ỨNG DỤNG MẠNG NƠRON, MẠNG NƠRON XOẮN VÀ SỬ DỤNG KẾT HỢP CPU - GPU ĐỂ TĂNG HIỆU NĂNG TÍNH TOÁN TRONG PHÂN LOẠI ẢNH

Hồ Sỹ Phương, Phan Văn Dư, Lê Văn Chương, Tạ Hùng Cường
Viện Kỹ thuật và Công nghệ, Trường Đại học Vinh
Ngày nhận bài 30/9/2018 , ngày nhận đăng 29/11/2018

Tóm tắt: Bài báo trình bày và so sánh các phương pháp phân loại ảnh dựa trên mạng nơron nhân tạo nhiều lớp (Multi Layer Perceptron - MLP) và mạng nơ ron xoắn (Convolutional Neural Network - CNN). Dữ liệu được đưa vào huấn luyện là 50.000 bức ảnh của 10 đối tượng khác nhau. Kiến trúc thứ nhất được sử dụng là mạng MLP gồm có 3.853.298 tham số (weight), kiến trúc thứ hai là mạng CNN gồm 528.054 tham số. Bài báo đã đề xuất một vài phương pháp và cấu trúc mạng nhằm tránh hiện tượng quá khớp (overfitting), tăng cường độ chính xác cho mô hình xấp xỉ 80%. Bên cạnh đó, bài báo cũng trình bày và so sánh về thời gian huấn luyện khi sử dụng CPU và kết hợp sử dụng CPU với GPU.

1. MỞ ĐẦU

Trong những năm gần đây, sự phát triển của khoa học công nghệ và cách mạng công nghiệp 4.0 đang làm cho các nghiên cứu về trí tuệ nhân tạo (Artificial Intelligence - AI) ứng dụng trong lĩnh vực robotics, robot tương tác thời gian thực với môi trường xung quanh... thu hút được sự quan tâm của các chuyên gia trong lĩnh vực điều khiển. Trong robot tự hành, để có thể tương tác với môi trường hoạt động và điều khiển robot chuyển động theo đúng quỹ đạo mong muốn, vấn đề nhận biết, phân tích, nhận dạng và phân loại các vật thể đóng vai trò hết sức quan trọng, giúp chúng ta có cơ sở để đưa ra các tín hiệu điều khiển một cách chính xác, kịp thời. Nhiều công trình nghiên cứu được công bố trên các tạp chí khoa học trong và ngoài nước [1], [10], [12], [15] cho thấy vấn đề này có thể giải quyết và đưa lại hiệu quả cao khi sử dụng mạng MLP, trong đó việc nhận dạng chữ viết tay với độ chính xác lên đến 99,8%.

Trong bài báo này, nhóm tác giả nghiên cứu và ứng dụng các cấu trúc mạng MLP và mạng CNN kết hợp với các kỹ thuật tối ưu nhằm nâng cao khả năng phân loại các đối tượng, thực hiện so sánh hiệu quả huấn luyện mạng khi sử dụng CPU với sử dụng kết hợp CPU - GPU về độ chính xác và tốc độ huấn luyện mạng.

2. KIẾN TRÚC MẠNG MLP, CNN VÀ CÁC KỸ THUẬT TỐI ƯU MẠNG

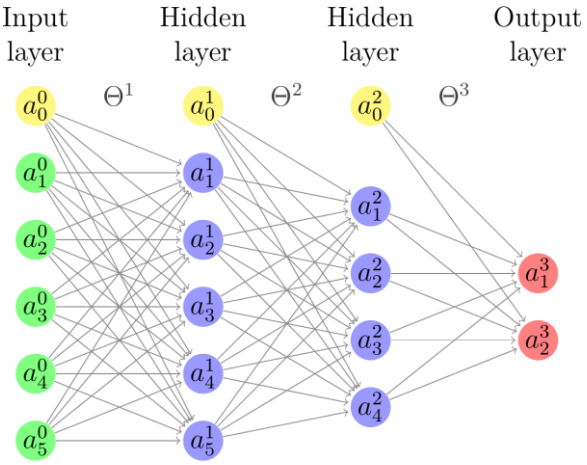
2.1. Mạng nơron nhân tạo và kiến trúc mạng MLP

Mạng nơron nhân tạo, gọi tắt là mạng nơron là một mô hình toán học được xây dựng dựa trên cơ sở các mạng nơron sinh học gồm một số lượng lớn các phần tử (gọi là nơron) kết nối với nhau thông qua các liên kết (gọi là trọng số liên kết) làm việc như một thể thống nhất để giải quyết các vấn đề cụ thể như nhận dạng mẫu, phân loại dữ liệu,v.v... thông qua một quá trình học từ tập các mẫu huấn luyện.

Email: Hophuong@vinhuni.edu.vn (H. S. Phương)

Mô hình mạng nơron thường được sử dụng rộng rãi nhất là mô hình mạng truyền thẳng nhiều lớp (MLP - Multi Layer Perceptron). Một mạng MLP tổng quát là mạng có n lớp ($n \geq 2$) trong đó bao gồm một lớp vào, một lớp ra và một hoặc nhiều lớp ẩn (hình 1).

Hoạt động của mạng MLP như sau: Tại lớp vào, các nơron nhận tín hiệu vào xử lý (tính tổng trọng số, gửi tới hàm truyền) rồi cho ra kết quả (là kết quả của hàm truyền); kết quả này sẽ được truyền tới các nơron thuộc lớp ẩn thứ nhất; các nơron tại đây tiếp nhận như là tín hiệu đầu vào, xử lý và gửi kết quả đến lớp ẩn thứ 2;…; quá trình tiếp tục cho đến khi các nơron thuộc lớp ra cho kết quả.



