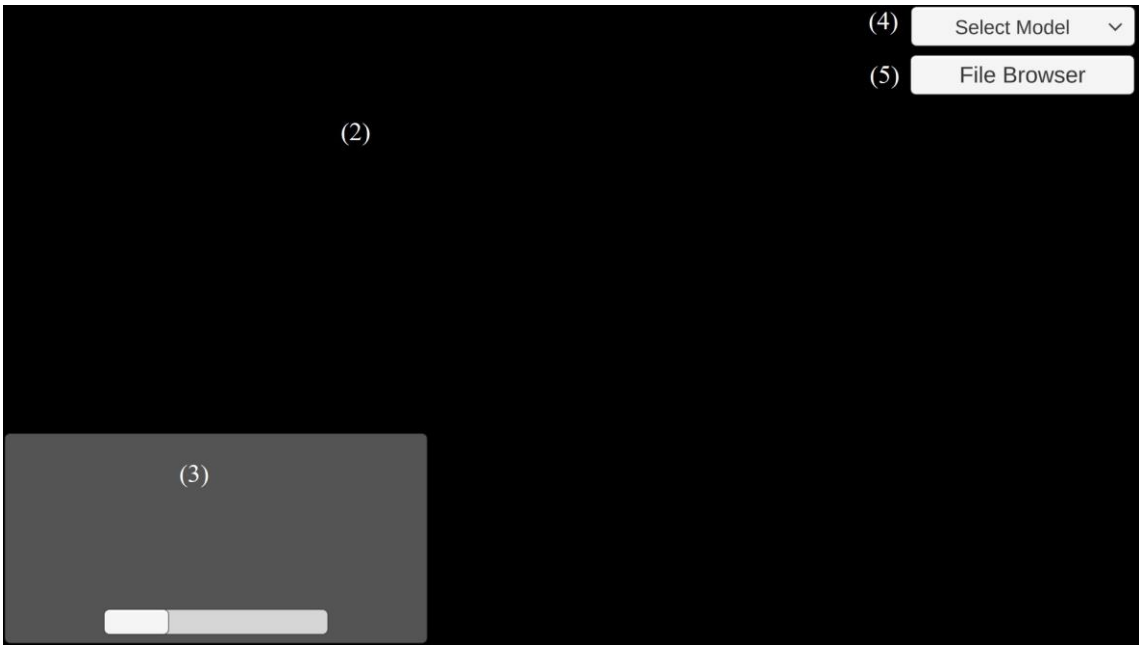
Giao diện gồm các thành phần:

(1) Nút kết nối đến server, sau khi nhấn nút này, giao diện người dùng sẽ kết nối đến server

(2) Giao diện hiển thị mô hình

(3) Bảng đồ nhỏ, nơi hiển thị các lớp có trong mô hình, có tích hợp một thanh trượt ngang giúp hiển thị các mô hình máy học có nhiều lớp.

Sau khi tiến hành kết nối đến server, giao diện sẽ có dạng như hình 14

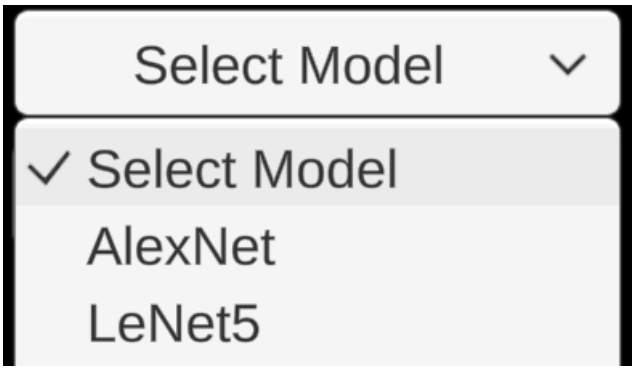


Hình 14. Giao diện sau khi kết nối với server

Giao diện có thêm các thành phần:

