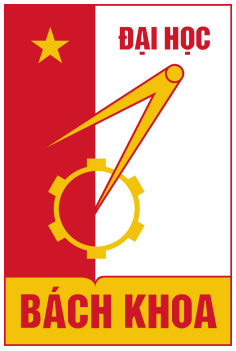
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG



BÁO CÁO BÀI TẬP LỚN

Môn học: Học máy

Đề tài: Nhận Dạng Cảm Xúc Sử Dụng Mạng Nơ-ron Tích Chập Kết Nối Đầy Đủ

Nhóm sinh viên thực hiện :

Lê Trần Bảo Cương - MSSV: 20140542
Nguyễn Huy Hoàng - MSSV: 20141773
Trần Trung Hiếu - MSSV: 20141567
Phan Vũ Hồng Hải - MSSV: 20141394
Phạm Bá Toàn - MSSV: 20144552

Giáo viên hướng dẫn : **TS. Thân Quang Khoát**

HÀ NỘI
Ngày 15 tháng 11 năm 2017

Mục lục

Lời nói đầu

Phân tích biểu hiện khuôn mặt đóng một vai trò vô cùng quan trọng trong việc phân tích cảm xúc và hành vi con người. Tuy nhiên để máy tính có thể phân tích biểu hiện trên khuôn mặt người với độ chính xác cao là một thách thức không hề nhỏ đối với ngành thị giác máy tính. Bài toán phân loại cảm xúc thực hiện phân loại 7 cảm xúc chính của con người: giận dữ, kinh tởm, sợ hãi, vui vẻ, buồn, ngạc nhiên, bình thường. Đây là bài toán có rất nhiều ứng dụng thực tế, ví dụ như: phân tích hành vi trong các cơ quan an ninh; phân tích cảm xúc của khách hàng từ đó xem xét chất lượng dịch vụ đã tốt hay chưa. Đặc biệt trong công nghệ robot, nếu có thể nhận diện cảm xúc của người đối diện thì robot có thể đưa ra những ứng xử phù hợp.

Đã có nhiều nghiên cứu được triển khai nhưng kết quả thu được chưa có nhiều bước đột phá. Ngày nay, với sự phát triển của kỹ thuật học sâu, bài toán nhận dạng cảm xúc trên khuôn mặt người đã có nhiều tiến triển đáng chú ý. Trong những năm trở lại đây, người ta đã đề xuất rất nhiều các mô hình học sâu khác nhau để giải quyết các bài toán trong lĩnh vực thị giác máy tính, đặc biệt là các mô hình mạng nơ-ron tích chập (Deep Convolutional Neural Networks). Mô hình mới nhất hiện nay là mô hình mạng nơ-ron tích chập kết nối đầy đủ (DenseNet) được đề xuất tại hội nghị CVPR 2017. DenseNet cho kết quả tốt hơn tất cả các mô hình trước đây trên các bộ dữ liệu quen thuộc như CIFAR hay ImageNet.

Với mục đích nghiên cứu để có thể sử dụng DenseNet trong các dự án thực tế, trong khuôn khổ bài tập lớn môn học máy, chúng em tiến hành phân tích cấu trúc mạng, đồng thời lập trình mạng DenseNet giải quyết bài toán nhận dạng cảm xúc.

Do trình độ và kinh nghiệm còn non kém nên trong nghiên cứu này có thể còn nhiều sai sót. Nhóm chúng em rất mong nhận được sự phản hồi từ thầy để hoàn thiện hơn nữa công trình của nhóm. Chúng em xin trân trọng cám ơn