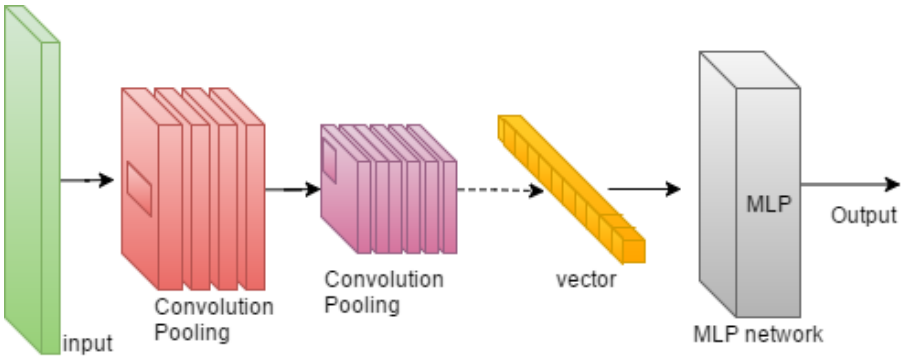
Hình 1: *Mạng MLP 4 lớp*

Mạng MPL được sử dụng rất thành công trong việc nhận dạng chữ viết tay [2], [10]. Cơ sở dữ liệu phổ biến rộng rãi nhất cho bài toán này là MINST [16], khi huấn luyện chỉ với 2 lớp ẩn cho độ chính xác lên tới 99,8%. Kiến trúc mạng MPL như hình hình 1, gồm có 1 lớp đầu vào, 2 lớp ẩn và 1 lớp ra. Tùy theo yêu cầu của bài toán, ta có được số lượng đầu vào. Để có độ chính xác cao, tránh hiện tượng quá khớp (overfitting) thì số lượng lớp ẩn và số nơron trên nó là yếu tố quyết định [7].

Ví dụ, trong bài toán nhận dạng chữ viết tay, bộ dữ liệu huấn luyện từ tập MINST có kích thước 28x28 nên số nơron đầu vào là (28x28)= 784, số nơron lớp ẩn 1 là 512, lớp ẩn 2 là 512 và số nơron đầu ra tương ứng từ 0→9 là 10. Độ chính xác khi huấn luyện là 99.93% và khi kiểm chứng trên mô hình thì độ chính xác đạt gần 99,8%. Trên cơ sở kết quả đó, nhóm tác giả thực hiện thử nghiệm sử dụng mạng MPL trong việc phân loại các đối tượng.

2.2. Kiến trúc mạng CNN và các kỹ thuật tối ưu mạng

Kiến trúc mạng CNN [1], [2], [12] là kiến trúc mở rộng của mạng MLP được sử dụng rộng rãi trong kỹ thuật học sâu (deep learning) đặc biệt là trong lĩnh vực thị giác máy tính (computer vision). Một trong những giải pháp nhằm tối ưu quá trình huấn luyện mạng được đề xuất là giảm số lượng các trọng số (weight) để tăng tốc độ tính toán, giảm thời gian huấn luyện, tránh hiện tượng quá khớp khi mà lượng dữ liệu đầu vào là rất lớn như các bức ảnh màu, video…



Hình 2: *Kiến trúc của một mạng CNN*

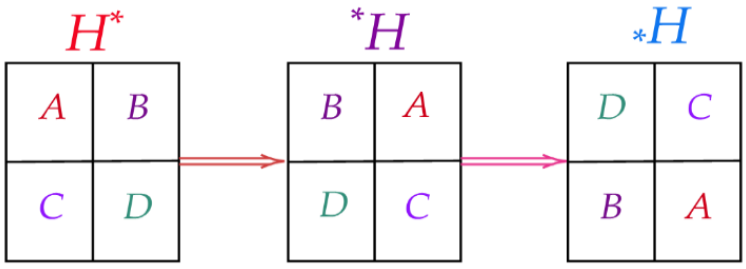
Mạng CNN có cấu trúc như hình 2, với đầu vào sẽ được nhân xoắn với các ma trận lọc, công việc này có thể được xem như phép lọc ảnh với ma trận lọc khi sử dụng dạng ma trận [7], cũng như phép lọc ảnh bình thường trong không gian 2D thì tích xoắn này cũng được ứng dụng trong trong không gian ảnh màu 3D và trong cả không gian n chiều. Sau khi thu gọn ma trận dưới dạng một véc tơ thì nó sẽ được kết hợp với một mạng MLP đầy đủ như được mô tả ở mục 2.1, với các ảnh xám ma trận đầu vào là 2 chiều, còn với ảnh màu ma trận vào sẽ là 3 chiều.

