ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

NGUYỄN TẤN THIÊN

ĐỒ ÁN 1

THIẾT KẾ GIẢI PHÁP NHẬN DẠNG CHÓ/MÈO SỬ DỤNG CONVOLUTION NEURAL NETWORK TRÊN PHẦN CỨNG

KỸ SƯ NGÀNH KỸ THUẬT MÁY TÍNH

TP.HỒ CHÍ MINH, 6/2022

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH

Nguyễn Tấn Thiên – 19522266

ĐỒ ÁN 1

THIẾT KẾ GIẢI PHÁP NHẬN DẠNG CHÓ/MÈO SỬ DỤNG CONVOLUTION NEURAL NETWORK TRÊN PHẦN CỨNG

KỸ SƯ NGÀNH KỸ THUẬT MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN

Th.s Phạm Thanh Hùng

TP. HỒ CHÍ MINH, 6/2022

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

TÓM TẮT BÁO CÁO ĐỒ ÁN

Trong đồ án này, nhóm tiến hành nghiên cứu về xây dựng mạng CNN trong ứng dụng nhận dạng chó/mèo trên FPGA. Trong đó, mạng sử dụng trọng số đã được huấn luyện trước trên phần mềm (keras, tensorflow) sau đó trích xuất các trọng số để nạp vào mạng.

Bên trong kiến trúc, nhóm sử dụng kiểu dữ liệu là dấu chấm động độ chính xác đơn.

MỞ ĐẦU

Ngày nay, Mạng nơ ron tích chập (Convolutional Neural Network) là một trong những thuật ngữ vô cùng quen thuộc trong lĩnh vực công nghệ thông tin và đặc biệt là trí tuệ nhântạo (AI). mạng nơ ron là một trong những mô hình Deep Learning vô cùng tiên tiến và được ứng dụng phổ biến nhất trong dữ liệu ảnh, trong đó nhận diện, phân loại là các bàit oán điển hình của mạng nơ ron tích chập.

Trong lĩnh vực phát triển phần cứng trên FPGA, các nghiên cứu và hiện thực chủ yếu vào một khối thành phần, khối chuyên dụng nào đó trong Mạng nơ ron tích chập hoặc thiết kế các mạng có kích thước nhỏ đa phần sử dụng kiểu dữ liệu Fixed Point. Điều này cũng dễ hiểu bởi sự hạn chế, giới hạn về mặt tài nguyên phần cứng của FPGA. Chính vì vậy, mục tiêu của nhóm muốn triển khai và hiện thực một mạng nơ ron đầy đủ các lớp theo như mô hình mạng đã huấn luyện, sử dụng kiểu dữ liệu chấm động có độ chính xác 32 Bit (floating point 32bit) theo chuẩn IEEE 754.

Báo cáo này sẽ gồm các phần sau được tổ chức như sau: Chương 1 Tổng quan đề tài. Chương 2 Tổng quan về Convolution Neural Network. Chương 3 Bài toán nhận diện Chương 4 Thiết kế. Chương 5 Mô phỏng, đánh giá.

Chương 6 Tổng kết lại những việc đã thực hiện được trong đồ án này

Chương 1. TỔNG QUAN ĐỀ TÀI

1.1 Nguồn gốc và sự phát triển.

Convolution Neural Network – CNN (Mạng nơ ron tích chập) hiện đang là một trong các đề tài được nghiên cứu và ứng dụng nhiều nhất trong lĩnh vực xử lý, nhận diện ảnh. Đây là mô hình Deep Learning hiện đại nhất ngày nay và ngày càng phát triển mạnh mẽ để đưa ra những kiến trúc tốt nhất, gần gũi nhất với con người. Mạng nơ-ron tích chập được lấy cảm hứng từ não bộ con người. Những nghiên cứu về lĩnh vực nơ-ron não bộ của D.H Hubel và T.N Wiesel trong những thập niên năm 1950 [1] và 1960 [2] trên não bộ của động vật đã đề xuất một mô hình cho việc động vật nhìn nhận thế giới. Trong báo cáo, hai ông đã diễn tả 2 loại tế bào trong não và cách hoạt động khác nhau là tế bào đơn giản (simple cell –S cell) và tế bào phức tạp (complex cell – C cell).

Năm 1980, dựa theo các khái niệm s-cell và c-cell, Fukushima đã đề xuất mô hình mạng nơ ron có cấp bậc gọi là "neocognitron" [3], trong nghiên cứu này, mạng neocognitron có thể nhận diện mẫu dựa trên việc học hình dáng của đối tượng. Sau đó, năm 1998 mạng nơ ron tích chập được giới thiệu bởi Bengio, Le Cun, Bottou và Haffner với tên gọi LeNet-5 lần đầu tiên được ứng dụng trong nhận diện chữ số viết tay. Với sự ảnh hưởng từ cuộc cách mạng 4.0 đã làm thay đổi chóng mặt về sự phát triển và ứng dụng của các mạng nơron tích chập và điển hình cho xu hướng ứng dụng CNN trong computer vision là mạng AlexNet vào năm 2012. Liên tiếp sau đó là hang loạt các kiến trúc mạng CNN ra đời với các cải tiến về độ sâu, cách thiết kế các block, kết nối giữa các block. Tiêu biểu như VGG Net (2014), GoogleNet (2014), ResNet (2015), DenseNet (2016), … Song song với các kiến trúc ngày càng phát triển và có độ chính xác cao hơn, số lượng layer ngày càng nhiều hơn thì số lượng các trọng số huấn luyện sẽ ngày càng tăng lên. Điều này sẽ đòi hỏi sự phát triển phần cứng máy tính như các GPU có tốc độ nhanh hơn để có thể giảm thiểu tối đa thời gian huấn luyện mạng.

1.2 Giới thiệu đề tài

Dưới sự chạy đua trong lĩnh vực thị giác máy tính thì convolution neural network –CNN (mạng nơ ron tích chập) đang là một trong những chủ đề được yêu thích và nghiên cứu nhiều nhất trong xử lý ảnh. Convolution neural network giúp chúng ta xây dựng được những hệ thống thông minh và có độ chính xác cao trong nhiều bài toàn như nhận diện, phân loại ảnh, phân tích video, …

Hiện nay các ứng dụng nhận diện, phân loại bằng CNN đang được nghiên cứu phát triển rất nhiều. Đối với các mô hình CNN nhận diện, phân loại với số lượng lớp lớn, Người ta thường lựa chọn GPU là phần cứng để huấn luyện và triển khai một mạng CNN cho các ứng dụng nhận diện, điều này sẽ có những hạn chế như chi phí quá cao, tiêu tốn nhiều năng lượng, ... Ngoài ra cũng có những nghiên cứu cho việc triển khai mạng CNN trên FPGA đem lại những tích cực hơn như tốn ít năng lượng, giảm chi phí hơn so với GPU nhưng bị giới hạn độ sâu của mạng và số lớp nhận diện. Đây được xem là một hạn chế và là bài toán hiện thực mạng CNN trên FPGA.

Bài toán trên xảy ra khi số lượng lớp nhận diện tăng và đòi hỏi độ chính xác cao trong nhận diện, phân loại hình ảnh sử dụng CNN. Điều này làm cho các mạng CNN có độ sâu lớn và số lượng layer nhiều và độ sâu của layer tăng lên, khi đó số lượng tham số sẽ tăng lên dẫn đến tài nguyên phần cứng sẽ tăng lên. Điều này đòi hỏi phải có những giải pháp phù hợp, tối ưu thiết kế để có thể đáp ứng được vấn đề tài nguyên cho việc triển khai, hiện thực mạng CNN có độ chính xác cao và nhận diện được nhiều lớp trên FPGA. Chính vì vậy, đề tài này muốn đưa ra giải pháp và triển khai, hiện thực một mạng CNN có số lượng tham số lớn trên FPGA cho ứng dụng nhận diện biển chó/mèo. Đề tài nhận diện ảnh có chó/mèo với số lượng tham số sau khi trích xuất từ quá trình huấn luyện là 2090 parameter.

1.3 Hướng nghiên cứu

Dựa trên một kiến trúc mạng CNN được xây dựng và đào tạo, huấn luyện trên một GPU server, nhóm sẽ nghiên cứu và đưa ra các giải pháp và hiện thực được mạng CNN này sao cho đáp ứng được các yêu cầu về phần cứng trên FPGA. Sau đó tiến hành mô phỏng.

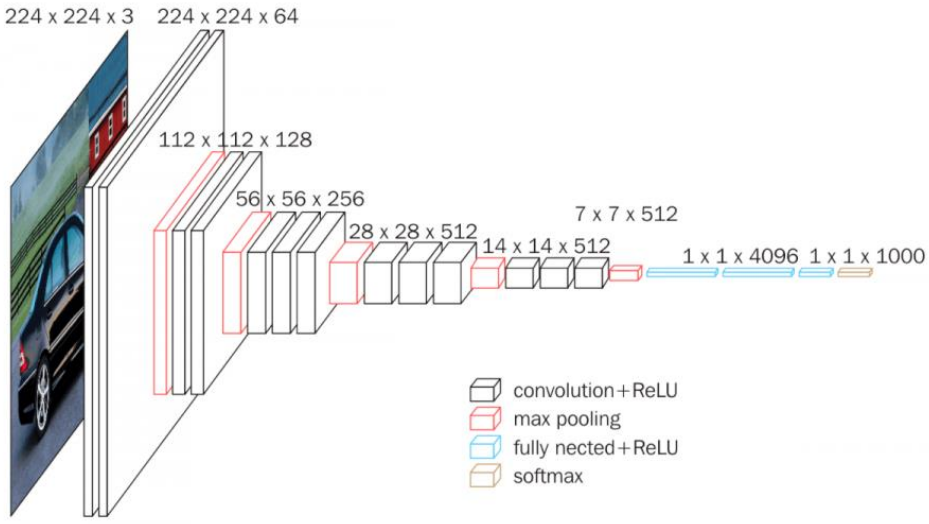
1.4 Mục tiêu đề ra

Trong đồ án này, nhóm đặt ra mục tiêu là thiết kế thành công mô hình mạng CNN. Hiện thực, mô phỏng thành công ứng dụng của thiết kế cho nhận diện ảnh có chó.

Chương 2. TỔNG QUAN VỀ CONVOLUTION NEURAL NETWORK

2.1 Convolution nerual network – mạng nơron tích chập.

Mạng nơron tích chập được lấy cảm hứng từ vỏ não thị giác. Mỗi khi chúng ta nhìn thấy một vật, một loạt các lớp tế bào thần kinh được kích hoạt, và mỗi lớp thần kinh sẽ phát hiện một tập hợp các đặc trưng như đường thẳng, cạnh, màu sắc, ... của đối tượng, lớp thần kinh càng cao sẽ phát hiện các đặc trưng phức tạp hơn để nhận ra những gì chúng ta đã thấy. Mạng nơron tích chập là mạng bao gồm một lớp đầu vào và một lớp đầu ra, ngoài ra, ở giữa còn có nhiều lớp ẩn (hidden layers). Các lớp ẩn của mạng thường bao gồm nhiều lớp tích chập (convolutional), lớp tổng hợp (pooling), lớp kết nối đầy đủ (Fully Connected) và theo sau các lớp ẩn là các hàm kích hoạt (ReLU, Softmax, …).



Hình 2-1. Kiến trúc mạng CNN VGG16

Các mạng nơron tích chập nổi bậc như là LeNet - một trong những mạng CNN lâu đời nổi tiếng nhất được Yann LeCun phát triển vào năm 1998, AlexNet - mạng CNN đã dành chiến thắng trong cuộc thi ImageNet LSVRC-2012 năm 2012 với large margin (15.3% VS 26.2% error rates), VGGNet - có tỉ lệ sai (error rate) nhỏ hơn AlexNet trong ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2014, ResNets phát triển bởi Microsoft vào năm 2015, thắng giải ImageNet ILSVRC competition 2015 với tỉ lệ sai 3.57%, …

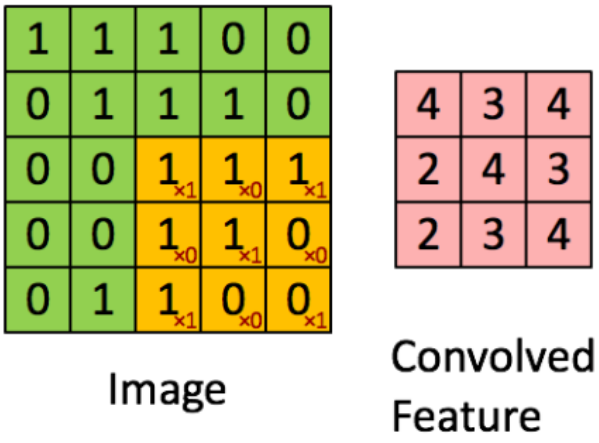
2.1.1 Convolutional Layer – Lớp tích chập

Mỗi lớp tích chập trong mạng nơron sẽ có đầu vào là một tensor với kích thước (chiều cao ảnh) x (chiều rộng ảnh) x (chiều sâu của ảnh). Lớp tích chập sẽ thực hiện nhân

chập với một kernel (ma trận lọc), chiều dài và rộng của kernel sẽ được chọn theo dạng tham số, còn chiều sâu thì phải bằng với chiều sâu của ảnh, sau đó, kết quả của lớp này sẽ được truyền sang lớp tiếp theo.