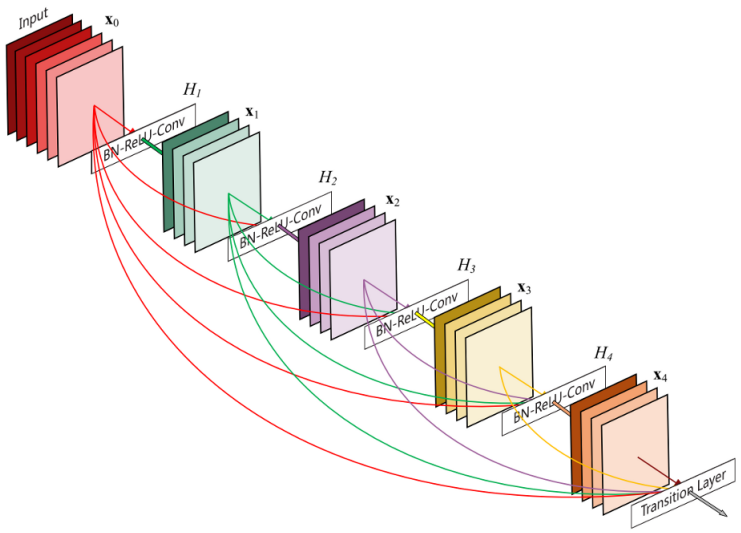
Nhóm sinh viên

Chương 1

Giới thiệu về DenseNet

Trong chương này, chúng ta sẽ cùng nghiên cứu và phân tích cấu trúc của mạng nơ-ron tích chập kết nối đầy đủ [4], qua đó thấy được sự khác biệt trong thiết kế của mạng này so với các mạng nơ-ron tích chập trước đây.

1.1 Tổng quan

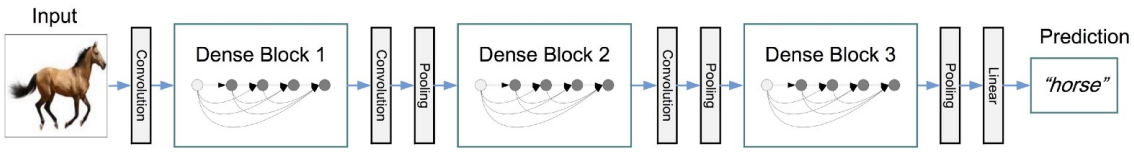


Hình 1.1: Kiến trúc tổng quát DenseNet

Như chúng ta đã biết, trong các mạng nơ-ron tích chập trước đây, đầu ra của các lớp trước thường không được sử dụng trong tính toán ở các lớp phía sau, điều này có thể gây mất mát một lượng nào đó thông tin. Để đảm bảo luồng thông tin luôn được truyền trong suốt quá trình tính toán của mạng, DenseNet kết nối trực tiếp tất cả các lớp của mạng với nhau, tạo thành một mạng nơ-ron tích chập kết nối đầy đủ. Hình 1.1 mô tả kiến trúc tổng quan của DenseNet. Lớp tích chập thứ l của mạng sẽ nhận đầu vào là đúng l khối, bao gồm khối mô tả hình ảnh đầu vào và $l-1$ khối đầu ra của $l-1$ lớp

trước đó. Sau đó, đầu ra của lớp l lại được gộp với đầu vào của chính nó để chuyển tới các lớp tiếp theo. Như vậy với một mạng gồm L lớp, chúng ta có $\frac{L(L+1)}{2}$ kết nối. Quá trình tính toán trong từng lớp của mạng sẽ được trình bày trong phần tiếp theo.

1.2 Kiến trúc DenseNet



Hình 1.2: Các khối chính trong DenseNet

Các khối trong mạng nơ-ron tích chập kết nối đầy đủ được thể hiện trên hình 1.2. Ảnh đầu vào được đẩy qua một lớp tích chập khởi đầu, sau đó là một vài khối dày đặc (Dense Block), giữa các khối dày đặc này là một khối pooling. Số lượng các khối dày đặc và pooling phụ thuộc vào kích thước của ảnh đầu vào. Sau đây chúng ta sẽ tìm hiểu về từng khối này.

1.2.1 Dense block

Mỗi dense block bao gồm l lớp tích chập. Gọi x_i là đầu ra của lớp tích chập thứ i, coi x_0 là đầu vào của dense block (hay chính là đầu ra của dense block trước đó). Khi đó, x_i được tính như sau:

$$x_i = H_i([x_0, x_1, ..., x_{l-1}]), \tag{1.1}$$

trong đó $[x_0, x_1, ..., x_{l-1}]$ là khối kết hợp (theo chiều sâu) của các khối $x_0, x_1, ..., x_{l-1}$, $H_i()$ là một hàm kích hoạt mà ta gọi là *composite function*. Nhìn vào công thức (1.1) ta thấy rằng đầu ra của lớp thứ l được tính toán từ đầu ra của tất cả các lớp phía trước

1.2.2 Composite function

Composite function là hàm kích hoạt được sử dụng trong các lớp tích chập của DenseNet. Thông thường, trong các mạng nơ-ron tích chập trước đây, tại lớp tích chập ta có 3 phép toán được thực hiện theo thứ tự: convolution, batch normalization [5] và hàm kích hoạt ReLU. Tuy nhiên trong bài báo nghiên cứu về *shortcut connection*[3], Kaiming He và cộng sự đã đề xuất thay đổi thứ tự các phép toán trong lớp tích chập: batch normalization, hàm kích hoạt ReLU và convolution. Sự kết hợp của 3 phép toán theo thứ tự trên được gọi là composite function. Ngoài ra tại phép toán tích chập (convolution) ta sử dụng filter với kích thước 3×3, số lượng filter sử dụng sẽ được đề cập ở các phần tiếp theo.