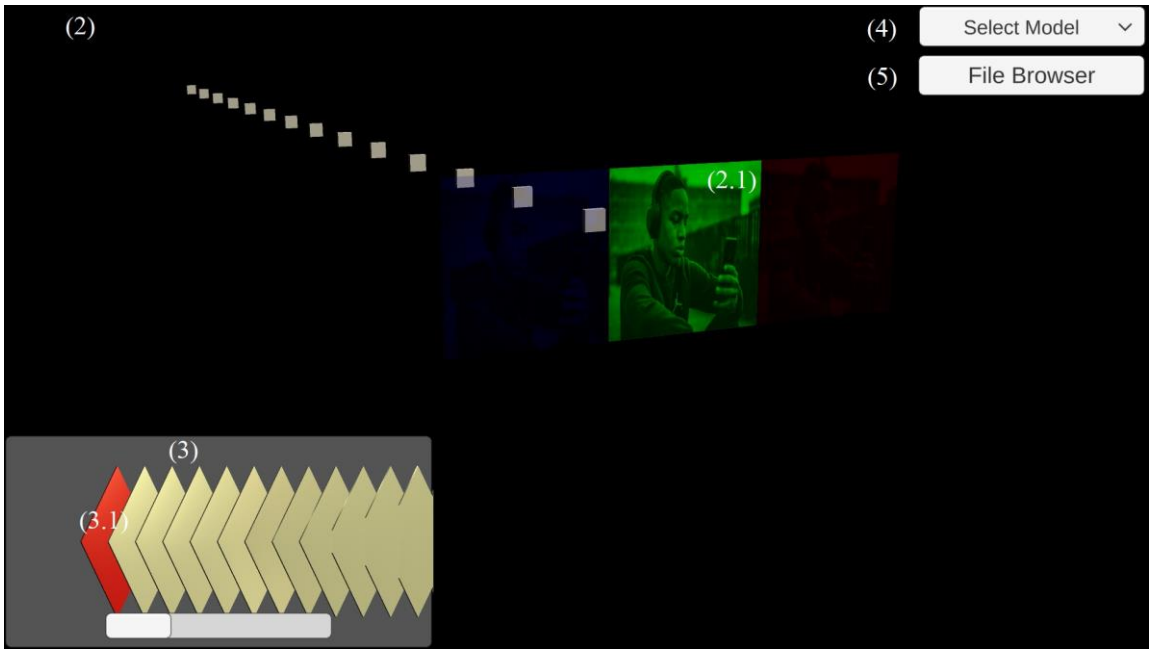
(4) Nút chọn các mô hình có sẵn, như hình 15

(5) Nút chọn mô hình từ thư mục, người dùng có thể chọn mô hình có sẵn từ thư mục



Hình 15. Các mô hình có sẵn

Sau khi chọn một mô hình bất kỳ ta sẽ có giao diện như hình 16



Hình 16. Giao diện sau khi chọn mô hình

Như hình 16 ta thấy ở mục (3) và (2) sẽ xuất hiện thêm các vật thể

- Ở (2) ta thấy dãy các khối vuông nhỏ trải dài, đó là các khối biểu tượng cho mỗi lớp
- Ở (3) ta thấy dãy các hình vuông trải ngang qua, đó cũng là các hình biểu tượng cho các lớp, lý do là khi ta di chuột vào một hình ở một lớp nào đó ở (2) thì hình biểu tượng cho lớp đó ở (3) sẽ đổi sang màu đỏ
- Ở những nơi được đánh số:

(2.1) Hình ảnh của lớp đầu tiên đang được hiển thị, đây là hệ quả sau khi nhấp chuột vào khối vuông

(3.1) Hình vuông đầu tiên đang màu đỏ khác với các hình vuông còn lại, do người dùng đang di chuột vào hình màu xanh lá ở (2.1)

1.2 Kiểm thử

1.2.1 Kiểm thử kết nối client với server

Do ứng dụng cần thiết đầu tiên là phải kết nối giữa giao diện người dùng (client) và phần xử lý mô hình (server) nên cần thiết kiểm thử khả năng kết nối giữa client và server, kiểm thử thời gian kết nối, khả năng kết nối và khả năng kết nối lại sau khi tắt client.

Kiểm thử thời gian kết nối, sử dụng Time.realtimeSinceStartup của Unity tính toán từ lúc ấn phím đến lúc hoàn thành kết nối

Lần thử	Thời gian kết nối (giây)
1	0.005559444
2	0.005220413
3	0.006093502
4	0.003484488

Bảng 1. Kiểm thử thời gian kết nối

Thời gian kết nối là quá nhỏ, lâu nhất trong 4 lần thử chỉ có 0,6 miliseconds

Kiểm thử khả năng kết nối, thực hiện kết nối 100 lần

Số lần kết nối	Số lần kết nối thành công	Tỷ lệ thành công
100	100	100%

Bảng 2. Kiểm thử khả năng kết nối

Thiết lập kết nối thành công hoàn toàn

Kiểm thử khả năng kết nối lại sau khi tắt client

Số lần kết nối lại sau lần thứ nhất	Số lần kết nối thành công	Tỷ lệ thành công
100	100	100%

Bảng 3. Kiểm thử khả năng kết nối lại

1.2.2 Kiểm thử truyền tải và chiết suất dữ liệu

Để nhận biết được khả năng truyền tải và chiết suất dữ liệu từ server. Cần thiết phải kiểm thử các khả năng như thời gian truyền tải và chiết suất, khả năng truyền tải sau mỗi lần kết nối, và khả năng truyền tải sau khi đã truyền tải và ngắt kết nối