Để tính tích chập ta cần quan tâm đến hai thông số là Stride và Padding. Kernel sẽ được trượt qua ảnh, với mỗi lần trượt ta sẽ tính kết quả tại vị trí đầu ra. Số bước ở mỗi lần trượt kernel được gọi là Stride, thông thường, trong tích chập, Stride bằng 1. Nếu tích chập Same Padding, sẽ tạo các điểm ảnh bằng 0 ở rìa của ảnh, đặc trưng đầu ra sẽ có kích thước không đổi so với ảnh đầu vào. Còn Valid Padding, đặc trưng đầu ra sẽ giảm K - 1 so với ảnh đầu vào, với K là kích thước kernel.

Hình 2-2 là ví dụ về nhân tích chập 2 chiều trên ảnh kích thước 5x5, kích thước kernel là 3x3, stride bằng 1 và Valid Padding. Sau đó, đặc trưng thu được có kích thước 3x3.



Hình 2-2. Ví dụ về tích chập 2 chiều Stride 1, Valid Padding

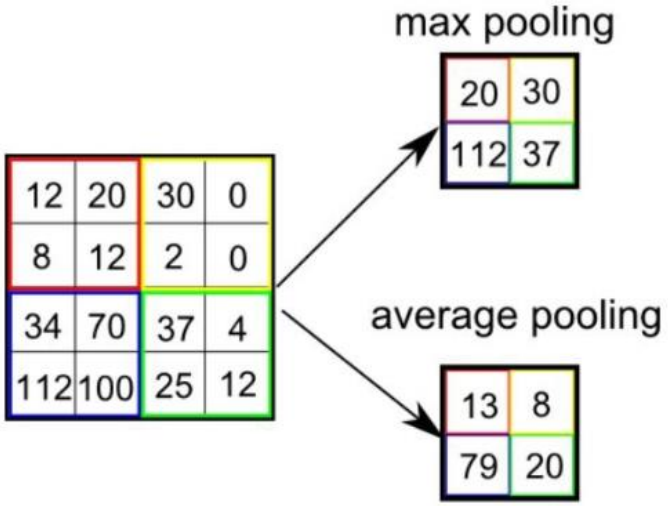
2.1.2 Pooling layer – lớp tổng hợp.

Lớp tổng hợp chịu trách nhiệm giảm kích thước không gian của đặc trưng sau khi tích chập. Nhờ đó, lớp tổng hợp giúp giảm số lượng tính toán của dữ liệu. Hơn nữa, lớp này rất hữu ích trong việc lọc ra những đặc trưng "trội", giúp cho model được huấn luyện tốt và dễ hơn. Lớp tổng hợp cũng có hai thông số Stride và Padding, Stride được định nghĩa giống với Stride ở lớp tích chập. Với Padding Same, kích thước đầu ra sẽ được xác định với công thức (2.1), với Padding Valid, kích thước đầu ra sẽ được xác định với công thức (2.2), gọi H, W là kích thước đầu vào, H', W' là kích thước đầu ra, K là kích thước của cửa sổ trượt.

$$\begin{cases} H' = \text{Ceil}(\frac{H}{K}) \\ W' = \text{Ceil}(\frac{W}{K}) \end{cases} \quad (2.1)$$

$$\begin{cases} H' = \frac{H}{K} \\ W' = \frac{W}{K} \end{cases} \quad (2.2)$$

Có hai loại Pooling là Max Pooling (trả về giá trị lớn nhất trong cửa sổ trượt) và Average Pooling (trả về giá trị trung bình trong cửa sổ trượt). Hình 2-3 là ví dụ của Max Pooling và Average Pooling với cửa sổ trượt kích thước 2x2, đầu vào có kích thước 4x4, stride bằng 2.



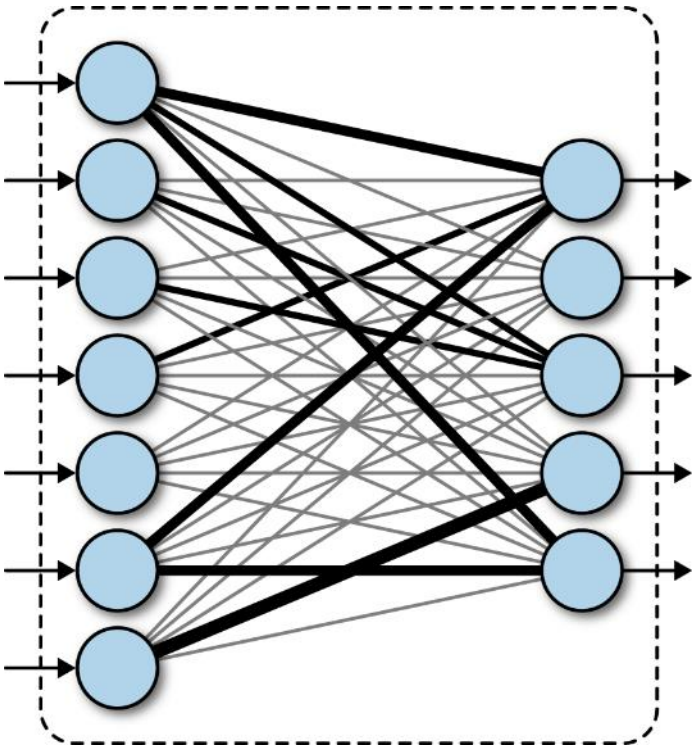
Hình 2-3. Lớp Pooling khi Stride bằng 2

2.1.3 Fully connected layer – lớp kết nối đầy đủ.

Sau khi thực hiện các lớp tích chập và tổng hợp, mạng CNN thường kết thúc bằng lớp kết nối đầy đủ - kết nối tất cả nốt của lớp này với tất cả nốt của lớp khác. Lớp này thường dùng trong mục đích để phân loại ảnh và để dữ liệu dữ đoán có thể hiểu được, hàm kích hoạt Softmax sẽ được áp dụng để đưa dữ liệu về dạng xác suất.

Đầu vào của lớp sẽ được "làm phẳng" thành ma trận một chiều, tất cả các nốt của ma trận sẽ được nhân với một trọng số để tạo ra một nốt đầu ra. Lớp này được tính bằng công thức (2.3), với N là số nốt đầu ra. Hình 2-4 mô tả lớp kết nối đầy đủ với 7 nốt đầu vào và 5 nốt đầu ra.

$$y_i = \sum_{k=0}^{W*H-1} x_k * w_k + b_i, 0 \leq i < N \quad (2.3)$$



Hình 2-4. Lớp kết nối đầy đủ với 7 nốt đầu vào và 5 nốt đầu ra.

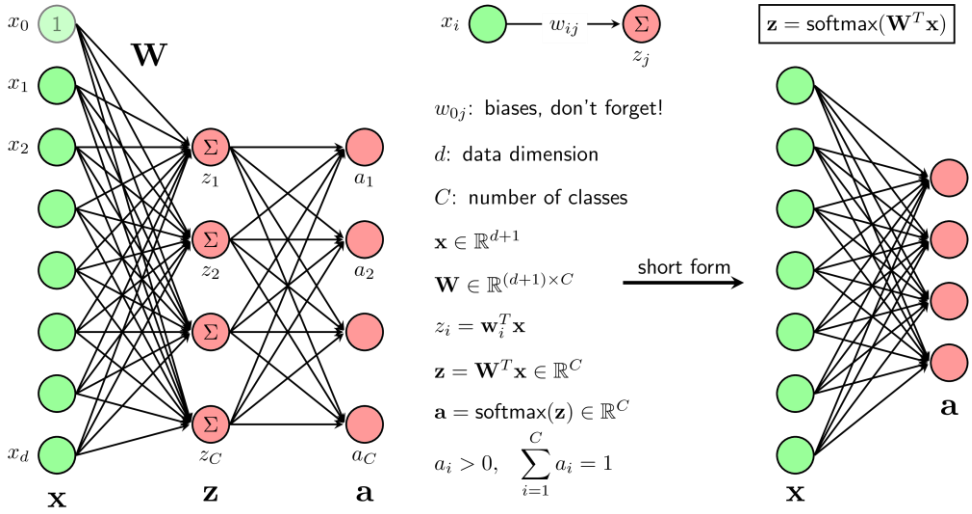
2.1.4 Các hàm kích hoạt thường dùng.

2.1.4.1 RELU

ReLU, đầy đủ là Rectified Linear Unit, loại bỏ các giá trị âm bằng cách đặt nó về 0, làm tăng các đặc trưng phi tuyến tính của mạng tổng thể mà không ảnh hưởng trực tiếp đến lớp tích chập. Công thức đầy đủ của hàm kích hoạt ReLU được thể hiện ở (2.4).

$$f(x) = \max(0, x) \quad (2.4)$$

2.1.4.2 Softmax

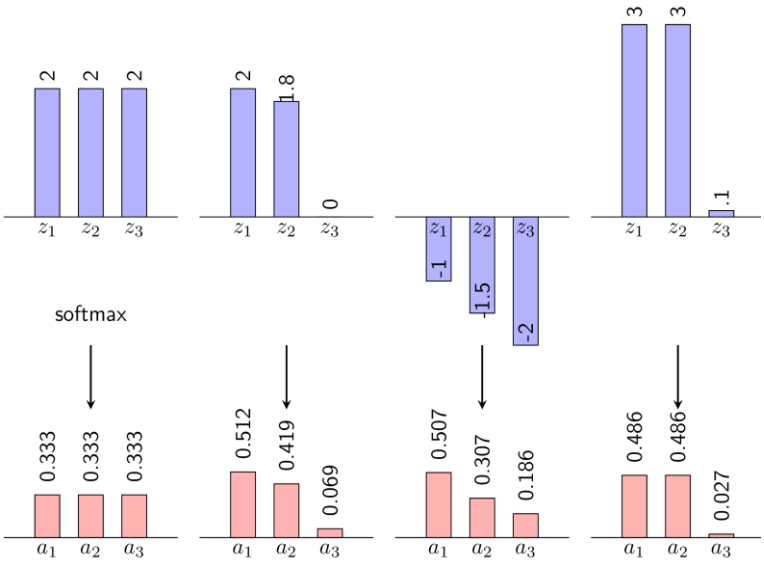


Hình 2-5. Hàm kích hoạt Softmax kết nối với lớp kết nối đầy đủ

Hàm kích hoạt Softmax thường được dùng sau khi thực hiện FC Layer, mục đích để giá trị được chuẩn hoá dưới dạng xác suất, Hình 2-5 mô tả lớp kết nối đầy đủ và theo sau là hàm kích hoạt Softmax.

Công thức tổng quát của hàm kích hoạt Softmax, với z_i là kết quả đề cập ở trên, $a_i$là xác suất dự đoán của từng lớp được thể hiện như công thức (2.5).

$$a_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \forall i = 1,2,\dots,C \qquad (2.5)$$



Hình 2-6. Ví dụ về kết quả của hàm kích hoạt Softmax

Với những đầu vào z_i bằng nhau, xác suất sẽ bằng nhau, z_i càng lớn sẽ cho xác suất càng lớn, z_i càng bé sẽ cho xác suất càng bé. Hình 2-6 minh hoạ cơ bản ví dụ về Softmax.

2.2 Dấu chấm động – Floating Point

2.2.1 Định nghĩa.

Trong tin học, dấu chấm động được dùng để chỉ một hệ thống biểu diễn số mà trong đó sử dụng một chuỗi chữ số (hay bit) để biểu diễn một số hữu tỉ.

Số dấu chấm động là giải pháp thông dụng nhất để biểu diễn số thực trong kí hiệu khoa học.

Biểu diễn các số dưới dạng: $X=M*R^E$

- M là phần định trị (Mantissa)
- R là cơ số (Radix)
- E là phần mũ (Exponent)

Ví dụ: 69.0 có thể biểu diễn là $6.9*10^1$ trong hệ số thập phân.

2.2.2 Chuẩn dấu chấm động IEEE754

Hiệp hội IEEE đã chuẩn hóa cho việc biểu diễn số dấu phẩy động nhị phân trong máy tính bằng cách đưa ra chuẩn IEEE 754. Ngày nay hầu hết các máy tính đều tuân thủ theo chuẩn này. Một số trường hợp ngoại lệ như máy tính lớn IBM và máy vector Cray. Loại máy tính lớn IBM ngoài định dạng thập phân và nhị phân IEEE 754 còn có một định dạng riêng của IBM. Còn với máy vector Cray thì họ T90 có một phiên bản IEEE nhưng máy SV1 vẫn còn dùng định dạng dấu phẩy động của chính Cray. Chuẩn IEEE 754 đưa ra nhiều định dạng rất gần nhau, chỉ khác nhau ở một ít chi tiết. Năm trong số những định dạng này được gọi là định dạng cơ bản, và hai trong chúng đặc biệt được dùng rộng rãi trong cả phần cứng máy tính và ngôn ngữ lập trình:

- Độ chính xác đơn, được gọi bằng tên là "float" trong họ ngôn ngữ lập trình C và tên là "real" hay "real*4" trong ngôn ngữ Fortran. Đây là định dạng nhị phân chiếm 32 bits (4 bytes) và phần định trị của nó có độ chính xác 24 bit (tương đương với khoảng 7 chữ số thập phân).
- Độ chính xác kép, được gọi bằng tên là "double" trong họ ngôn ngữ lập trình C và tên là "double precision" hay "real*8" trong ngôn ngữ

Fortran. Đây là định dạng nhị phân chiếm 64 bit (8 byte) và phần định trị của nó có độ chính xác 53 bit (tương đương với khoảng 16 chữ số thập phân).