1.2.3 Transition block

Một phần thiết yếu trong tất cả các mạng nơ-ron tích chập đó là lớp pooling. Lớp pooling có vai trò tổng hợp thông tin và giảm kích thước đầu ra của lớp tích chập phía trước. Tuy nhiên trong DenseNet, tác giả đã thay đổi lớp pooling, xây dựng một kiến trúc gọi là *transition block*. Transition block được đặt ngay sau mỗi dense block, bao gồm một lớp tích chập với filter kích thước 1×1, theo sau đó là một lớp average pooling với filter kích thước 2×2 và độ trượt filter (stride) bằng 2. Lớp convolution với filter kích thước 1×1 là một cấu trúc đặc biệt, số lượng filter của lớp tích chập này cũng sẽ được đề cập ở các phần tiếp theo. Ngoài ra, thay vì việc chỉ sử dụng đơn thuần phép toán tích chập, ta có thể sử dụng *composite function* với phép toán tích chập có kích thước filter là 1×1. Ở transition block sau dense block cuối cùng, thay vì sử dụng average pooling với filter kích thước 2×2, ta sử dụng global average pooling, theo sau đó là một lớp kết nối đầy đủ gồm n nơ-ron, với n là số lớp cần phân loại. Việc sử dụng global average pooling tương tự như một regularization [6], nó ép cho các tham số trước đó của mạng đều phải học tốt thay vì phụ thuộc nhiều vào bộ phân loại.

1.2.4 Growth rate

Xét dense block thứ i, gọi k_0 là độ sâu của khối đầu ra từ transition layer $i-1$ (nếu $i = 1$ thì k_0 chính là số kênh của ảnh đầu vào, với ảnh RGB ta có $k_0 = 3$, với ảnh xám ta có $k_0 = 1$). Nếu trong mỗi *composite function*, ở phép toán tích chập ta sử dụng k filter, thì đầu vào của *composite function* thứ l trong dense block này sẽ có độ sâu là $k_0 + (l-1) \times k$. Tác giả gọi tham số k là *hệ số tăng* hay *growth rate*. Chúng ta có thể hiểu hệ số k chính là số lượng filter trong phép toán tích chập ở mỗi *composite function*. Điểm khác biệt rất quan trọng giữa DenseNet và các mạng nơ-ron tích chập trước đó là nó chỉ sử dụng tham số k rất nhỏ, trong các phiên bản thông thường, người ta thường sử dụng $k = 12$. Trong những phiên bản mở rộng hơn như *wide densenet* thì ta sử dụng $k = 32$ hoặc $k = 48$. Điều này làm số lượng tham số cần học của DenseNet nhỏ hơn rất nhiều so với các thế hệ mạng trước đây. Hệ số k này quyết định lượng thông tin của mỗi lớp đóng góp vào lượng thông tin chung được mang đi xuyên suốt mạng. Chính vì thông tin từ các lớp được mang theo xuyên suốt mạng nên ta không cần sử dụng quá nhiều filter như các mạng nơ-ron tích chập khác.

1.2.5 Bottleneck layers

Ta thấy rằng, vì tính chất của DenseNet, đầu vào của *composite function* càng về sau càng lớn, theo cấp số cộng với công sai chính bằng hệ số tăng k, điều này làm tăng khối lượng tính toán rất nhiều. Vì vậy trong các bài báo [2] [7], với các mô hình ResNet và Inception, người ta đã đề xuất sử dụng một lớp tích chập với filter kích thước 1×1 và số lượng filter tương đối nhỏ. Cấu trúc này được gọi là nút thắt cổ chai (bottleneck). Lúc này *composite function* sẽ được thiết kế lại như sau: batch normalization - ReLU - bottleneck - batch normalization - ReLU - convolution 3×3. Số lượng filter được sử dụng trong nút thắt cổ chai là $4k$. Người ta gọi mạng DenseNet sử dụng bottleneck là *DenseNet-B*