Khi xem xét mạng CNN, khái niệm tích xoắn (tích chập) trong đại số là cơ sở của phần mạng CNN:

Đưa vào ảnh xám X và bộ lọc ω có kích thước [m,n], tích xoắn giữa ω và X là: $y = X * \omega$, trong đó các thành phần của ma trận y sẽ được tính theo công thức:

$$y_{ij} = \sum_{a=0}^{m-1}\sum_{b=0}^{n-1} X[i-a, j-b]\cdot\omega[a,b] \text{ hay } y_{ij} = \sum_{a=0}^{m-1}\sum_{b=0}^{n-1} X[i+a, j+b]\cdot\omega[-a,-b] \quad (1)$$

Giá trị của ma trận $\omega[-a,-b]$ được hiểu là giá trị tại $[a,b]$của ma trậnω sau khi được lật từ phải sang trái và đảo ngược từ dưới lên (như hình 1)



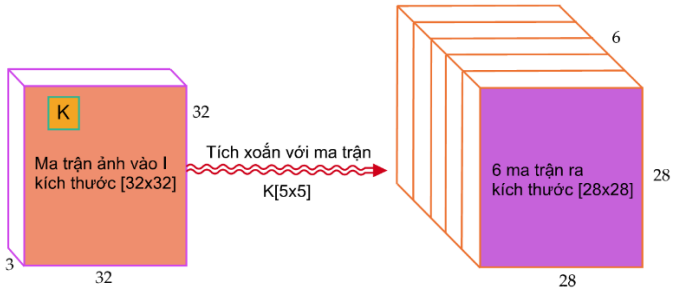
Hình 3: *Minh họa phương pháp tìm giá trị $\omega[-a,-b]$ từ ma trận gốc bên trái qua phép lật giữa và đảo từ dưới lên cho kết quả là ma trận bên phải*

Kiến trúc mạng CNN bao gồm các lớp được gọi là lớp xoắn với ma trận đầu vào là I, bộ lọc K và trọng số b. Ta giả thiết rằng I là ma trận ảnh màu có kích thước $[C \times H \times W]$, trong đó $C = 3$ là số ma trận màu R,G,B và ảnh có kích thước $[H,W]$. Khi đó $I \in R^{H\times W\times C}$, $K \in R^{k_1\times k_2 xC}$ và b có kích thước $b \in R^D$. Tích xoắn giữa I và K sẽ có ma trận mới chính là ma trận đầu ra của lớp xoắn:

$$(I*K)_{ij}=\sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1}\sum_{c=1}^{C}K_{m,n,c}\cdot I_{i+m,j+n,c}+b, \quad (2)$$

trong trường hợp này, I chính là ma trận được lật như hình 3 từ ma trận lọc I.

Với lớp CNN có kích thước $C>3$ thì công thức tính toán vẫn như trên và ma trận K sẽ có độ sâu C như ma trận I. Ví dụ, cho 6 ma trận lọc có kích thước 5x5, tích xoắn của ảnh màu I với lần lượt từng bộ lọc K sẽ tạo ra 6 ma trận được sắp xếp như hình 4.



Hình 4: *Ma trận đầu ra được tạo từ 6 ma trận con qua phép tích xoắn giữa ma trận vào I và ma trận lọc K trượt trên I*

Khi xây dựng được mạng CNN người ta thêm vào các kỹ thuật pooling, dropout, normalizing, regularization để tối ưu thời gian huấn luyện và tránh hiện tượng quá khớp [7]:

- Pooling [7]: Quá trình giảm kích thước ảnh sau khi tính tích xoắn để lấy các pixel đặc tính đặc trưng nhất.

- Dropout [7], [11]: Cắt bớt số nơron khi thực hiện thuật toán lan truyền ngược nhằm tăng tốc độ huấn luyện mạng

- Normalizing [7]: Kỹ thuật chuẩn hóa dữ liệu về dạng dữ liệu trong dải tính toán phù hợp.

Giả sử X là dữ liệu cần chuẩn hóa, khi đó:

+ Kỳ vọng của dữ liệu tính được: $$\mu=\frac{1}{m}\sum_{1}^{m}X^{(i)}; \quad (3)$$

+ Sai lệch của dữ liệu so với kỳ vọng: $$\Delta X=X-\mu; \quad (4)$$

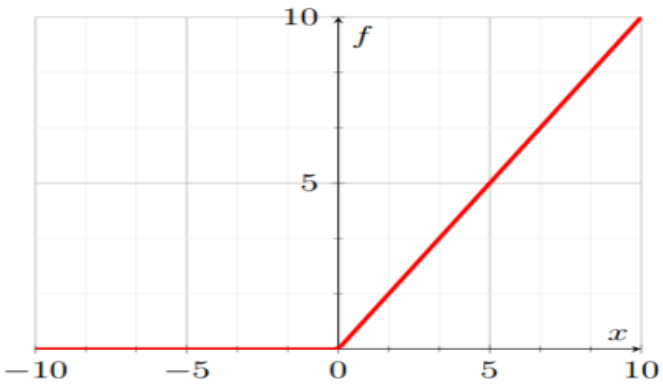
+ Phương sai của dữ liệu ΔX : $$\sigma^2=\frac{1}{m}\sum_{1}^{m}(\Delta X)^2, \quad (5)$$

do đó đầu ra dữ liệu được chuẩn hóa sẽ là: $X=\frac{\Delta X}{\sigma^2}$. (6)

- Regularization [7]: Kỹ thuật sử dụng thêm tham số λ trong hàm mục tiêu J khi tối ưu hóa:

$$J_{regularized}=-\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(a^{[L](i)}\right)+\left(1-y^{(i)}\right)\log\left(1-a^{(L)(i)}\right)\right)+\frac{1}{m}\frac{\lambda}{2}\sum_{l}\sum_{k}\sum_{j}W_{k,j}^{[l]2} \quad (7)$$

Các mô hình CNN được xây dựng chủ yếu sử dụng hàm ReLU (hình 5) để tăng tốc độ tính toán so với các hàm phi tuyến khác và có đạo hàm không đổi khi huấn luyện.