Các định dạng khác là nhị phân với độ chính xác bậc bốn (128 bit), cũng như là dấu phẩy động thập phân (64 bit) và dấu phẩy động thập phân "kép" (128 bit).

Các định dạng ít thông dụng hơn:

- Định dạng độ chính xác mở rộng, mỗi số chiếm 80 bit.
- Định dạng bán chính xác cũng gọi là dấu phẩy động 16, mỗi số chiếm 16 bit.

Bất kỳ một số nguyên nào có giá trị tuyệt đối nhỏ hơn hay bằng 2^{24} đều có thể biểu diễn một cách chính xác bằng định dạng độ chính xác đơn, và bất kỳ số nguyên nào có giá trị tuyệt đối nhỏ hơn hay bằng 2^{53} cũng có thể biểu diễn một cách chính xác bằng định dạng độ chính xác kép.

Mặc dầu các định dạng 32 bit ("đơn") và 64 bit ("kép") hiện nay là phổ biến, nhưng chuẩn IEEE 754 cũng cho phép nhiều mức chính xác khác nhau. Lấy ví dụ, các phần cứng máy tính như họ Pentium Intel và họ 68000 Motorola thường có thêm định dạng độ chính xác mở rộng 80 bit, với phần mũ 15 bit, phần định trị 64 bit (không có bit ẩn) và 1 bit dấu. Một dự án nhằm mục đích sửa đổi chuẩn IEEE 754 đã được khởi động trong năm 2000 (xem IEEE 754 sửa đổi. Dự án này đã hoàn thành và được công nhận vào tháng 6 năm 2008. Nó bao gồm các định dạng dấu phẩy động thập phân và định dạng dấu phẩy động 16 bit ("nữa"). Định dạng 16 bit nhị phân có cùng cấu trúc và quy luật như các định dạng cũ khác với 1 bit dấu, phần mũ 5 bit và 10 bit phần định trị. Định dạng này hiện đang được sử dụng trong ngôn ngữ đồ họa Cg của NVIDIA, và có mặt trong chuẩn mở EXR. [4]

2.2.2.1 Cấu trúc biểu diễn trong máy tính.

Thông thường thì các số dấu phẩy động được thể hiện trong bộ nhớ máy tính theo thứ tự từ trái sang phải gồm bit dấu, phần mũ, rồi đến phần định trị. Với định dạng nhị phân IEEE 754 chúng thường được biểu diễn bằng các phần sau trong Bảng 2-1.

Bảng 2-1. Các kiểu dấu chấm động

Kiểu	Dấu	Phần mũ	Phần định trị	Tổng số bit
Nửa	1	5	10	16
Đơn	1	8	23	32
Kép	1	11	52	64
Bậc bốn	1	15	112	128

2.2.2.2 Ưu điểm, nhược điểm.

Ưu điểm lớn nhất của dấu chấm động là nó biểu diễn được tầm giá trị rộng hơn nhiều so với các kiểu biểu diễn khác.

Định dạng dấu chấm động sẽ mất chiếm nhiều bộ nhớ hơn so với các định dạng khác. Các phép toán trên dấu chấm động phức tạp hơn so với các cách định dạng khác.

Chương 3. BÀI TOÁN NHẬN DIỆN

Như đã trình bày trong các phần trước về ứng dụng rộng rãi nhất của CNN là phân loại và nhận diện ảnh.

Trong đồ án này, nhóm chọn đề tài là nhận dạng chó/mèo (2 đầu ra) sử dụng tập dữ liệu cifar10. Ngày nay có nhiều mô hình nhận dạng nhưng đa phần được triển khai trên phần mền.

Nhóm muốn áp dụng một mạng CNN đơn giản gồm 4 layer được xây dựng bởi ngôn ngữ mô tả phần cứng cho nhận dạng chó/mèo.

Mô hình mà nhóm lựa chọn gồm 4 layer trong đó có 3 convolution layer và một fully connected layer. Đầu vào là ảnh có kích thức 32x32x3. Bảng 3-1 sau đây sẽ thống kê lại số layer của mạng CNN này

Bảng 3-1. Tổng quan thông số của mạng CNN.

Layer Name	Input size	Output size	Kernel/number	Parameter
Convolution layer1	32x32x3	16x16x4	3x3x3/4	112
Convolution layer2	16x16x4	8x8x8	3x3x4/8	296
Convolution layer3	8x8x8	4x4x16	3x3x8/16	1168
Fully connected	256	2		514
Total parameter				2090

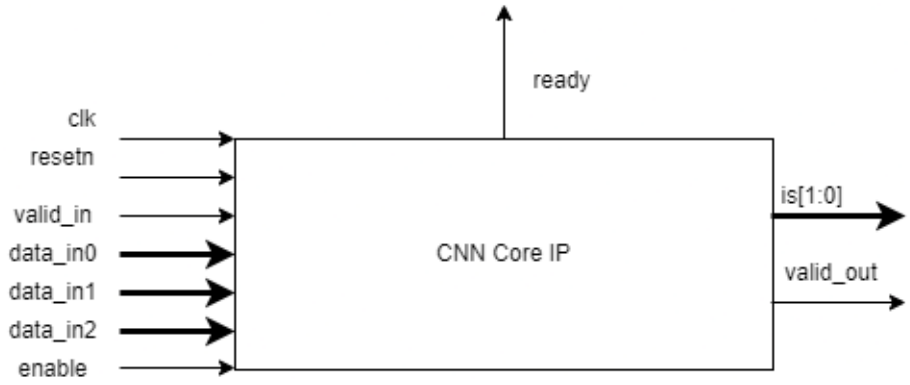
Chương 4. THIẾT KẾ [5]

4.1 Thiết kế tổng quan

4.1.1 Interface IP core.

CNN core IP có chức năng là nhận dạng ảnh, với đầu vào là pixel của ảnh và đầu ra là nhãn dự đoán ở dạng one-hot. Ví dụ trong khi đưa vào 1 một ảnh bất kì mô hình sẽ tính toán và đưa ra dư đoán đầu vào này thuộc lớp 0(ảnh chứa chó) thì đầu ra của

IP sẽ ở dạng nhị phân 01 và ngược lại nếu thuộc lớp 1(ảnh không chứa chó) thì là 10.



Hình 4-1. Interface của CNN Core IP

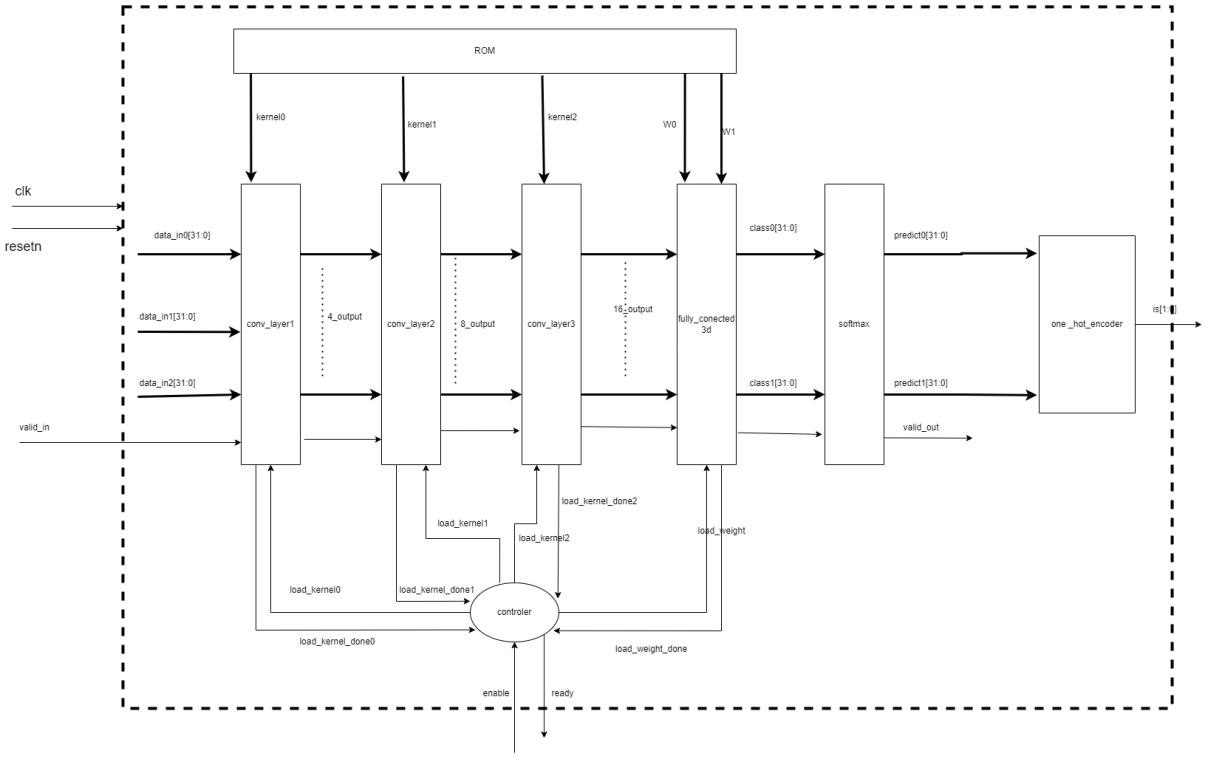
Bảng 4-1 dưới đây sẽ mô tả các tín hiệu của core IP.

Bảng 4-1. Mô tả tín hiệu interface CNN Core IP

Tín hiệu	**I/O**	**Số bit**	**Mô tả**
clk	I	1	Xung clock hệ thống, kích cạnh lên.
resetn	I	1	Reset bất đồng bộ tích cực thấp.
valid_in	I	1	Tín hiệu valid in bằng 1 khi dữ liệu vào đúng.
data_in0/data_in1/ data_in2	I	32	Dữ liệu đầu vào là các pixel được đọc từ ảnh và lưu vào file .txt
enable	I	1	Khởi động hệ thống và bắt đầu nạp trọng số từ ROM.
ready	O	1	Cho biết trọng số đã nạp xong, sẵn sàng nhận dữ liệu.
valid_out	O	1	Tín hiệu valid out bằng 1 khi dữ liệu ra đúng.
is	O	2	Nhãn của ảnh input, mã hoá dưới dạng one-hot,

4.1.2 Sơ đồ khối thiết kế.

CNN Core IP được chia làm hai thành phần chính, một là mạng nhận diện ảnh và trả về nhãn dự đoán ảnh, hai là module điều khiển chức năng nạp trọng số vào mạng CNN. Mục đích của việc chia làm hai thành phần như vậy giúp cho thay đổi trọng số đã huấn luyện đơn giản hơn, chỉ cần thay đổi tệp chứa các trọng số. Việc làm này vô tình làm cho thiết kế phức tạp hơn ở quá trình nạp trọng số vào các thành phần của mạng như Convolution và Fully Connected Layer. Hình 4-2 mô tả sơ đồ khối CNN Core IP, các tín hiệu sử dụng trong thiết kế và Bảng 4-2 mô tả chi tiết chức năng từng khối trong thiết kế khối CNN Core IP.



Hình 4-2. Sơ đồ khối CNN Core IP

Bảng 4-2. Mô tả các tín hiệu của CNN core IP

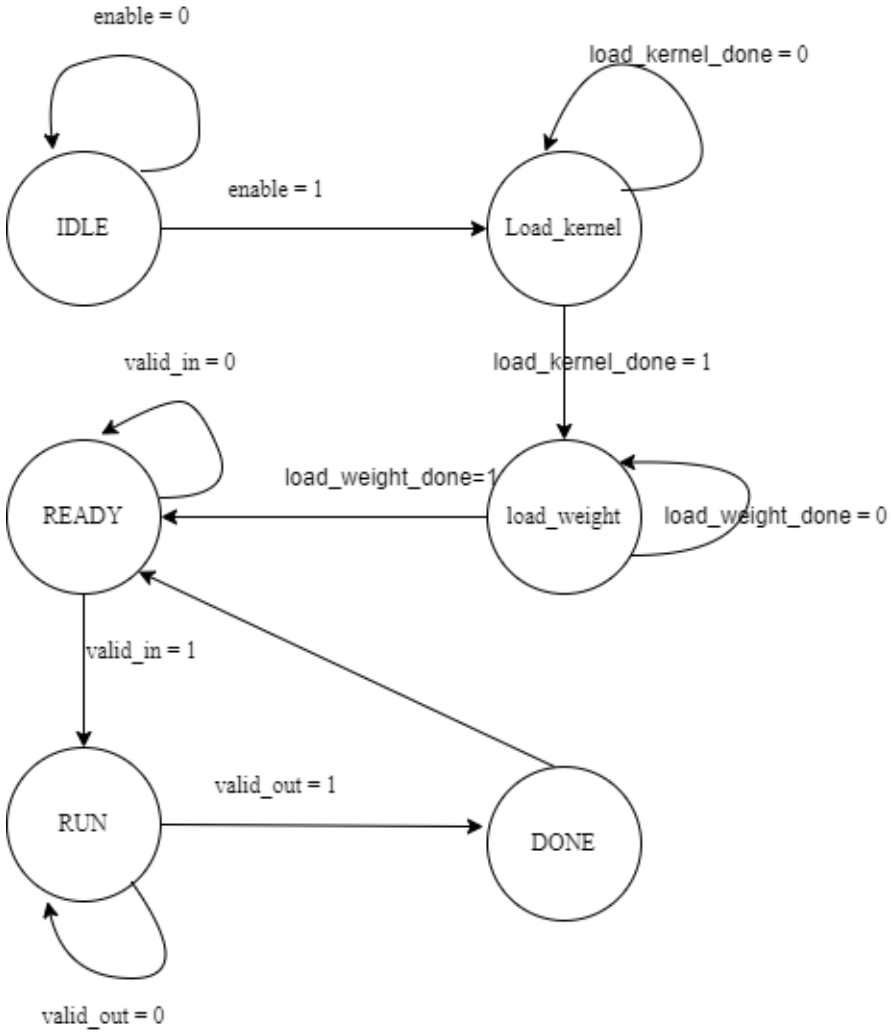
Tín hiệu	I/O	Số bit	Mô tả
clk	I	1	Xung clock hệ thống, kích cạnh lên.
resetn	I	1	Reset bất đồng bộ tích cực thấp.
valid_in	I	1	Tín hiệu valid in bằng 1 khi dữ liệu vào đúng.