1.2.6 Compression

Việc sử dụng bottleneck tại composite function chỉ làm giảm khối lượng tính toán chứ không hề thay đổi kích thước đầu ra của dense block (vẫn có độ sâu tăng theo cấp số cộng với công sai k). Để làm tăng tính chặt chẽ của mô hình, người ta xem xét việc giảm độ sâu khối đầu ra của dense block tại transition block. Nếu đầu ra của dense block là khối có độ sâu m, transition block sẽ sinh ra một khối có độ sâu $\lfloor \theta m \rfloor$ với $0 < \theta \leq 1$ gọi là hệ số nén (compression factor). Quá trình nén này được thực hiện dựa vào lớp tích chập với filter kích thước 1×1 trong transition block mà ta đã đề cập ở phần trên. Như vậy ở lớp tích chập này ta sử dụng $\lfloor \theta m \rfloor$ filter. Nếu $\theta = 1$ thì ta không thực hiện việc nén. Nếu có thực hiện bước nén này, thực nghiệm cho thấy giá trị $\theta = 0.5$ là tốt nhất. Mạng DenseNet có thực hiện thao tác nén được gọi là *DenseNet-C*. Mạng kết hợp cả bottleneck và thao tác nén được gọi là *DenseNet-BC*.

Chương 2

Nhận diện cảm xúc với DenseNet

Trong chương này chúng ta sẽ tìm hiểu về bài toán nhận diện cảm xúc và cài đặt mạng DenseNet giải quyết bài toán này. Chúng ta sẽ thử nghiệm một vài phiên bản DenseNet khác nhau để so sánh và đưa ra kết luận.

2.1 Bài toán nhận diện cảm xúc

Nhận diện cảm xúc là một bài toán nằm trong lớp các bài toán về phân tích ảnh khuôn mặt người. Bài toán này tập trung vào 7 loại cảm xúc chính của con người, bao gồm: giận dữ, kinh tởm, sợ hãi, vui vẻ, buồn, ngạc nhiên, bình thường. Tuy nhiên đây không phải là một bài toán đơn giản, ngay cả đối với con người vì có những loại cảm xúc rất khó để phân biệt. Cuộc thi về nhận diện cảm xúc được tổ chức bởi Kaggle.com vào năm 2013 đã thu hút được rất nhiều nhóm tham dự với nhiều phương pháp khác nhau, từ các phương pháp học máy truyền thống đến các phương pháp hiện đại sử dụng kỹ thuật học sâu.

Bộ dữ liệu được sử dụng trong cuộc thi là FER-2013, được chia làm 3 phần: tập dữ liệu huấn luyện gồm 28709 ảnh, tập dữ liệu kiểm tra công khai gồm 3589 ảnh và tập dữ liệu kiểm tra kín gồm 3589 ảnh, tất cả các ảnh đều đã được gán nhãn và ở dạng ảnh xám kích thước 48×48. Kết quả của các phương pháp được đánh giá dựa trên độ chính xác trên tập dữ liệu kiểm tra kín. Hình 2.1 mô tả một số ví dụ trong tập dữ liệu.

2.2 Xây dựng mạng

Với bài toán nhận dạng cảm xúc, nhóm chúng em xây dựng mạng DenseNet gồm 3 dense block. Độ sâu D của mạng có dạng $3L + 4$, ở đó L là số lớp tích chập trong mỗi dense block, 4 lớp còn lại bao gồm 3 transition block và 1 lớp tích chập khởi đầu (trước dense block đầu tiên). Trong khuôn khổ bài tập lớn này chúng em thử nghiệm mô hình với $L = 12$ tương đương với $D = 40$. Với hệ số tăng k, nhóm thử nghiệm 2 phiên bản: phiên bản DenseNet với $k = 12$ và phiên bản WideDenseNet với $k = 32$. Đồng thời nhóm cũng thử nghiệm phiên bản DenseNet-BC và WideDenseNet-BC. Bảng 2.1

Hình 2.1: Một vài ví dụ về bộ dữ liệu FER-2013

chỉ ra thiết kế chi tiết của các mạng DenseNet và DenseNet-BC, hai phiên bản WideDenseNet và WideDenseNet-BC có kiến trúc tương tự với $k = 32$

Bảng 2.1: Cấu hình mạng DenseNet và DenseNet-BC

Layers	Output Size	DenseNet ($L = 40, k = 12$)	DenseNet-BC($L = 40, k = 12$)
Conv	21×21	3×3 conv, $2k$ filters, stride 2	
Dense Block 1	21×21	$\left[\ 3 \times 3conv\ \right] \times 12$	$\left[\begin{matrix} 1 \times 1conv \\ 3 \times 3conv \end{matrix}\right] \times 12$
Transition Block 1	21×21	1×1 conv	1×1 conv, $m\theta$ filters
	10×10	2×2 average pooling, stride 2	
Dense Block 2	10×10	$\left[\ 3 \times 3conv\ \right] \times 12$	$\left[\begin{matrix} 1 \times 1conv \\ 3 \times 3conv \end{matrix}\right] \times 12$
Transition Block 2	10×10	1×1 conv	1×1 conv, $m\theta$ filters
	5×5	2×2 average pooling, stride 2	
Dense Block 3	5×5	$\left[\ 3 \times 3conv\ \right] \times 12$	$\left[\begin{matrix} 1 \times 1conv \\ 3 \times 3conv \end{matrix}\right] \times 12$
Last Transition Block	1×1	5×5 global average pooling	
		$7D$ fully-connected, softmax	