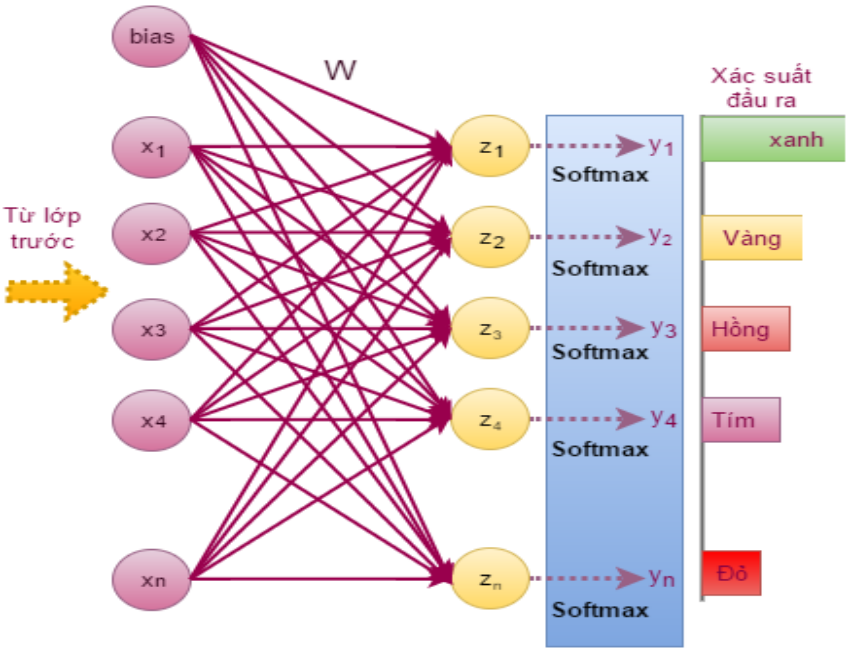
Hình 5: *Hàm ReLU*

$$\text{ReLU}(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \quad (8)$$

Chúng ta cần một mô hình xác suất sao cho với mỗi đầu vào thì xác suất để đầu vào đó rơi vào lớp i phải dương và tổng của chúng bằng 1. Để làm được việc đó, ta xây dựng hàm Softmax (hình 6). Ta có:

$$y_k = \frac{e^{z_k}}{\sum_{j=1}^{C} e^{z_j}}, \forall i = 1,2,....,C \; ; \; z = W^T X \; . \quad (9)$$

Ta có: $y_k = P(a_k = k \mid i_k, W)$, trong đó y_k thể hiện xác suất của đầu ra mô hình rơi vào lớp k khi có đầu vào i_k và tham số mô hình W (ma trận trọng số).



Hình 6: *Mô hình Softmax Regression dưới dạng Neural Network*

Trong mạng nơron, việc tìm giá trị nhỏ nhất của hàm mất mát (cost functions) là điều bắt buộc. Việc tìm điểm cực tiểu toàn cục của nó rất phức tạp, thậm chí là bất khả thi. Thay vào đó, người ta thường cố gắng tìm các điểm cực tiểu địa phương và có thể xem đó là nghiệm cần tìm của bài toán. Cách tiếp cận phổ biến nhất là xuất phát từ một điểm mà chúng ta xem là gần với nghiệm của bài toán, sau đó dùng một phép toán lặp để tiến dần đến điểm cần tìm, nghĩa là đến khi đạo hàm gần với 0. Giải thuật Gradient Descent và các biến thể của nó là một trong những giải thuật được dùng nhiều nhất.

Giả thiết mạng nơron có K lớp, có tập huấn luyện với m dữ liệu vào - ra như sau:

$$\left\{\left(x^{(1)},y^{(1)}\right),\left(x^{(2)},y^{(2)}\right),\ldots,\left(x^{(m)},y^{(m)}\right)\right\}. \quad (10)$$

Lúc đó hàm mục tiêu (hàm mất mát - cost function) trong bài toán phân lớp hồi quy sẽ là [5,7]:

$$J(\theta)=-\frac{1}{m}\left[\sum_{i=1}^{m}y^{(i)}\log h_\theta\left(x^{(i)}\right)+\left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right]+\frac{\lambda}{2m}\sum_{j=1}^{m}\theta_j^2 \quad (11)$$

Còn với mạng nơron K lớp, hàm mục tiêu sẽ là [10][7]:

$$J(\theta)=-\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K}y_k^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right)_k+\left(1-y_k^{(i)}\right)\log\left(1-\left(h_\theta\left(x^{(i)}\right)\right)_k\right)\right]+\frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(\theta_{ji}^{(l)}\right)^2, \quad (12)$$

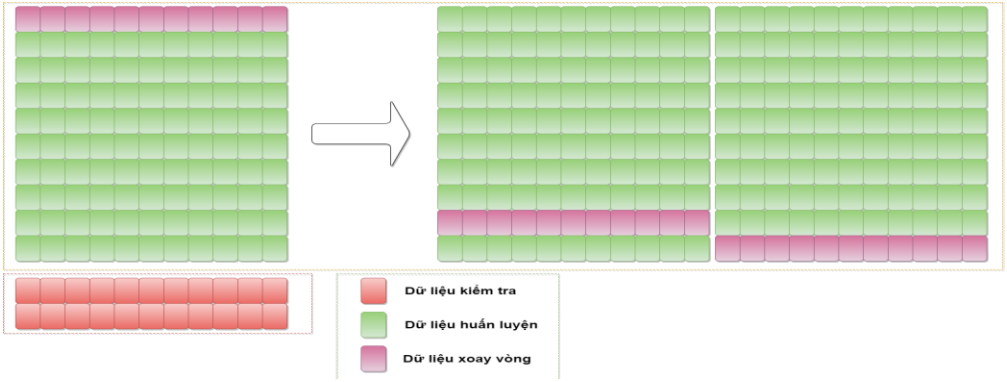
với $\left(h_\theta\left(x\right)\right)_i=i^{th}$ đầu ra thứ *i* của mạng.