data_in0/data_in1/ data_in2	I	32	Dữ liệu đầu vào là các pixel được đọc từ ảnh và lưu vào file .txt
enable	I	1	Khởi động hệ thống và bắt đầu nạp trọng số từ ROM.
ready	O	1	Cho biết trọng số đã nạp xong, sẵn sàng nhận dữ liệu.
valid_out	O	1	Tín hiệu valid out bằng 1 khi dữ liệu ra đúng.
is	O	2	Nhãn của ảnh input, mã hoá dưới dạng one-hot,

4.1.3 Sơ đồ chuyển trạng thái.

Trong lúc khởi tạo một mạng CNN, chúng ta sẽ mất một khoảng thời gian để nạp trọng số vào mạng. Do đó, khi khởi động module, các trọng số của kernel (bộ lọc) và Fully Connected Layer sẽ được nạp. Quá trình nạp trọng số diễn ra nhanh hay chậm sẽ được quyết định do số lượng trọng số vì trong mỗi chu kỳ, chỉ một trọng số được nạp vào trong mạng. Thứ tự trọng số của lớp Convolution nạp vào mạng sẽ tuần tự từ trái sang phải, từ trên xuống dưới và chiều sâu tăng dần (đối với trường hợp sử dụng nhiều bộ lọc).

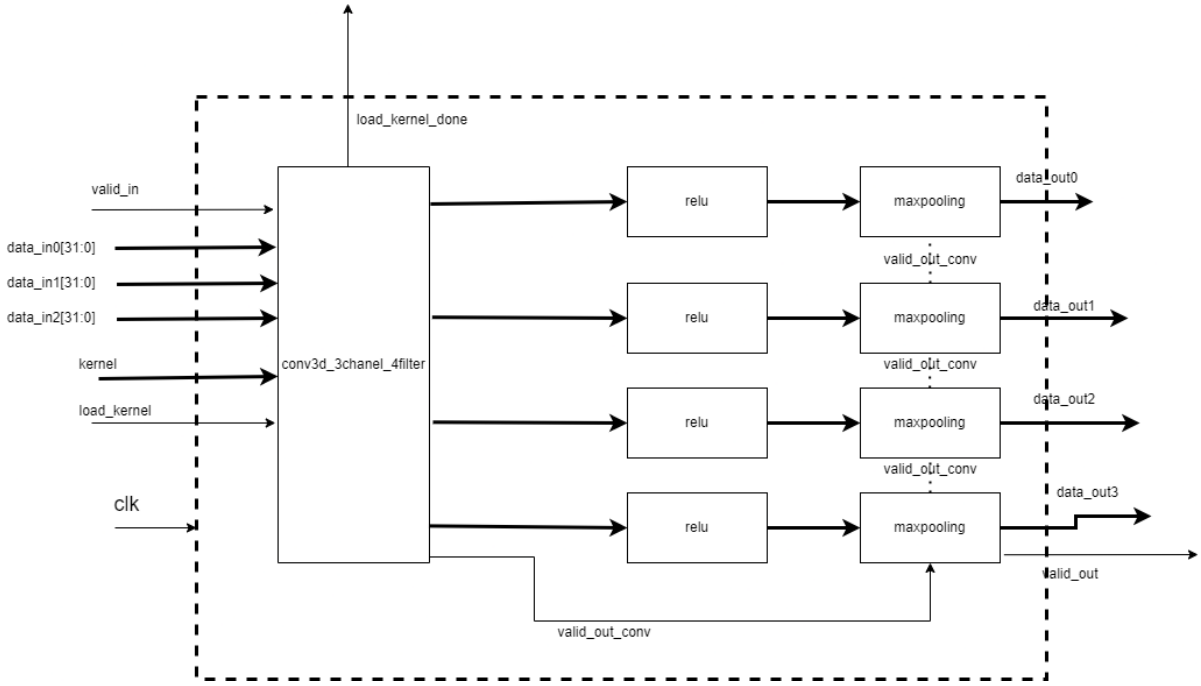
Khi nạp xong trọng số, module sẽ ở trạng thái “Ready”, lúc này module sẽ chờ dữ liệu ảnh nạp vào mạng. Dữ liệu ảnh là binary 32 bit. Hình 4-3 mô tả chi tiết quá trình và các tín hiệu chuyển đổi trạng thái.



Hình 4-3. Sơ đồ chuyển trạng thái.

4.2 Khối convolution layer

Mạng CNN của nhóm có tất cả 3 khói Convolution layer1, Convlution layer2, Convolution layer3 (Hình 4-2 Mục 4.1.2). Cả 3 khối đều là khối tích chập 3 chiều trên bộ lọc có số lượng lần lượt là 4, 8, 16. Trong phần này nhóm sẽ trình bày thiết kế của khối convoluton layer1 (với thông số đầu vào đầu ra và kernel đã trình bày ở mục chương 3 bảng 3-1. Hình 4-4 là sơ đồ khối của convolution layer 1 và Bảng 4-3 mô tả các tín hiệu của convolution layer 1.



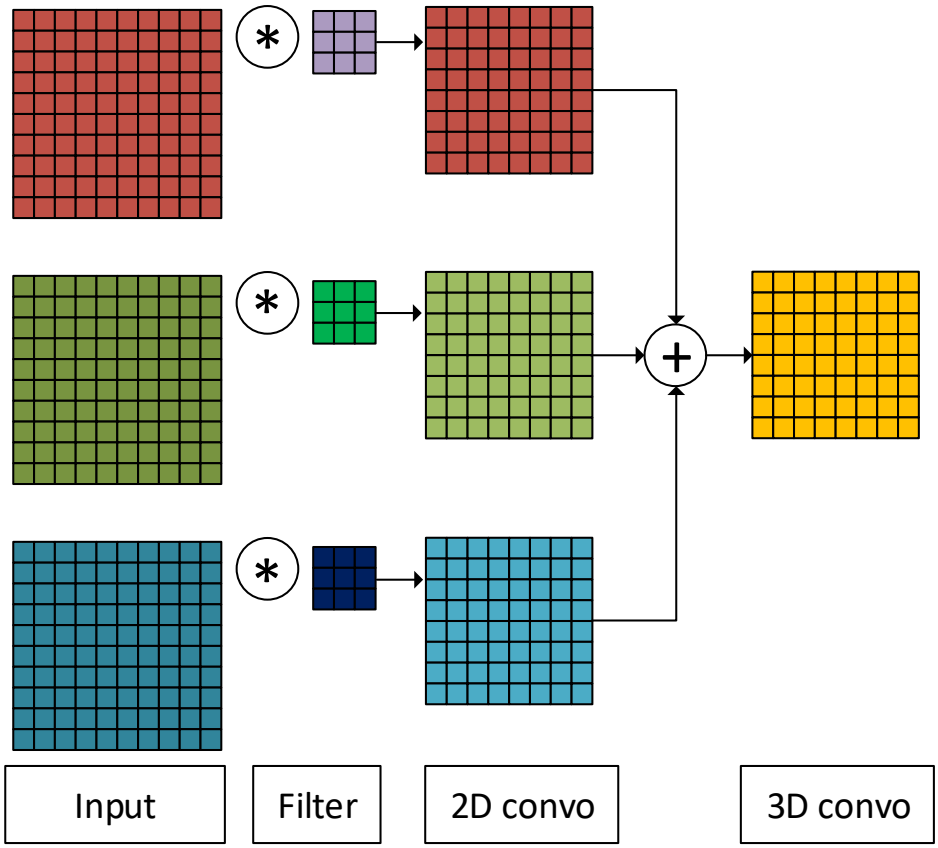
Hình 4-4. Sơ đồ khối của convolution layer1

Bảng 4-3. Mô tả các tín hiệu của convolution layer1

Tín hiệu	**I/O**	**Số bit**	**Mô tả**
clk	I	1	Xung clock hệ thống, kích cạnh lên.
resetn	I	1	Reset bất đồng bộ tích cực thấp.
valid_in	I	1	Tín hiệu valid in bằng 1 khi dữ liệu vào đúng.
data_in0/data_in1/ data_in2	I	32	Dữ liệu đầu vào là các pixel được đọc từ ảnh và lưu vào file .txt
Load_kernel	I	1	Tín hiệu load_kernel bằng 1 khi đang nạp kernel vào mô đun.
Kernel	I	32	Truyền kernel vào mô đun
valid_out	O	1	Tín hiệu valid out bằng 1 khi dữ liệu ra đúng.
Load_kernel_done	O	1	Tín hiệu cho biết đã hoành thành việc nạp kernel vào mô đun.
Data_out0/data_out1 /data_out2	O	32	Dữ liệu đầu ra của mô dun.

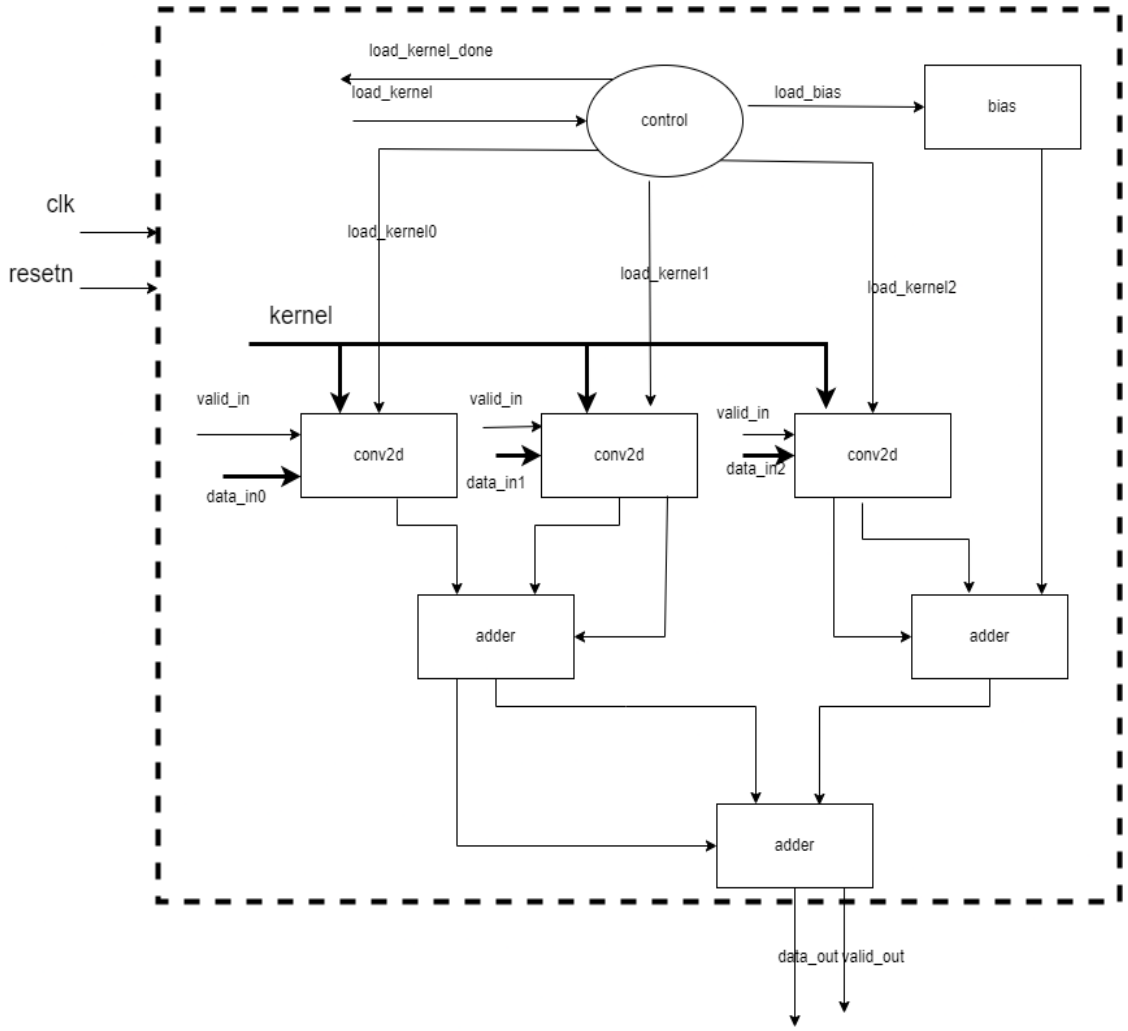
4.2.1 Khối convolution 3D

Tương tự với tích chập 2 chiều, tích chập 3 chiều là khi bộ có thêm cả chiều sâu bằng với chiều sâu của ảnh. Hình 4-5 là ví dụ của nhân tích chập 3 chiều, với đầu vào có kích thước bằng 10x10x3, và bộ lọc có kích thước bằng 3x3x3 (chiều sâu của bộ lọc phải bằng với chiều sâu của đầu vào), kết quả là tổng của kết quả tích chập 2 chiều, đầu ra có kích thước 8x8x1.