2.3 Tiền xử lý dữ liệu

Do bộ dữ liệu đã được Kaggle xử lý khá tốt: ảnh được cắt bao sát khuôn mặt của người trong hình, ảnh đã ở dạng ảnh xám, vì vậy ở bước tiền xử lý dữ liệu, nhóm em chỉ sử dụng duy nhất bước

normalization: chia tất cả các điểm ảnh cho 255

2.4 Phương pháp thêm dữ liệu

Ta thấy rằng bộ dữ liệu huấn luyện của bài toán nhận dạng cảm xúc chỉ gồm gần 30.000 ảnh. Đây là lượng dữ liệu khả nhỏ, vì vậy để việc huấn luyện mạng được tốt hơn, tránh trường hợp học quá khớp chúng ta cần sử dụng các phương pháp tăng dữ liệu (data augmentation). Đây cũng là một phương pháp rất hay được sử dụng trong các bài toán Thị giác máy tính. Trong trường hợp bài toán nhận dạng cảm xúc, nhóm chúng em thực hiện 3 phương pháp chính để sinh thêm dữ liệu từ tập dữ liệu gốc:

- Phương pháp 1: Cắt vùng ảnh ngẫu nhiên: Từ ảnh trong bộ dữ liệu gốc có kích thước 48×48, ta thực hiện cắt ngẫu nhiên một vùng có kích thước 42×42
- Phương pháp 2: Đối xứng gương: Lấy đối xứng gương của mỗi ảnh với xác suất 0.5
- Phương pháp 3: Xoay ngẫu nhiên: Xoay trái/phải ảnh đầu vào với một góc ngẫu nhiên.

Hình 2.2: Một vài ví dụ về dữ liệu ảnh sau khi thực hiện các phương pháp tăng dữ liệu

Qua thực nghiệm chúng em thấy rằng việc áp dụng các phương pháp tăng dữ liệu nói trên giúp cho cải thiện rất tốt hiệu năng của mô hình. Hình 2.2 mô tả một số ví dụ của phương pháp này.

2.5 Huấn luyện mạng

Với mỗi nhóm dữ liệu (batch), trước khi truyền vào mạng, chúng em thực hiện ngẫu nhiên các bước mở rộng dữ liệu đề cập ở trên. Điều này giúp bộ dữ liệu huấn luyện được phong phú hơn. Mạng được huấn luyện bằng cách cực tiểu hàm mất mát sử dụng thuật toán lan truyền ngược và phương pháp tối ưu Stochastic Gradient Descent [1]. Hàm mất mát được sử dụng là cross-entropy:

$$H = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y_i}) + (1-y_i) log(1-(\hat{y_i}))), \tag{2.1}$$

ở đó

- N là số mẫu trong tập huấn luyện
- y_i là nhãn đúng được mã hóa dưới dạng one-hot đối với ảnh thứ i
- $\hat{y_i}$ là phân bố xác suất dự đoán của mạng đối với ảnh thứ i

Ngoài ra trong quá trình huấn luyện mạng nơ-ron, khi giá trị của hàm mất mát bắt đầu bão hòa, ta cần giảm hệ số học (learning rate) để tiếp tục cực tiểu hóa hàm mất mát. Trong quá trình thực nghiệm, khi thấy giá trị hàm mất mát không giảm sau một vài vòng huấn luyện, chúng em giảm hệ số học đi 10 lần.

2.6 Đánh giá mô hình

Như đã đề cập ở trên, Kaggle đã chia bộ dữ liệu FER-2013 thành 3 phần: tập dữ liệu huấn luyện, tập dữ liệu kiểm tra công khai và tập dữ liệu kiểm tra kín. Trong quá trình thực nghiệm, chúng em huấn luyện mô hình trên tập dữ liệu huấn luyện, điều chỉnh tham số của mô hình thông qua tập dữ liệu kiểm tra công khai (coi tập kiểm tra công khai là tập dữ liệu phát triển - validation set) và đánh giá độ chính xác của mô hình thông qua tập dữ liệu kiểm tra kín.