Để tối ưu hóa hàm mục tiêu trên ta tìm $MinJ(\theta)$ sau đó tiến hành cập nhật tham số mô hình θ (ma trận các trọng số của mạng) bằng giải thuật Gradient Descent. Giải thuật này phổ biến và cho hiệu quả tốt nhất khi tập dữ liệu lớn (Big data), trong học sâu người ta đề xuất thêm các biến thể của giải thuật Gradient Descent như SGD [7], [13], [17], Adam [5], [6], [7], [8], RMSProp [7], [8]. Trong bài báo này, chúng tôi sử dụng thuật toán tối ưu RMSProp là một phương pháp đơn giản, cho hiệu quả cao trong bài toán phân loại ảnh [1], [12].

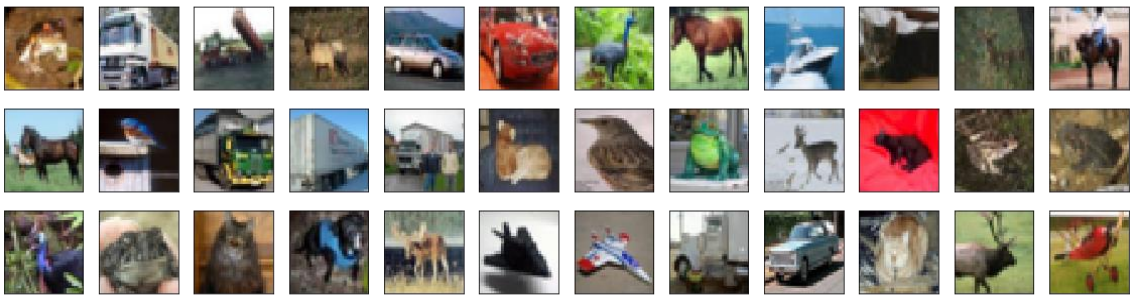
3. THỬ NGHIỆM HUẤN LUYỆN VÀ KIỂM CHỨNG MÔ HÌNH

3.1. Tập hợp dữ liệu huấn luyện

Trong bài báo này, tập hợp dữ liệu được trích ra từ tập hợp dữ liệu CIFAR-10 [15] gồm 50.000 bức ảnh dùng làm tập huấn luyện (training data), 10.000 bức ảnh dùng làm tập kiểm chứng mô hình (test data), trong tập dữ liệu huấn luyện có 10% dữ liệu (5.000 bức ảnh) ngẫu nhiên được loại ra bằng cách xoay vòng để tránh trường hợp chưa khớp (underfitting), mỗi bức ảnh trong tập dữ liệu là ảnh màu có kích thước 32x32x3 điểm ảnh, các bức ảnh trong tập dữ liệu này đã được tiền xử lý và đảm bảo chỉ có một đối tượng xuất hiện trong một bức ảnh, dữ liệu phân loại được phân theo 10 lớp.