Hình 4-5. Ví dụ nhân tích chập 3 chiều trên một bộ lọc

Dựa trên ý tưởng đã nêu ở phần trên, chúng ta có thiết kế như Hình 4-6, giả sử đầu vào có chiều sâu bằng 3, chức năng của từng khối được mô tả trong Bảng 4-4.

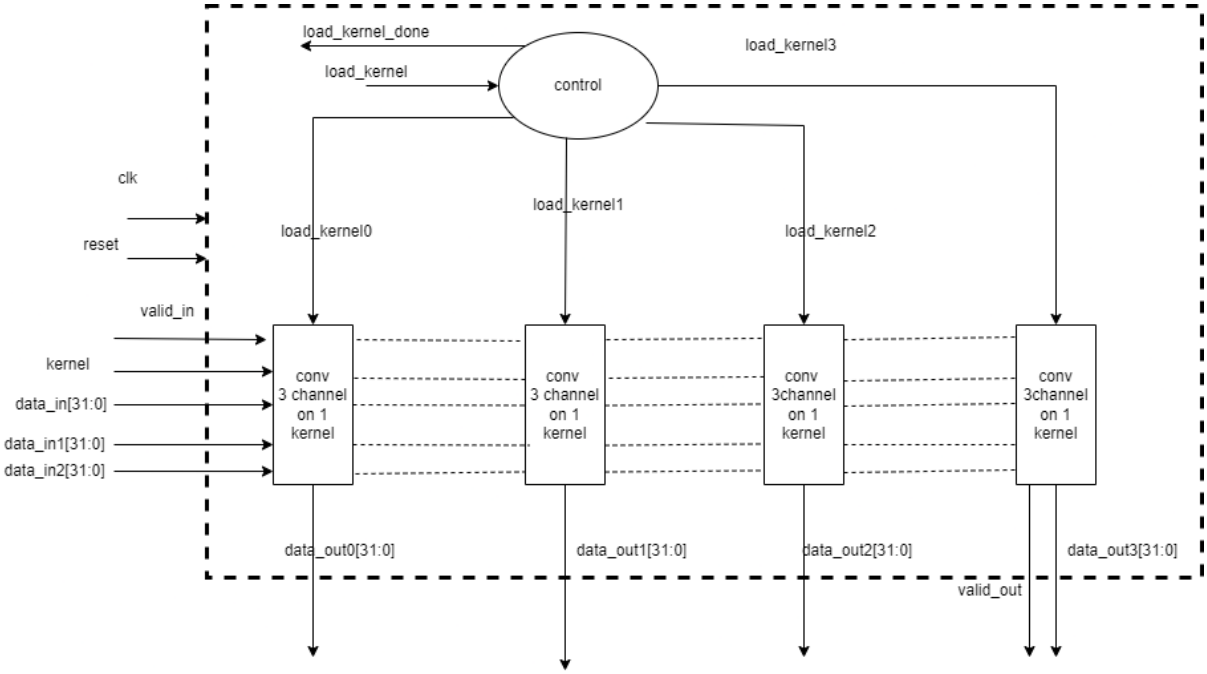


Hình 4-6. Sơ đồ khối của convolution 3D (3 channel , 1 kernel)

Bảng 4-4. Mô tả các khối trong Convolution 3D trên một bộ lọc

Khối	**Mô tả**
Conv2D	Nhân tích chập 2 chiều.
control	Khối điều khiển tín hiệu nạp trọng số vào mạng.
bias	Nơi lưu trữ các bias.
adder	Bộ cộng dấu chấm động.

Thông thường, phép tính tích chập không chỉ có một bộ lọc, mà có thể có nhiều bộ lọc. Trong thiết kế phần cứng, việc này được thực hiện khá đơn giản do khả năng xử lý song song. Hình 4-7 thể hiện bộ nhân tích chập 3 chiều trên 4 bộ lọc, kích thước đầu ra sẽ có chiều sâu bằng 4, chức năng các khối bên trong bộ tích chập 3 chiều được mô tả chi tiết trong Bảng 4-5.



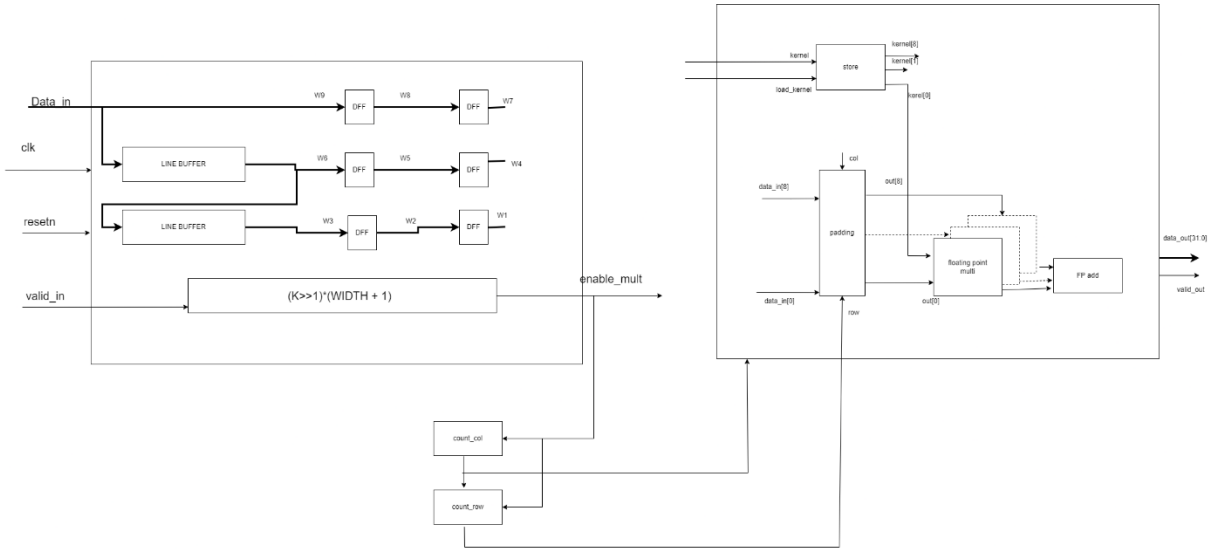
Hình 4-7. Sơ đồ khối Convolution 3D (3 channel, 4 kernel)

Bảng 4-5. Mô tả các khối của convolution 3D(3 channel, 4 kernel.

Khối	**Mô tả**
Conv 3channel on 1 kernel	Nhân tích chập 3 chiều trên 1 bộ lọc với channel là 3.
Control	Khối điều khiển tín hiệu nạp trọng số vào mạng.

4.2.2 Convolution 2D 3x3

Convolution 2D 3x3 là một trong những khối chính của thiết kế convolutional layer nói chung và mạng CNN nói riêng cho việc trích xuất đặc trưng.



Hình 4-8. Sơ đồ khối của convolution 2D 3x3.

Nhóm sử dụng line buffer dùng để trích xuất ma trận 3x3 trong nhân tích chập. Sau đó ma trận này sẽ được đi qua khối padding để áp dụng padding dạng "same" để đạt để đạt được kích thước đầu ra tương tự như model training bằng phần mềm.

Những khối count_col và count_row dùng để đếm số hàng và cột của ma trận 3x3 để áp dụng padding.

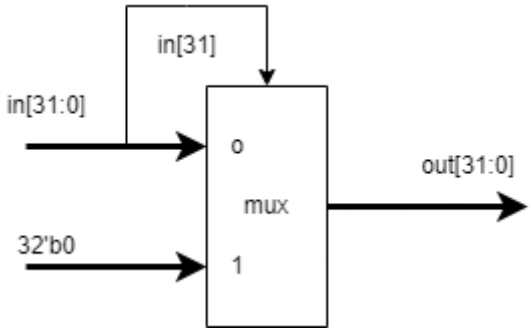
Khối store dùng để lưu trữ kernel.

Bảng 4-6. Mô tả tín hiệu của convolution 2D 3x3

Tín hiệu	**I/O**	**Số bit**	**Mô tả**
clk	I	1	Xung clock hệ thống, kích cạnh lên.
resetn	I	1	Reset bất đồng bộ tích cực thấp.
valid_in	I	1	Tín hiệu valid in bằng 1 khi dữ liệu vào đúng.
data_in	I	32	Dữ liệu đầu vào là các pixel được đọc từ ảnh và lưu vào file .txt
Load_kernel	I	1	Tín hiệu load_kernel bằng 1 khi đang nạp kernel vào mô đun.
Kernel	I	32	Truyền kernel vào mô đun
valid_out	O	1	Tín hiệu valid out bằng 1 khi dữ liệu ra đúng.
Data_out	O	32	Dữ liệu đầu ra của mô dun.

4.2.3 Thiết kế khối RELU

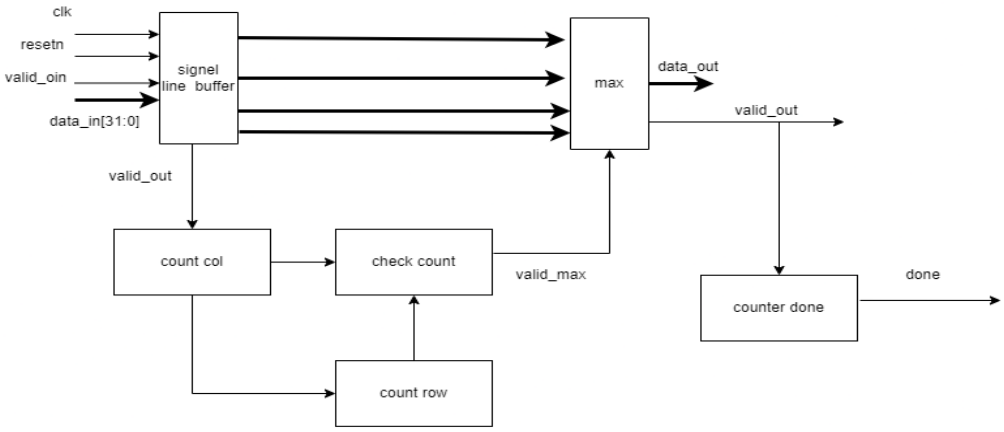
Module trả về giá trị lớn nhất của 0 và dữ liệu đầu vào. Nhờ vào MSB của dấu chấm động, chúng ta có thể xác định được dữ liệu là số âm nếu MSB bằng 1. Vì thế, ReLU được thiết kế như module sử dụng cổng MUX 2-1, với select là MSB của dữ liệu đầu vào. Nếu MSB bằng 1, module sẽ trả về 32’b0. Ngược lại, nếu MSB bằng 0, module sẽ trả về chính đầu vào. Hình 4-9 mô tả sơ đồ khối chi tiết của module ReLU Activation.



Hình 4-9. khối RELU

4.2.4 Thiết kế khối Max Pooling

Trong thiết kế này, chúng ta cố định kích thước của ma trận Pooling là 2x2, và chỉ sử dụng Max Pooling với bước nhảy bằng 2. Tức là, với đầu vào có kích thước HxWxD, thì đầu ra sẽ có kích thước H/2xW/2xD.



Hình 4-10. Sơ đồ khối của MaxPooling

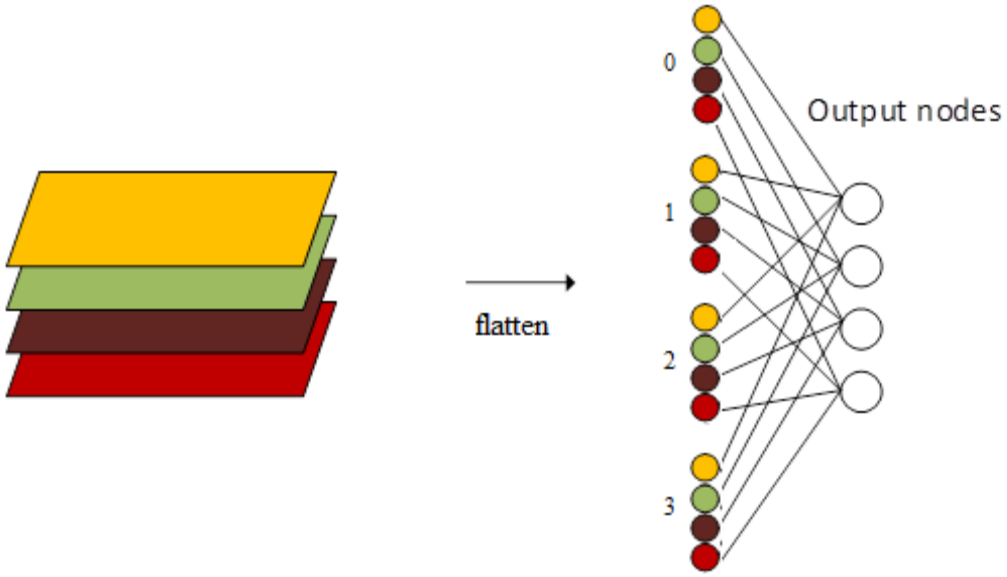
Giống với convolution 2D, Maxpooling 2x2 cũng trích xuất ma trận cùng với kỹ thuật line buffer ta có thể trích xuất ma trận 2x2 nhưng ở maxpooling thì các ma trận không được “đè” lên nhau, do đó module cần phải kiểm tra vị trí hiện tại của cửa sổ trượt để chắc chắn không bị đè lên nhau, sau đó ta so sánh 4 giá trị đó để tìm ra giá trị lớn nhất.