Ngoài cách đánh giá truyền thống là sử dụng mô hình tốt nhất cho tập dữ liệu phát triển, nhóm chúng em thực hiện thêm việc đánh giá bằng cách sử dụng kỹ thuật *Ensemble Learning*. Với kĩ thuật này, ta sẽ lưu lại k mô hình tốt nhất trên tập dữ liệu phát triển. Với mỗi dữ liệu trong tập kiểm tra kín, ta sử dụng cả k mô hình này để đưa ra dự đoán, kết quả dự đoán cuối cùng sẽ là **trung bình cộng** của k dự đoán. Thực nghiệm cho thấy rằng phương pháp này giúp cải thiện độ chính xác của mô hình.

2.7 Chi tiết cài đặt

Ngôn ngữ lập trình được sử dụng là Python 3.5.2, sử dụng thư viện lập trình chính là TensorFlow, ngoài ra có sử dụng một số thư viện cơ bản về xử lý số học, vẽ đồ thị, thao tác file,... như: numpy, pandas, motplotlib, sklearn, scipy, OpenCV,... Mã nguồn cài đặt bao gồm các file sau:

- *read_ data.py:* Đây là file đảm nhiệm chức năng đọc dữ liệu từ tệp dữ liệu của Kaggle. Từ file .csv, ta trích xuất các thông tin về ảnh bao gồm: mảng các giá trị pixel của ảnh, nhãn tương ứng của ảnh. Trong file này ta cũng thực hiện bước normalize dữ liệu, chia tất cả các pixel của ảnh cho 255. Cuối cùng ta lưu dữ liệu vào 3 file nhị phân: *train_ data.npy, public_ test_ data.npy, private_ test_ data.npy.* Dữ liệu chứa trong các file này là các *list*, mỗi phần tử của list là một *pair* (image, label).

- *DenseNet_ input.py:* Đây là file đọc dữ liệu từ các file nhị phân do file *read_ data.py* tạo ra. Đồng thời file này chứa các hàm augmentation phục vụ cho phương pháp tăng dữ liệu đã trình bày ở trên.

- *DenseNet.py:* Đây là file cài đặt mạng DenseNet sử dụng TensorFlow. File này chứa lớp DenseNet gồm các tham số đầu vào chính như sau:

  - *train_ data, valid_ data, test_ data:* Kiểu dữ liệu: list, chứa các tập dữ liệu huấn luyện, phát triển và kiểm tra, mỗi phần tử của list là một *pair* (image, label).
  - *num_ channel:* Kiểu dữ liệu: số nguyên, số chiều của ảnh, ở đây mặc định là ảnh xám.
  - *input_ size:* Kiểu dữ liệu: số nguyên, kích thước của ảnh đầu vào.
  - *num_ class:* Kiểu dữ liệu: số nguyên, số lớp đầu ra của bài toán.
  - *reduce_ lnr:* Kiểu dữ liệu: list, các thời điểm giảm learning rate.
  - *growth_ rate:* Kiểu dữ liệu: số nguyên, hệ số tăng.
  - *depth:* Kiểu dữ liệu: số nguyên, chiều sâu của mạng.
  - *total_ blocks:* Kiểu dữ liệu: số nguyên, số Dense block sử dụng.
  - *reduction:* Kiểu dữ liệu: số thực, hệ số nén.
  - *bc_ mode:* Kiểu dữ liệu: boolean, có sử dụng kiến trúc BC hay không?

  Lớp DenseNet có một vài phương thức quan trọng như:

  - *_ input()*: Định nghĩa đầu vào của mạng.
  - *inferences()*: Định nghĩa các lớp của mạng.
  - *losses()*: Định nghĩa các hàm mất mát sử dụng.
  - *train_ step()*: Định nghĩa training operator.
  - *init_ session()*: Khởi tạo phiên làm việc mới, có thể tải lại dữ liệu từ bản sao lưu.
  - *train()*: Huấn luyện mạng.
  - *test()*: Đánh giá hiệu năng của mạng trên tập test.
  - *test_ snapshot_ ensemble()* Đánh giá hiệu năng bằng phương pháp ensemble.

- *train.py:* Huấn luyện mạng DenseNet
- *ensemble.py:* Kiểm tra hiệu năng, vẽ confusion matrix.

Chương trình được chạy trên máy tính có cấu hình: CPU AMD Ryzen 1600X, 16GB RAM, GPU GTX 1070 8GB.