Hình 7: *Phân chia tập hợp dữ liệu trong bài báo*



Hình 8: *Lấy ngẫu nhiên 36 bức ảnh trong tập dữ liệu huấn luyện*

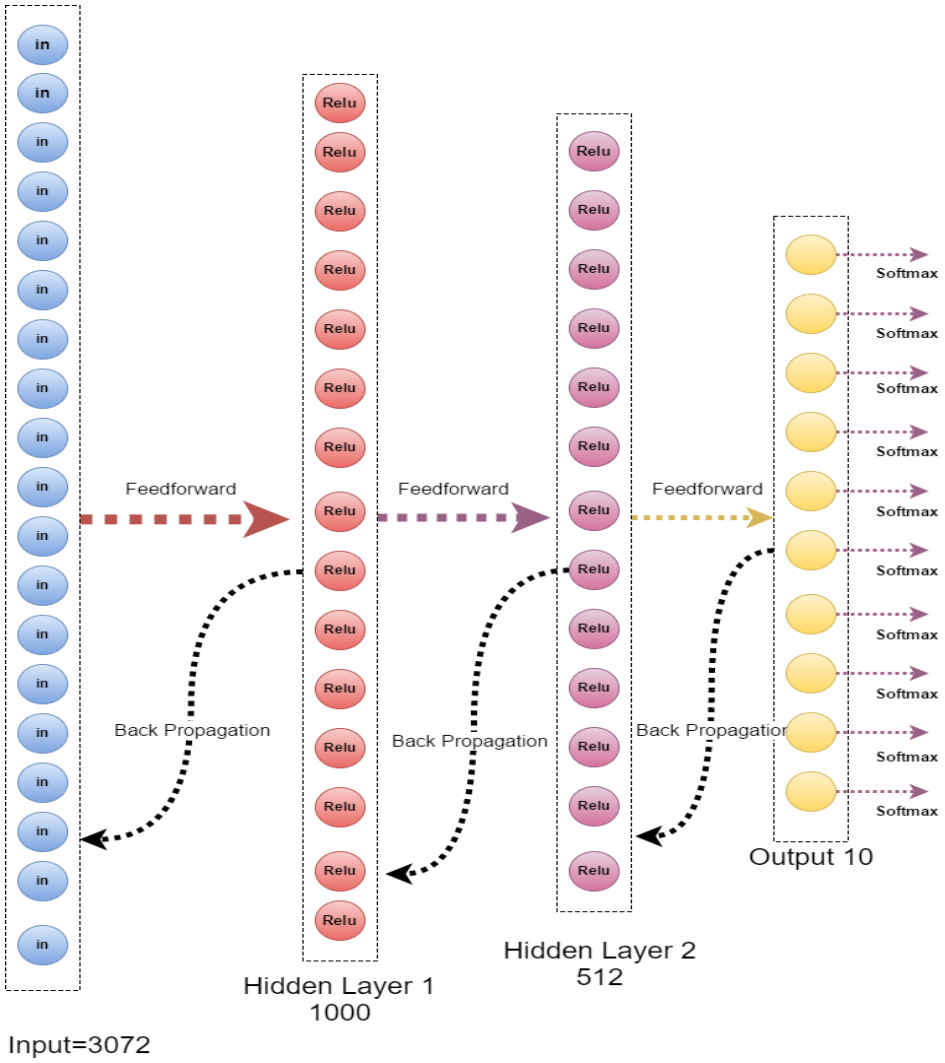
3.2. Cấu trúc phần cứng

Máy tính sử dụng có cấu hình:
- CPU: Intel® Core™ i7 6700HQ Processor;
- RAM: 2 thanh DDR4 2133 MHz SDRAM 4GB;
- Card đồ họa: Integrated Intel® HD Graphics 530;
- GPU rời: NVIDIA® GeForce® GTX 950M với 4G GDDR5 VRAM, trong đó:
+ Lõi CUDA: 640;
+ Xung BoostBase (MHz): 914;
+ Xung cho bộ nhớ: 2500 MHz;
+ Giao tiếp bộ nhớ chuẩn GDDR5;
+ Độ dài dữ liệu giao tiếp bộ với bộ nhớ:128-bit;
+ Tốc độ giao tiếp với bộ nhớ (GB/sec): 80.

Kiến trúc CUDA (Compute Unified Device Architecture) [8] cho phép tăng tốc độ tính toán của chương trình lên nhờ khả năng tính toán song song, hỗ trợ mọi chức năng tính toán thông qua ngôn ngữ C, hỗ trợ các các ngôn ngữ như Python, Fortran, Java và MATLAB để cài đặt các thuật toán chạy trên GPU. Phần mềm được sử dụng là Tensor Flow hỗ trợ GPU và OpenCV.

3.3. Thử nghiệm với mạng MLP

Ta xây dựng mạng MLP có kiến trúc như hình 9 với các thuộc tính như bảng 1 để phân loại các đối tượng trong tập dữ liệu đã nêu ở mục 3.1.