Bảng 4-7. Mô tả tín hiệu của MaxPoling

Tín hiệu	I/O	Số bit	Mô tả
clk	I	1	Xung clock hệ thống, kích cạnh lên.
resetn	I	1	Reset bất đồng bộ tích cực thấp.
valid_in	I	1	Tín hiệu valid in bằng 1 khi dữ liệu vào đúng.
data_in	I	32	Dữ liệu đầu vào là các pixel được đọc từ ảnh và lưu vào file .txt
valid_out	O	1	Tín hiệu valid out bằng 1 khi dữ liệu ra đúng.
Data_out	O	32	Dữ liệu đầu ra của mô dun.
done	O	1	Tín hiệu cho biết đã cho ra đủ số lượng đầu ra cần thiết.

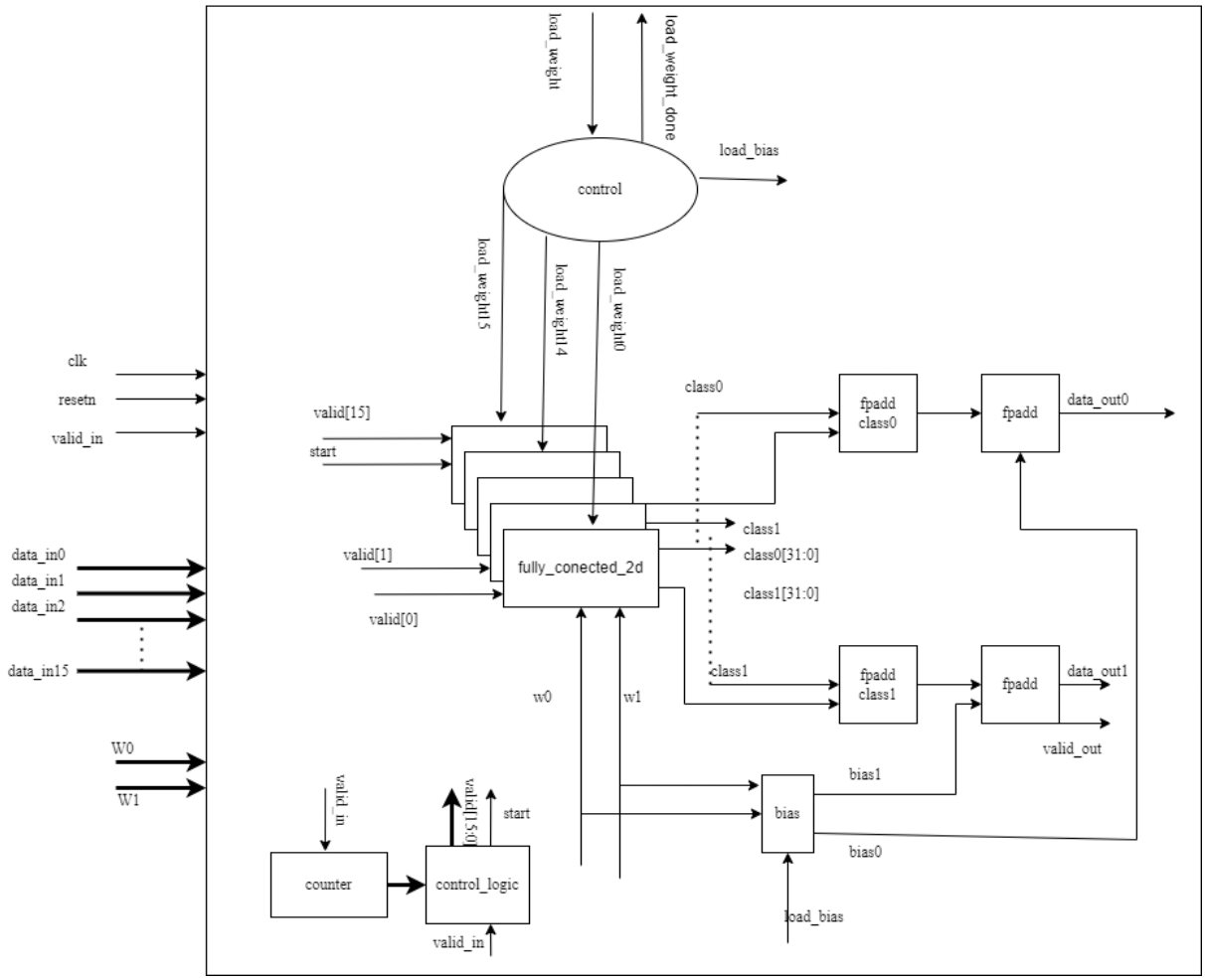
4.3 Thiết kế khối fully connected layer 3D

Với trường hợp dữ liệu có kích thước HxWxD và D > 1, ta sẽ chia dữ liệu thành HxW ma trận 1 chiều D và thực hiện nhân ma trận trên HxW ma trận đó. Lúc này chúng ta sẽ có HxWxN nốt đầu ra. Kết quả đầu ra sẽ là N nốt, mỗi nốt là tổng của HxW nốt tại vị trí đó. Mô tả cụ thể tại hình .



Hình 4-11. Mô tả 3D fully connected

Với ý tưởng trên ta thiết kế được như hình dưới đây với D = 16, H = 4, W = 4 và N =2.



Hình 4-12. Sơ đồ khối của fully connected 3D

Bảng 4-8. Mô tả các khối của fully connected 3D.

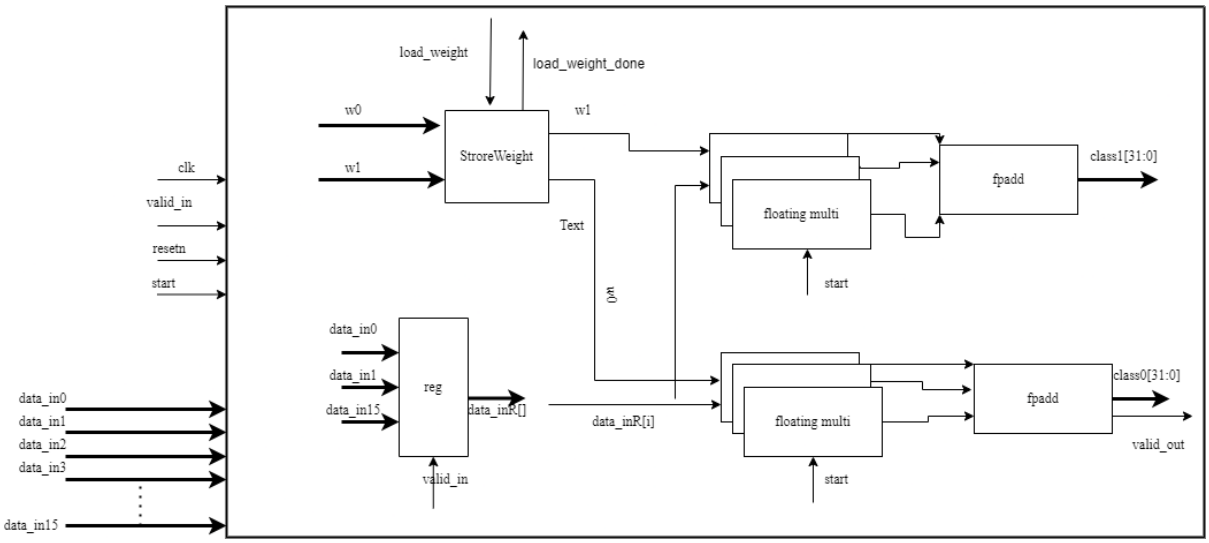
Khối	**Mô tả**
Fully_connected_2d	Thực hiện nhân ma trận 1 chiều.
Control	Khối điều khiển tín hiệu nạp trọng số vào mạng.
fpadd	Khối thực hiện cộng số floating point.
counter	Đếm thứ tự của ngõ vào.
Control_logic	Với từng kết của của counter thì ta tạo ra được giá trị valid[15:0] sao cho tương ứng để khối fully_connected_2d lưu dữ liệu data_in vào và chờ tín hiệu start để bắt đầu tính toán.

4.3.1 Thiết kế khối fully connected 2d

Module thực hiện chức năng nhân đặc trưng đầu vào với N ma trận một chiều, với N là số nốt đầu ra, đầu ra của module là dữ liệu của N nốt.

Khi tín hiệu valid_in bằng 1 thì ta lần lượt nhân 16 giá trị của của data_in với 16 giá trị weight đã nạp vào từ trước (trọng số được lưu vào storeWeight). Rồi ta tính tổng của 16 giá trị kết quả của phép nhân thì sẽ ra được kết quả của 1 nốt đầu ra.

Tương tự nhu trên ta tính được kết quả của nốt thứ 2.

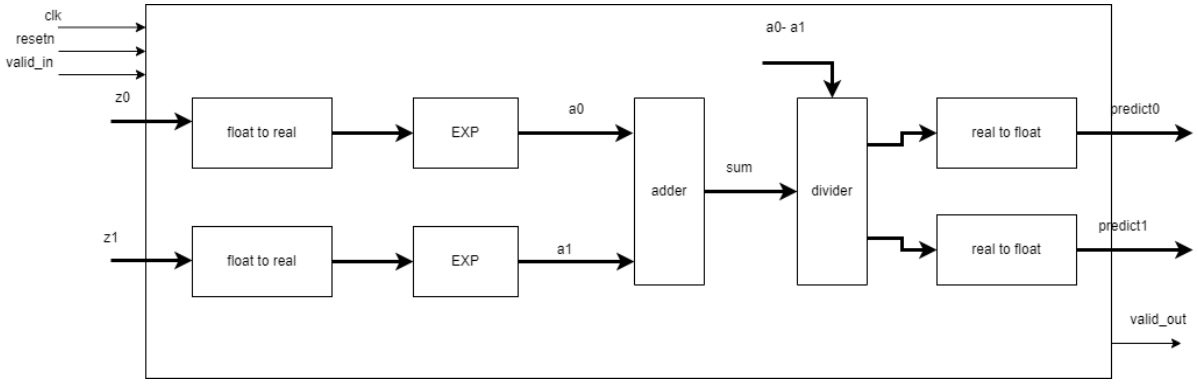


Hình 4-13. Sơ đồ khối của fully connected 2D

Bảng 4-9. Mô tả các khối của fully connected 2D.

Khối	**Mô tả**
storeWeight	Lưu trọng số của mạng
Reg	Lưu giá trị data_in.
fpadd	Khối thực hiện cộng số floating point.
Floating multi	Nhân với ma trận 1 chiều.

4.4 Khối softmax.



Hình 4-14. Sơ đồ khối của Softmax

Module này được sử dụng với mục đích kiểm tra kết quả mô phỏng với kết quả Keras, trong module sử dụng các toán tử luỹ thừa, chia, ... dẫn đến module này sẽ không tổng hợp được.

Trong ngôn ngữ mô tả phần cứng Verilog có hỗ trợ mô phỏng các phép toán trên dấu chấm động chính xác kép (Real), nhưng IP lại sử dụng dấu chấm động chính xác đơn (Float). Vì vậy, cần phải chuyển đổi qua lại giữa hai loại dấu chấm động trên bằng hai thuật toán như Hình 4-14 và Hình 4-15.

Thuật toán 1 Mã giả chuyển đổi Float sang Real

Input: z[31:0]

Output: y[63:0]

1: y = {z[31], z[30], {3{~z[30]}}, z[29:23], z[22:0], {29{1'b0}}}

Hình 4-15. Thuật toán chuyển đổi Float sang Real

Thuật toán 2 Mã giả chuyển đổi Real sang Float

Input: z[63:0]

Output: y[31:0]

1: y = {z[63], z[62], z[58:52], z[51:29]}

Hình 4-16. Thuật toán chuyển đổi Real sang Float

Module Softmax 2 sẽ được thiết kế như hình dựa trên công thức và lý thuyết đã nêu ở mục 2.1.4.2. Module nhận 2 giá trị đầu vào, 2 giá trị đó sẽ được chuyển sang kiểu Real, và được tính hàm mũ e. Sau đó tổng 2 kết quả của hàm mũ e. Kết quả của softmax sẽ là từng kết quả của hàm mũ e chia cho tổng 2 kết quả của hàm mũ e.