2.8 Hướng dẫn sử dụng

- Bước 1: Trích xuất dữ liệu từ file dữ liệu gốc của cuộc thi: Chạy file read_data.py.
- Bước 2: Huấn luyện mạng: Chỉnh tham số trong file train.py. Lưu ý: cần tự tạo các thư mục ứng với các tham số logs_folder, current_save_folder, valid_save_folder. Sau đó chạy file train.py để bắt đầu huấn luyện mạng.
- Bước 3: Đánh giá mạng: Chỉnh tham số trong file ensemble.py tương tự như file train.py. Tham số save_file chứa các model muốn sử dụng. Ta sử dụng các model lưu trong thư mục ứng với tham số valid_save_folder, các file lưu model có dạng modelX.ckpt với X là độ chính xác trên tập valid của model, ví dụ model0.701.ckpt, model0.68.ckpt,.... Sử dụng nhiều model tương đương với phương pháp ensemble learning.

2.9 Kết quả thực nghiệm

Nhóm chúng em đưa ra kết quả thử nghiệm trên cả 4 mô hình: DenseNet, DenseNet-BC, WideDenseNet, WideDenseNet-BC. Với cách đánh giá truyền thống, không sử dụng ensemble learning, hiệu năng của mô hình trên tập dữ liệu kiểm tra được đánh giá bằng mô hình cho kết quả tốt nhất trên tập dữ liệu phát triển. Với cách đánh giá sử dụng kĩ thuật ensemble learning, ta sử dụng 3 mô hình cho kết quả tốt nhất trên tập phát triển. Các kết quả thực nghiệm được trình bày trong bảng 2.3

Bảng 2.2: Số lượng tham số của các phiên bản DenseNet

Mô hình	Số lượng tham số
DenseNet	1.1M
DenseNet-BC	0.5M
WideDenseNet	7.4M
WideDenseNet-BC	3.4M

Bảng 2.3: Kết quả thực nghiệm

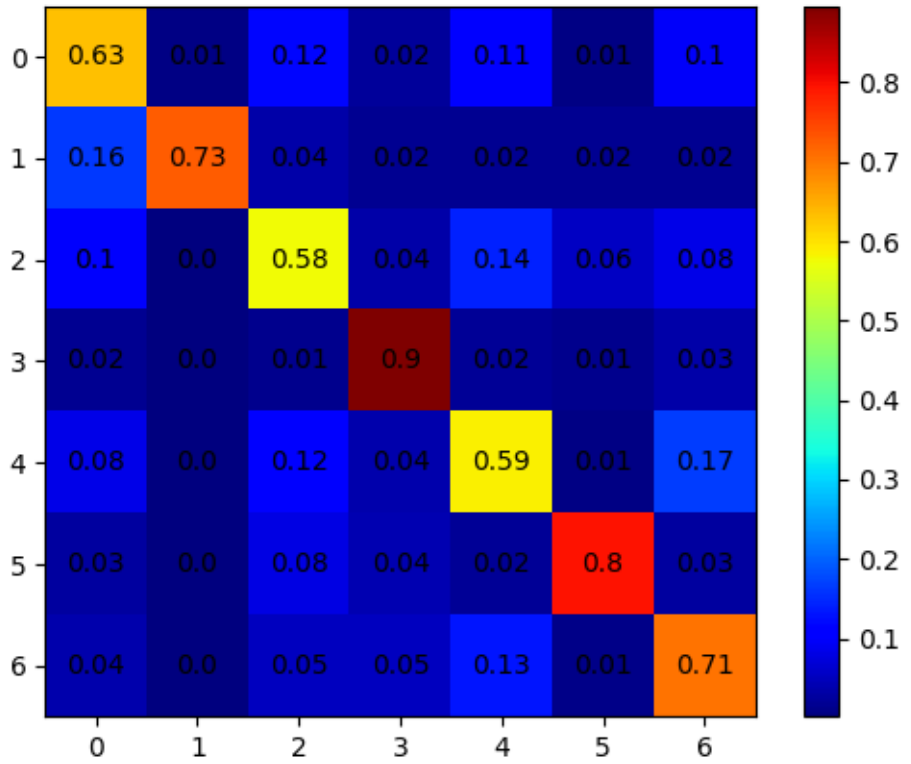
Mô hình	Không sử dụng ensemble learning		Sử dụng ensemble learning	
	Valid	Test	Valid	Test
DenseNet	68.83%	69.77%	-	69.95%
DenseNet-BC	69.45%	70.69%	-	71.02%
WideDenseNet	70.09%	70.55%	-	70.86%
WideDenseNet-BC	70.12%	**71.52%**	-	**71.86%**

Để đánh giá độ tốt của mô hình, ta sẽ so sánh hiệu suất của chúng với các nhóm có thứ hạng cao tại cuộc thi của Kaggle. Bảng 2.4 đưa ra kết quả của các phương pháp khác nhau trên tập dữ liệu kiểm tra kín.