Kiểm thử thời gian truyền tải và chiết suất, sử dụng Time.realtimeSinceStartup của Unity tính toán từ lúc ấn phím chọn mô hình đến lúc hoàn toàn hiển thị hết các lớp, sẽ sử dụng mô hình AlexNet cho đồng bộ

Lần thử	Thời gian hoàn thành (giây)
1	19.093

2	21.912
3	17.854
4	20.575

Bảng 4. Kiểm thử thời gian truyền tải và chiết suất

Thời gian nhiều nhất cần để truyền tải và chiết suất mô hình AlexNet qua 4 lần kiểm thử là 21,912 giây. Thời gian giữa các lần thử là có chênh lệch, với sự chênh lệch lớn nhất là 4 giây.

Kiểm thử khả năng truyền tải và chiết suất, thực hiện 100 lần

Số lần thực hiện	Số lần thành công	Tỷ lệ thành công
100	100	100%

Bảng 5. Kiểm thử khả năng truyền tải và chiết suất

Truyền tải và chiết suất thành công hoàn toàn

Kiểm thử khả năng truyền tải và chiết suất sau khi đã truyền tải và ngắt kết nối, thực hiện 100 lần

Số lần truyền tải sau lần thứ nhất	Số lần thành công	Tỷ lệ thành công
100	100	100%

Bảng 6. Kiểm thử khả năng truyền tải và chiết suất sau khi kết nối lại

Truyền tải và chiết suất thành công hoàn toàn

1.2.3 Kiểm thử tương tác

Ứng dụng là ứng dụng có giao diện đồ họa, có tích hợp khả năng chuyển động camera và khả năng tương tác với các vật thể trong không gian ba chiều, điều đó sẽ cần được kiểm thử để chứng minh.

Kiểm thử các nút bấm trên giao diện người dùng, gồm có 3 nút bấm và một thanh trượt ngang

Giao diện điều khiển	Số lần thực hiện	Số lần thành công	Tỷ lệ thành công
Nút kết nối	100	100	100%

Nút chọn mô hình	100	100	100%
Nút chọn tệp	100	100	100%
Thanh trượt ngang	100	100	100%

Bảng 7. Kiểm thử giao diện

Số lần thành công là 100%

Kiểm thử sự kiện điều khiển camera, bao gồm di chuyển bằng 4 phím điều hướng và xoay camera với chuột

Sự kiện	Số lần thực hiện	Số lần thành công	Tỷ lệ thành công
Phím W	100	100	100%
Phím S	100	100	100%
Phím A	100	100	100%
Phím D	100	100	100%
Nhấn giữ chuột phải + di chuột lên	100	100	100%
Nhấn giữ chuột phải + di chuột xuống	100	100	100%
Nhấn giữ chuột phải + di chuột qua trái	100	100	100%
Nhấn giữ chuột phải + di chuột phải	100	100	100%

Bảng 8. Kiểm thử sự kiện camera

Số lần thành công là 100%

Kiểm thử khả năng tương tác với các vật thể trong không gian ba chiều, gồm nhấp chuột vào cube để hiển thị tập ảnh của lớp tương ứng, di chuột vào ảnh của lớp xem hình bản đồ nhỏ có đổi màu đỏ

Sự kiện	Số lần thực hiện	Số lần thành công	Tỷ lệ thành công
Nhấn vào cube	100	100	100%
Di chuột vào ảnh	100	100	100%

Bảng 9. Kiểm thử tương tác vật thể ba chiều

Số lần thành công là 100%

1.2.4 Kiểm thử chức năng chọn mô hình

Chương trình cho phép chọn mô hình có sẵn trong server và mô hình người dùng đưa vào, cần phải kiểm thử chức năng chọn tệp và chọn mô hình.

Kiểm thử khả năng chọn tệp và chọn mô hình, ở đây sẽ chọn một tệp mẫu đã được luyện và lưu dạng SaveModel và chọn mô hình có sẵn là LeNet5

	Số lần thực hiện	Số lần thành công	Tỷ lệ thành công
Chọn SaveModel	100	100	100%
Chọn LeNet5	100	100	100%

Bảng 10. Kiểm thử chức năng chọn mô hình

1.3 Đánh giá kết quả kiểm thử

Kết quả kiểm thử qua các kiểm thử kết nối, kiểm thử truyền tải, kiểm thử tương tác, kiểm thử chọn mô hình nhìn chụp mang lại cái nhìn tổng quan hơn về các chức năng mà chương trình cung cấp. Các kiểm thử thời gian sử dụng phương thức thời gian thức của Unity mang lại thông số tương đối chính xác. Các kiểm thử về khả năng tương tác hay các chức năng luôn mang lại số lần thành công là 100%, lý do ở đây chính là chương trình sử dụng công cụ đúng đắn là Unity, đây là công cụ nổi tiếng trong phát triển game và các ứng dụng khác về đồ họa.