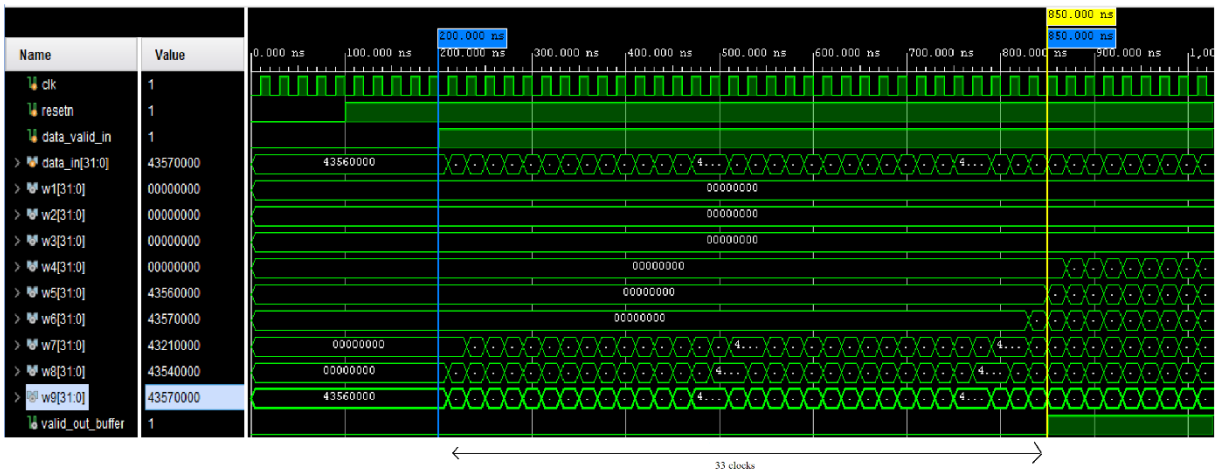
Chương 5. Mô phỏng, Đánh giá.

5.1 Mô phỏng thiết kế khối Convolution.

5.1.1 Khối convolution 2D 3x3.

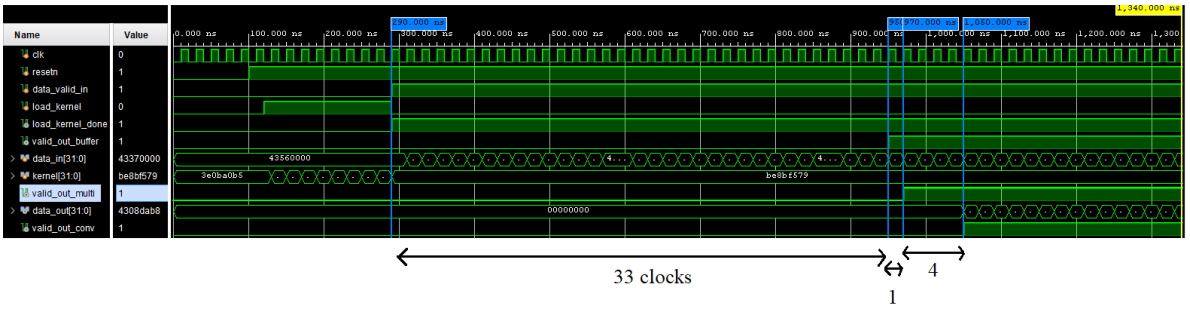
Để thực hiện chức năng Convolution 2 chiều trên phần cứng, dữ liệu điểm ảnh (đặc trưng) có kích thước HxW - với H là chiều cao, W là chiều rộng của ảnh - đầu vào của khối Convolution cần được làm trễ (delay) (K>>1)x(W + 1) chu kỳ so với đầu vào, với K là kích thước của bộ lọc. . Kỹ thuật delay để trích xuất cửa sổ trượt, hỗ trợ cho việc nhân tích chập gọi là Line Buffer

Hình 5-1 là dạng sóng của thiết kế Line buffer, với K bằng 3 và W, H bằng 32. Ta có thể thấy, khoảng cách giữa valid_in và valid_out của line buffer là 33 chu kỳ xung, các giá trị w1 đến w9 là giá trị cửa sổ trượt được trích xuất ra dựa trên đặc trưng đầu vào. Giá trị w1 đến w9 sẽ được nhân với giá trị kernel của bộ lọc, tín hiệu cho phép nhân chính là tín hiệu valid_out của line buffer. Khối nhân này có độ trễ là 1 chu kì xung. Tích của tín hiệu w1 đến w9 và giá trị kernel sẽ được thực hiện cộng song song 4 tầng, cho ra kết quả của khối Convolution 2D, độ trễ của mỗi tầng là 1 chu kì xung.



Hình 5-1. Dạng sóng của line buffer

Tổng cộng khối convolution sẽ cho ra kết quả sau 33 + 1 + 4 = 38 chu kì.

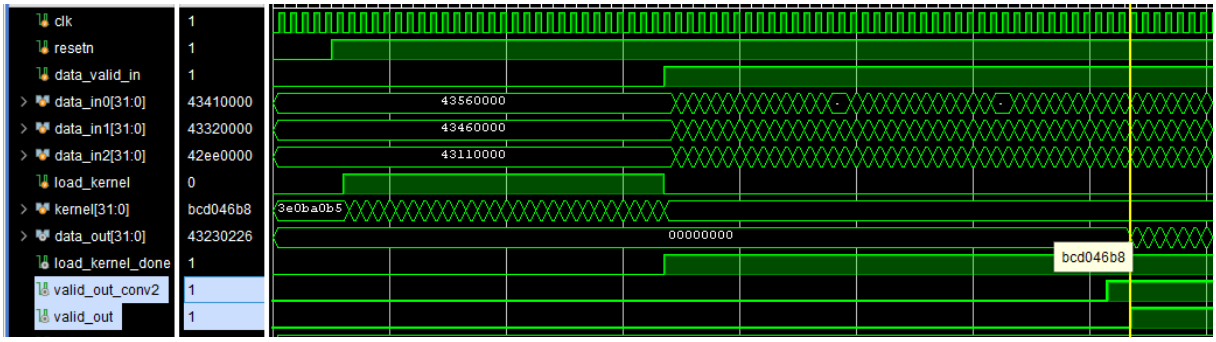


Hình 5-2. Dạng sóng của convolution 2D3x3

5.1.2 Khối Convolution 3D

Khối Convolution 3D được hoạt động dựa trên nguyên lý giống với Convolution 2D, nhưng kết quả của những khối Convolution 2D sẽ được tính tổng với nhau, cho ra kết quả của khối 3D và Convolution 3D có thêm 2 thông số là chiều sâu của đầu vào và số bộ lọc. Giả sử với nhân tích chập 3 chiều, kích thước bộ lọc là 3, chiều sâu của đầu vào là 3 và số bộ lọc là 4.

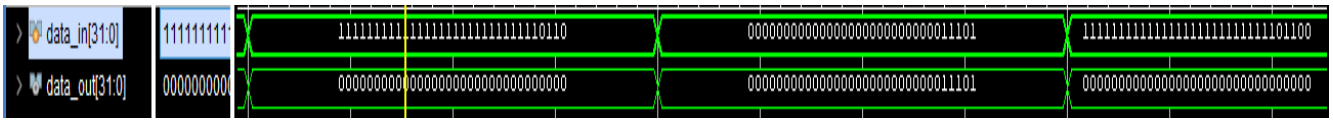
Dạng sóng trên Hình 5-3 mô tả hoạt động của tích chập 3 chiều trên một bộ lọc được mô tả ở Hình 4-6. Trong đó, các dữ liệu output của khối convolution 2D sẽ được tính tổng lại, sau đó được cộng với ‘bias’ để cho ra kết quả tích chập 3 chiều trên 1 bộ lọc. Tương tự, đối với trường hợp nhiều bộ lọc, ta gọi song song các khối Convolution 3 chiều trên một bộ lọc như Hình 4-7.



Hình 5-3. Dạng sóng của convolution 3D.

5.2 Mô phỏng thiết kế khối RELU

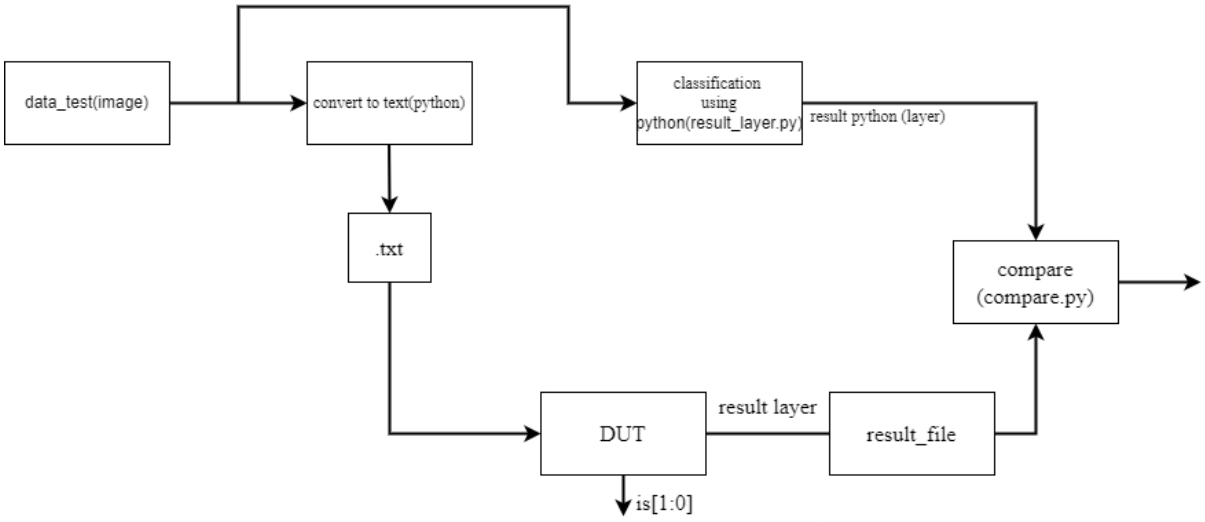
Khối ReLU (xem thiết kế ở mục 4.2.3) là một mạch tổ hợp, trả về kết quả lớn nhất của đầu vào vào '0'. Với thiết kế sử dụng cổng MUX 2-1, chân select là bit dấu của dấu chấm động, dạng sóng của khối ReLU như Hình 5-4. Như ta có thể thấy, với MSB bằng 0, đầu ra sẽ chính bằng đầu vào, còn với trường hợp MSB bằng 1, đầu ra bằng 0.



Hình 5-4. Dạng sóng của khối Relu.

5.3 Kết quả mô phỏng mạng kiến trúc CNN đã xây dựng

Đây là kết quả cuối cùng sau khi chúng ta xây dựng thành công mô hình mạng CNN như đã mô tả trong Chương 4. Tại phần này chúng ta sẽ đưa vào 1 ảnh trong tập test của bộ dữ liệu. với đầu vào của thiết kết HDL, ảnh sẽ được convert sang dạng text trước khi đưa vào DUT. Thao tác đó được thực hiện bằng python. Dưới đây là quy trình phân loại 1 ảnh bất kỳ với dữ liệu từ tập test của bộ dữ liệu .



Hình 5-5. Sơ đồ testbench.

Trong phần báo cáo này, chúng ta chỉ đưa 1 vài hình ảnh từ 2 loại ảnh(chứa chó và không có chó) để kiểm tra và đánh giá của mô hình CNN.

Dưới đây là 2 ảnh bất kì thuộc 2 loại ảnh khác nhau được sử dụng để kiểm tra và đưa vào trong báo cáo.

Hình 5-6. Test1

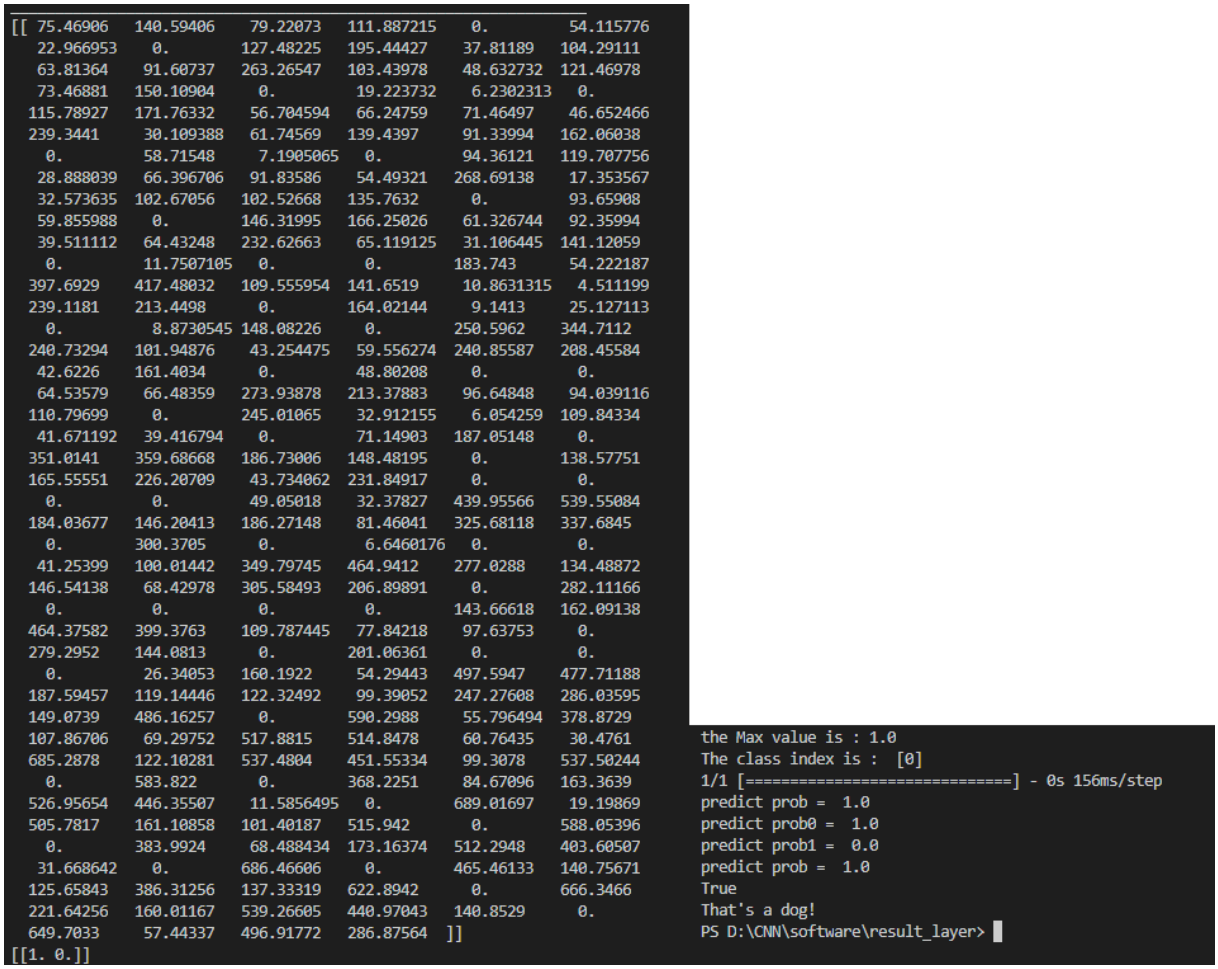
Hình 5-7. Test2

Dưới đây là kết quả dạng sóng chạy trên vivado.