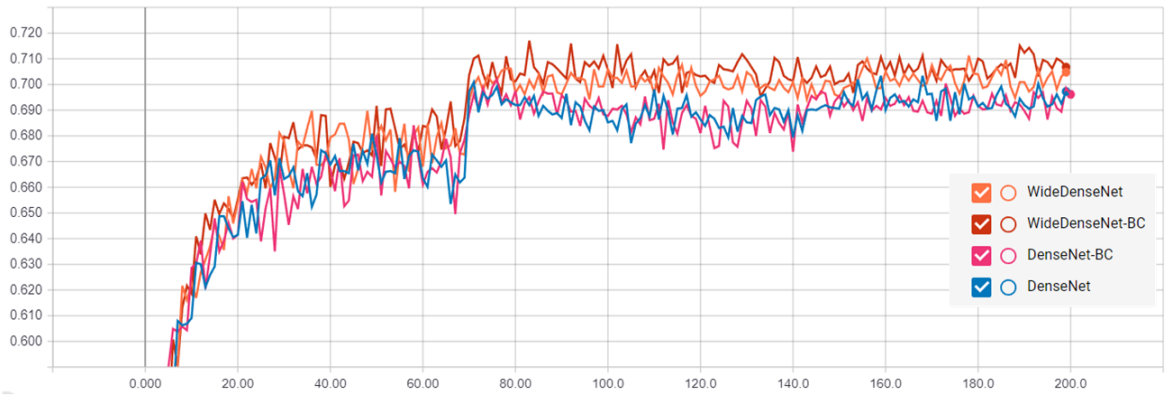
Bảng 2.4: So sánh với các kết quả trước đây

Method	Private test accuracy
CNN (team Maxim Milakov - rank 3 Kaggle)	68.82%
CNN (team Unsupervised - rank 2 Kaggle)	69.27%
CNN + SVM (team RBM - rank 1 Kaggle)	71.16%
DenseNet	69.77%
DenseNet + ensemble	69.95%
DenseNet-BC	70.69%
DenseNet-BC + ensemble	71.02%
WideDenseNet	70.55%
WideDenseNet + ensemble	70.86%
WideDenseNet-BC	**71.52%**
WideDenseNet-BC + ensemble	**71.86%**

Hình 2.3 cho ta thấy confusion matrix ứng với trường hợp sử dụng mạng WideDenseNet-BC kết hợp với ensemble learning. Nhìn vào ma trận này có thể thấy mô hình cho kết quả rất tốt với các cảm xúc như vui vẻ, ngạc nhiên.



Hình 2.3: Confusion matrix cho trường hợp WideDenseNet-BC sử dụng ensemble learning. Các nhãn tương ứng: 0 - Giận dữ, 1 - Kinh tởm, 2 - Sợ hãi, 3 - Vui vẻ, 4 - Buồn, 5 - Ngạc nhiên, 6 - Bình thường. Các nhãn trên trục hoành là nhãn đúng của ảnh, nhãn trên trục tung là nhãn dự đoán bởi mạng



Hình 2.4: Đồ thị biểu diễn độ chính xác trên tập test trong quá trình huấn luyện của 4 mô hình

Kết luận

Qua nghiên cứu này chúng ta có thể thấy được sức mạnh của mạng nơ-ron tích chập kết nối đầy đủ trong các bài toán nhận dạng cảm xúc nói riêng và các bài toán về thị giác máy tính nói chung. Số lượng tham số và thời gian huấn luyện của DenseNet là tương đối nhanh so với các thế hệ mạng nơ-ron trước đây. Đồng thời, dựa vào các kết quả đánh giá thực nghiệm đối với bài toán Nhận diện cảm xúc ta có thể thấy rằng các mô hình WideDenseNet cho kết quả tốt hơn so với mô hình DenseNet truyền thống và mô hình sử dụng bottleneck (BC) cũng cho kết quả tốt hơn so với mô hình không sử dụng bottleneck.

DenseNet là một kiến trúc mạng mới nhưng mô hình mạng tương đối tường minh và dễ hiểu, đồng thời việc lập trình cấu trúc mạng với các thư viện có sẵn không phải quá phức tạp. Vì vậy chúng ta hoàn toàn có thể áp dụng DenseNet vào các bài toán liên quan đến thị giác máy tính trong thực tế để đạt được những kết quả mong muốn.

Tài liệu tham khảo

[1] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[4] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[5] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[6] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.