Hình 9: *Mạng MLP ứng dụng trong nhận dạng*

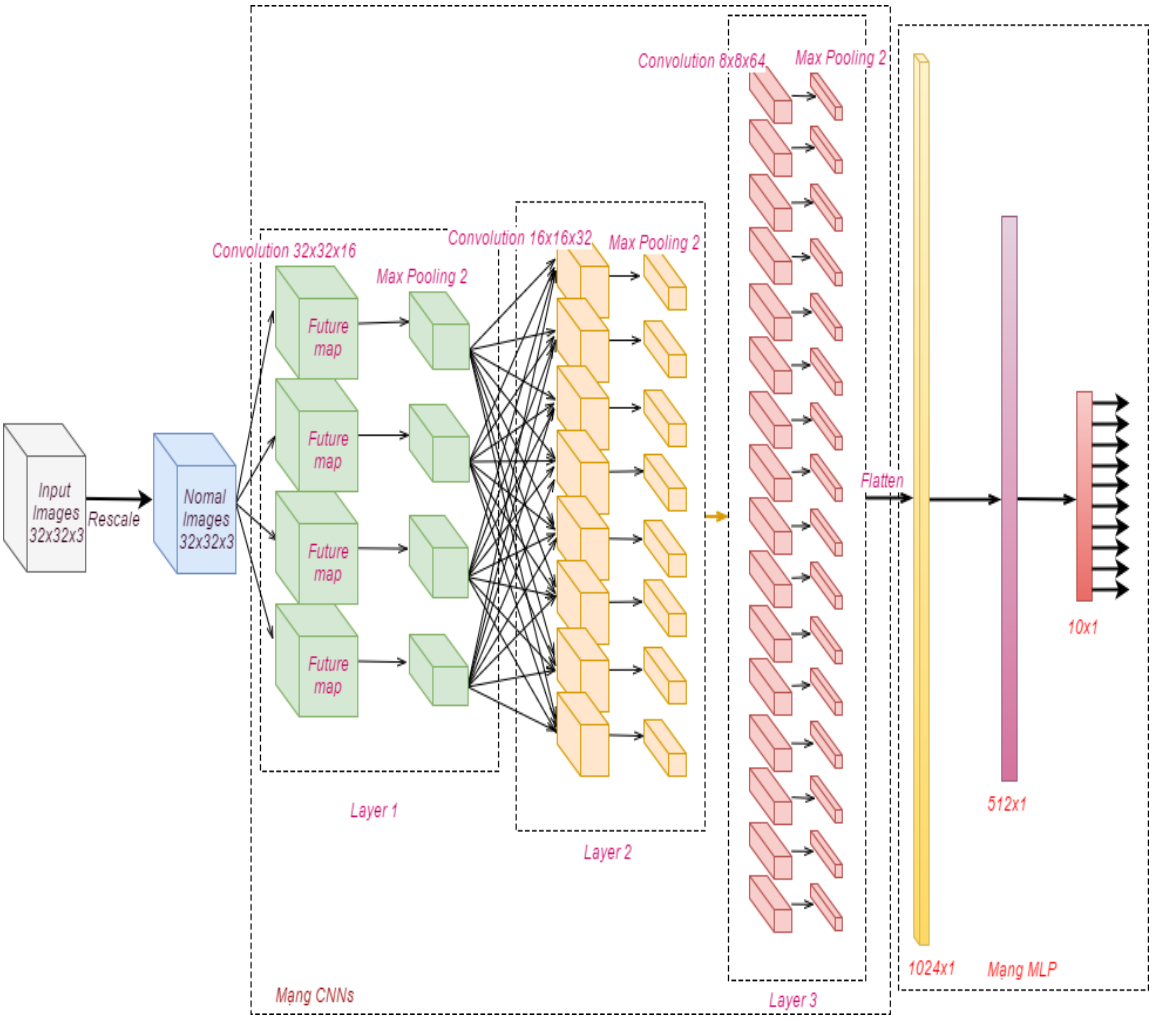
Bảng 1: *Thuộc tính của mạng MLP*

Tên lớp	Số lượng node	Hàm	Dropout	Tham số
Đầu vào	3.072			3.073.000
Lớp ẩn 1	1.000	ReLU	0,2	512.512
Lớp ẩn 2	512	ReLU	0,2	262.656
Đầu ra	10	Softmax		5.130
Tổng	4.594			3.853.298

Mạng MLP sử dụng phương pháp tối ưu RMSprop, kích thước mỗi lần đưa vào huấn luyện là 32 ảnh, số lần huấn luyện là 10.

3.3. Thử nghiệm với mạng CNN

Mạng CNN được xây dựng với các thuộc tính mạng như bảng 2 được chỉ rõ trong hình 10.

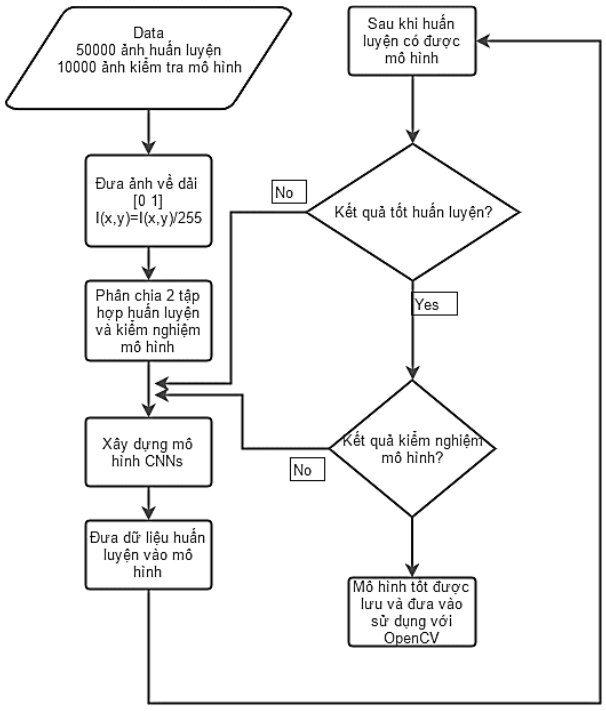


Hình 10: *Mạng CNN trong việc phân loại ảnh*

Bảng 2: *Thuộc tính mạng CNN*

Tên mạng	Tên lớp	Convolution	Pooling	Tham số
CNN	Layer 1	32x32x16	2	208
	Layer 2	16x16x32	2	2.080
	Layer 3	8x8x64	2	8.256
Tên mạng	Tên lớp	Hàm	Dropout	Tham số
MLP	Input			0
	Hidded Layer 1	ReLU	0,4	512.500
	OutPut	Softmax		5.010
Tổng				528.054

Việc xây dựng mô hình được thực hiện theo thuật toán trên hình 11 và để tối ưu hóa mô hình không bị quá khớp với độ chính xác cao, nhóm tác giả đã sử dụng mô hình mạng CNN ở hình 10.



Hình 11: *Sơ đồ chương trình huấn luyện mạng CNN*

3.4. So sánh các kết quả đạt được

Hình 12 thể hiện kết quả của mô hình đã được huấn luyện với bức ảnh không gạch chéo thể hiện kết quả nhận dạng chính xác, còn có gạch chéo cho kết quả nhận dạng sai.

Hình 11: *Kết quả thử nghiệm mô hình CNN*

Bảng 3 thể hiện các kết quả đạt được khi huấn luyện và kiểm nghiệm độ chính xác với hai kiến trúc của mô hình MLP và CNN đã nêu ở trên:

- Mạng MLP cho độ chính xác 45% khi huấn luyện 20 lần (Epochs=20) trong đó mạng CNN là 79% khi huấn luyện 40 lần (Epochs=40) (độ chính xác tăng lên xấp xỉ 80% khi tăng số lần huấn luyện). Khi sử dụng CPU kết hợp GPU thì có thể tối ưu được thời gian huấn luyện lên đến 3→4 lần (tùy thuộc vào cấu trúc phần cứng sử dụng) so với việc sử dụng CPU. Khi số lần huấn luyện tăng lên nhiều thì thời gian huấn luyện khi sử dụng CPU kết hợp GPU tăng lên không nhiều, và tăng không đáng kể khi dùng CPU.