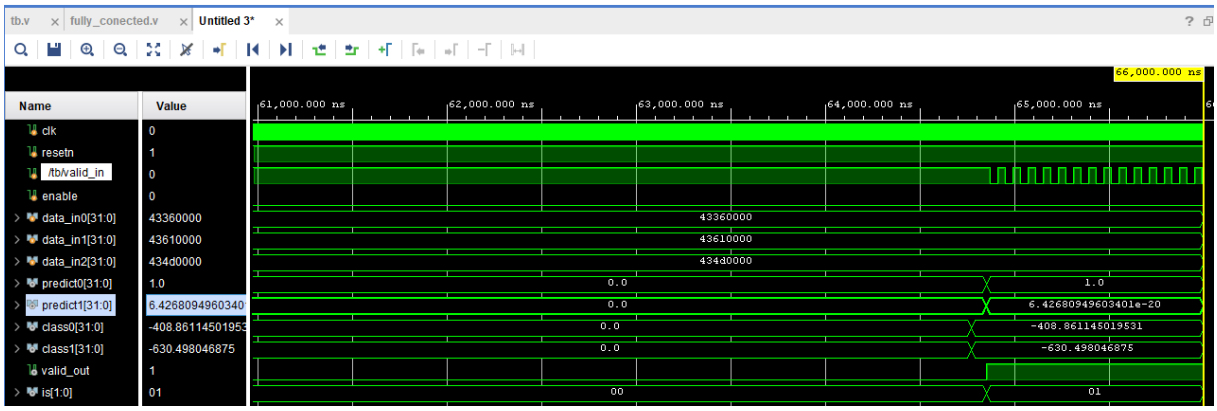
- Test1(Hình 5-6)

Đầu vào của ảnh là 32x32x3.

Kết quả trên python cho test1.



Hình 5-8. Kết quả trên python.



Hình 5-9. Kết quả trên thiết kế.

Từ kết quả của vivado ta thấy dữ liệu của is[1:0] = 2'b01 nên ảnh đó thuộc lớp 1 (ảnh chó) kết quả của vivado cho ra kết quả như kết quả python.

Dưới đây lần lượt là kết quả của test2.

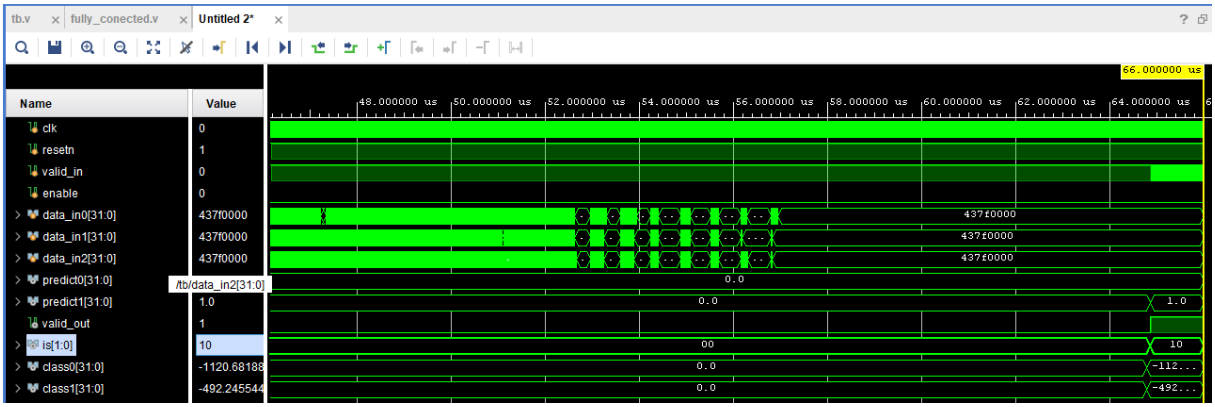
- Test2(Hình 5-7):

Đầu vào cũng là ảnh 32x32x3.

Kết quả python cho test2.

```
[[241.51265   386.01178    72.76229   428.4779     0.          0.
  106.24857     0.        447.24542   720.84393   245.47192   195.92287
  291.60623   173.29022   680.7061    391.10278   252.71419   358.8012
  132.23909   418.65024     0.          0.         42.426567  119.122536
  501.73706   480.24368   103.40807    86.1391    359.1161      0.
  727.31854    32.13208   222.65611   457.03784     0.        559.5199
    0.         65.56475   122.03356    75.15894   429.99472   563.2907
   29.90861    49.419518  425.83435   129.54828   671.91656   263.62064
  156.04999   416.30966   192.83324   486.75        0.        329.52173
   82.78407   130.28966   542.7722    522.3193    197.14577    72.12323
  398.9458      3.754939  754.62756   214.37018   142.73738   353.9262
    0.        160.6029      0.          0.         11.315459    0.
  582.8989    830.8749    281.28745   359.26373   122.68738   326.94916
  535.26      484.09265    54.1851    359.7439    160.36623   412.31192
    0.        279.07907   119.61025     0.        532.7885    673.2474
  432.44324   186.39026   287.32828   180.61769   601.7532    215.71344
  192.93558   252.31651     0.        518.2835      0.        321.08212
   29.340416    0.        499.69824   573.23535   202.12314   287.6483
  380.96362    45.807964  688.06964     0.         87.22202   252.41872
   63.250065  260.38818     0.        240.2475    356.6405      0.
  644.5868    575.4484    237.47775   275.22964   273.3043    198.8392
  696.1066    408.40332    70.76245   314.41333     0.        314.7354
    0.         82.505       6.2181773   0.        631.6125    883.2079
  332.1583    361.36243   198.811     377.473     647.36      543.67065
   89.559944  370.94244   164.05151   422.03033     0.        265.9362
  169.61104    31.152912  364.49997   710.0218    353.2138    243.14122
  289.92255    85.63948   680.0222     58.284966   21.34498   380.6702
    0.         23.81599     0.          0.        240.38863    89.10958
  720.80725   549.1053    155.3125    143.25877   202.97879     6.172145
  486.02252   151.18156    21.847391  217.78535     0.         84.92182
    0.        179.32974   209.76176     0.        712.4471    690.57806
  277.69724   166.9153    184.04369   204.17276   443.27713   389.56543
  234.91609   597.81213     0.        676.9987      0.        529.0878
  118.66599     0.        537.71436   700.3992    213.10217   245.21904
  741.9555     89.41407   814.7258    630.34906   100.409386  680.18207
    0.        679.39465     0.        446.157     290.41342    91.01696
  603.45056   460.52576    45.822193   46.341896  767.8345    24.417389
  793.8233    188.56802   110.56524   538.3069      0.        653.19415
    0.        457.6014     52.944923  117.75486   642.0901    560.63336]
the Max value is : 1.0
The class index is :  [1]
1/1 [==============================] - 0s 267ms/step
predict prob =  1.0
predict prob0 =  0.0
predict prob1 =  1.0
predict prob =  0.0
False
That's not a dog!
PS D:\CNN\software\result_layer>
```

Hình 5-10. Kết quả trên python



Hình 5-11. Kết quả trên thiết kế.

Từ kết quả của vivado ta thấy dữ liệu của is[1:0] = 2'b10 nên ảnh đó thuộc lớp 1 (ảnh không chứa chó) kết quả của vivado cho ra kết quả như kết quả python.

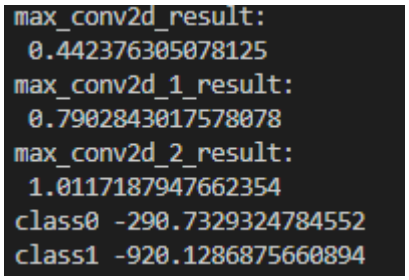
❖ Kết luận: từ 2 ảnh trên ta thấy về mặt nhận diện ảnh thì cả hai kết quả giữa python và verilog đều đưa ra kết quả như nhau và kết quả này đều đúng. Về mặt sai số thì thiết kế Verilog có tỉ lệ sai số lớn. Điều này do các module floating point add và floating point multiplier chưa thực sự chính xác.vì vậy các đầu ra về những layer sau càng lúc càng lớn.

5.4 Đánh giá

Nhóm đưa đầu vào là hình 0020.png(dog) trong tập test vào thiết kế CNN và trên model python thu được kết quả output của fully conneted như sau:

Bảng 5-1. Đánh giá sai số trên python và thiết kế

Phần trăm dự đoán lớp	**Kết quả trên python**	**Kết quả trên thiết kế**	**Sai số**
0	-290.7329324784552	-290.448364257813	0.284568221
1	-920.1286875660894	-921.29443359375	1.165746028

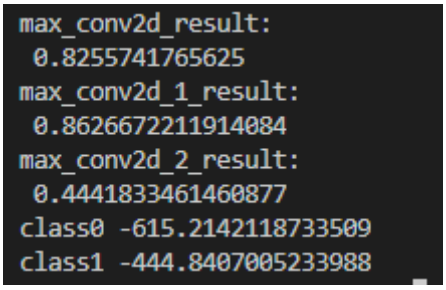


Hình 5-12. Sai số lớn nhất của các output của các khối convolution(dog)

Tương tự với hình 0020.png(not_dog).

Bảng 5-2. Đánh giá sai số trên python và thiết kế

Phần trăm dự đoán lớp	**Kết quả trên python**	**Kết quả trên thiết kế**	**Sai số**
0	-615.2142118733509	-615.159790039063	0.054421834
1	-444.8407005233988	-445.193481445313	0.352780922



Hình 5-13. Sai số lớn nhất của các output của các khối convolution(not_dog)

Chương 6. Tổng kết

6.1 Kết luận.

Nhóm đã đạt được mục tiêu là mô phỏng thành công mạng CNN dùng để nhận diện chó. Nhóm đã thiết kế một mạng CNN đơn giản gồm 4 lớp trong đó 3 lớp tích chập kèm lớp maxpooling sau khi mỗi lần tích chập.Một lớp fully connected và có phần chuẩn hóa đầu ra softmax. Thiết kế verilog hoạt động đúng như chức năng là xác định được ảnh đầu vào thuộc lớp nào giống với model CNN trên python.

6.2 Khó khăn gặp phải.

Kiến thức về CNN còn mới nên gặp khó khăn trong quá trình tìm hiểu và thiết kế, những nghiên cứu liên quan không trình bày những thiết kế của họ nên tài liệu tham khảo khan hiếm.

Quá trình mô phỏng mất nhiều thời gian, với mỗi ảnh sẽ mất 5 phút để cho ra kết quả, do đó, gây khó khăn trong quá trình sửa lỗi.

6.3 Hướng phát triển.

Mong muốn có thể ứng dụng thiết kế mạng CNN để nhận diện ảnh trong thời gian thực.

Cải thiện tài nguyên phần cứng

Tài liệu tham khảo

[1] D. H. Hubel and T. N. Wiesel Receptive Fields of Single Neurones in the Cat's Striate Cortex J. Physiol. pp. 574-591 (148) 1959

[2] D. H. Hubel and T. N. Wiesel Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex J. Physiol. 160 pp. 106-154 1962

[3] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, Vol. 36, No. 4, pp. 193–202, April 1980.

[4]Karen Simonyan, Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *arXiv:1409.1556*, 2014.

[5] Huynh Vinh Phu, Tran Minh Tan, Phan Van Men, Nguyen Van Hieu, Truong Van Cuong, "*Design and Implementation of Configurable Convolutional Neural Network on FPGA", 2019-12, University of Information Technology, VNU-HCM, Vietnam*