Đây là kết quả kiểm thử chủ quan không thông qua quy chuẩn nhất định nên không đảm bảo. Đều là kiểm thử bằng tay chứ không thông qua ứng dụng kiểm thử tự động nên không mang lại kết quả chính xác nhất.

PHẦN KẾT LUẬN

1. Kết quả đạt được

- Ứng dụng đã có thể được sử dụng để trực quan hóa các mô hình máy học tích chập phổ biến và các mô hình máy học tích chập do người dùng nhập vào
- Với khả năng được hỗ trợ bởi công cụ Unity, các mô hình đã được hiển thị và tương tác tự do trong môi trường ba chiều
- Sự kết nối thông qua giao thức TCP giúp cho ứng dụng có thể chạy riêng lẻ và dễ dàng cho việc cập nhật và phát triển.
- Hình ảnh thông qua xử lý của mô hình đã được hiển thị từng ảnh trên ứng dụng một cách phân biệt giúp nhận biết sự thay đổi của ảnh qua các lớp của mô hình máy học
- Ứng dụng cơ bản đã có thể giúp những người mới tiếp xúc với máy học, mạng nơ ron có cái nhìn cơ bản và hiểu biết và mạng nơ ron.
- Hạn chế: Ứng dụng chỉ ở mức cơ bản để hiển thị trực quan các mô hình nơ ron cơ bản. Ứng dụng chưa hiển thị được các mô hình có định dạng khác nhau, chỉ mới hiển thị được các mạng tuần tự.

2. Hướng phát triển

Với sự phát triển riêng biệt từ ban đầu đã hướng ứng dụng đến với khả năng phát triển trong tương lai một cách mạnh mẽ từ cả mặt xử lý mô hình máy học và cả mặt trực quan. Do là riêng biệt nên có thể thử nghiệm các công cụ trực quan mạnh mẽ khác, chỉ cần một API thích hợp là có thể sử dụng lại chương trình. Với sự phát triển mạnh mẽ của các công cụ phát triển game ba chiều cũng như các công nghệ thực tế ảo, thực tế tăng cường sẽ giúp công cụ trực quan hóa trong tương lai có thể trực quan sinh động hơn.

Có thể phát triển ứng dụng giúp người dùng có thể tự do tạo nên mô hình nơ ron bằng cách kéo thả trong giao diện ba chiều hoặc thực tế ảo, sau đó lưu lại và cho ra mô hình máy học. Sau đó, có thể xây dựng thêm khả năng trực quan hóa quá trình huấn luyện mô hình.

TÀI LIỆU THAM KHẢO

1. Roza Dastres and Mohsen Soori (2021), Artificial Neural Network Systems
2. Mary E. Webb, Andrew, et al. (2020), Machine learning for human learners: opportunities, issues,tensions and threats
3. Nadine Meissler, Annika Wohlan, et al. (2019), Using Visualization of Convolutional NeuralNetworks in Virtual Reality for Machine Learning Newcomers
4. Keiron O'Shea1 and Ryan Nash (2015), An Introduction to Convolutional Neural Networks
5. Tensorspaceteam (2018), https://tensorspace.org/
6. Christoph Linse, Hammam Alshazly and Thomas Martinetz (2022), A walk in the black-box: 3D visualization of large neural networks in virtual reality
7. Svetlin Nakov, Veselin Kolev, et al. (2013), Fundamentals of computer programming with C#, page 18-19.
8. Akshit J. Dhruv, Reema Patel and Nishant Doshi (2020), Python: The Most Advanced Programming Language for Computer Science Applications.
9. Tobias Kahlert and Kay Giza (2016), Visual Studio Code Tips & Tricks Vol. 1.
10. Timea Bezdan and Nebojša Bačanin Džakula (2019), Convolutional Neural Network Layers And Architectures.
11. Wojciech Gomolka (2014), The concept of Sockets and basic Function Blocks for communication over Ethernet Part 2 TCP Server and TCP Client.
12. F Arifin, H Robbani, Tiara Nur Annisa and Muhammad Izzat Nor Ma'arof (2019), Variations in the Number of Layers and the Number of Neurons in Artificial Neural Networks: Case Study of PatternRecognition.
13. Trương Gia Huy (2019), Xây dựng phòng thay đồ thực tế ảo bằng công nghệ nhận dạng cử chỉ sử dụng Camera Kinect V2.
14. Martín Abadi, Paul Barham, Jianmin Chen, et al. (2016), TensorFlow: A system for large-scale machine learning.

15. Alexander Mordvintsev and Abid Rahman K. (2013), Introduction to OpenCV-Python Tutorials