- Khi số lần huấn luyện tăng lên 50, 60,…,100 thì độ chính xác không cải thiện thêm mà chỉ dao động quanh 45% cho mạng MLP và 80% cho mạng CNN do hiện tượng quá khớp. Hiện tượng này xảy ra khi mạng có năng lực quá lớn và để hạn chế bớt năng lực của mạng ta có thể hạn chế số nút ẩn; ngăn không cho mạng sử dụng các trọng số lớn; giới hạn số bước luyện.

Bảng 3: *Kết quả đạt được*

Tên mạng	Thời gian huấn luyện sử dụng CPU (s)	Thời gian huấn luyện sử dụng CPU-GPU (s)	Thời gian chạy kiểm tra sử dụng CPU (s)	Thời gian chạy kiểm tra sử dụng CPU-GPU (s)	Độ chính xác của mô hình (%)
MLP Epochs=10 Batch_size=32	987	200	2,1	1	39
MLP Epochs=20 Batch_size=64	1.133	201	2,2	1	45
CNN Epochs=10 Batch_size=32	386	177	2,2	1,3	63
CNN Epochs=20 Batch_size=64	710	256	2,2	1,4	69
CNN Epochs=30 Batch_size=128	1.020	319	2,3	1,4	74
CNN Epochs=40 Batch_size=256	1.194	351	2,3	1,4	79

4. KẾT LUẬN

Bài báo trình bày phương pháp phân loại ảnh sử dụng mạng MLP, mạng CNN và ứng dụng các kỹ thuật tối ưu quá trình huấn luyện mạng. Sau khi xây dựng và thử nghiệm thành công hai mô hình mạng MLP và CNN trên cơ sở sử dụng kết hợp CPU-GPU, bài báo đã đưa ra kết quả so sánh về hiệu quả phân loại ảnh giữa trường hợp khi sử dụng CPU và trường hợp khi sử dụng kết hợp CPU-GPU trong quá trình huấn luyện cũng như kiểm nghiệm mô hình. Thực hiện kiểm nghiệm cho thấy khi sử dụng mạng

CNN cho kết quả phân loại ảnh đạt độ chính xác gần 80% và không cải thiện thêm khi tăng số lần huấn luyện. Để nâng cao kết quả của việc huấn luyện ta cần sử dụng tập huấn luyện có chất lượng ảnh cao hơn cũng như các kỹ thuật khác được trình bày trong [3], [4], [14].

TÀI LIỆU THAM KHẢO

[1] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, *ImageNet classification with deep convolutional neural networks*, Advances in Neural Information Processing Systems, 2012, pp. 1097-1105.

[2] A. Krizhevsky, *Learning Multiple Layers of Features from Tiny Images*. Ph.D dissertation, University of Toronto, 2009.

[3] Baoyuan Liu, Min Wang, Hassan Foroosh, *Marshall Tappen and Marianna Penksy, Sparse Convolutional Neural Networks*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 806-814.

[4] B. Graham, *Sparse 3D convolutional neural networks*, arXiv:1505.02890v2, 2015.

[5] Duchi, J., Hazan, E., & Singer, Y, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization*, Journal of Machine Learning Research, 2011, pp. 2121–2159.

[6] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, arXiv:1207.0580v1, 2012.

[7] Ian Goodfellow, Yoshua Bengio, Aaron Courville, *Deep Learning*, The MIT Press, 2016.

[8] Jayshree Ghorpade, Jitendra Parande, Madhura Kulkarni, Amit Bawaskar, *GPGPU processing in CUDA architecture, Advanced Computing*. An International Journal (ACIJ), Vol.3, No.1, 2012.

[9] John Duchi, Elad Hazan, and Yoram Singer, *Adaptive Subgradient Methods for Online Learning and Stochastic Optimization.* Journal of Machine Learning Research, 2012.

[10] Đinh Văn Nam, Phan Văn Dư, Hồ Sỹ Phương, *Nghiên cứu và thử nghiệm thiết kế thiết bị tự động đọc ghi dữ liệu các máy hiển thị số trên cơ sở mạng nơ ron nhân tạo*, Hội nghị toàn quốc về cơ điện tử lần thứ 8, 2016**,** tr. 96-102.

[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, Journal of machine learning research 15, 2014, pp. 1929-1958.

[12] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei, *ImageNet Large Scale Visual Recognition Challenge*, International Journal of Computer Vision, Volume 115, Issue 3, 2015, pp. 211–252.

[13] Sebastian Ruder, *An overview of gradient descent optimization algorithms*, arXiv:1609.04747, 2017.

[14] Sergey Zagoruyko, Nikos Komodakis, *Wide Residual Networks, Computer Vision and Pattern Recognition*, arXiv:1605.07146, 2017.

[15] https://www.cs.toronto.edu/~kriz/cifar.html.

[16] http://yann.lecun.com/exdb/mnist/

[17] http://ruder.io/optimizing-gradient descent.

SUMMARY

APPLYING NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS AND COMBINATION OF CPUS AND GPUS TO INCREASE CALCULATING PERFORMANCE FOR IMAGE CLASSIFICATION

This paper presents and compares the image classification methods based on MLPs and CNNs. Training data is 500,000 pictures of 10 different objects. The first architecture to be used is MLPs network that contains 3,853,298 weights, the second architecture is CNNs with 528,054 weights. This paper proposes several methods and architectures network to avoid overfitting phenomenon and increases the accuracy of modeling approximately 80%. Besides on it, the paper also presents and compares time training of models using CPUs, and combining CPUs with GPUs.