

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN

NGHIÊN CỨU CÁC KỸ THUẬT HIỆU QUẢ ĐỂ
NÉN DỮ LIỆU ĐA PHƯƠNG TIỆN

Sinh viên thực hiện: **1. Phạm Hồng Thái**

2. Đinh Thị Thuyên

Lớp: 2017DHCNTT05

2017DHCNTT02

Khoa: Công nghệ thông tin

Người hướng dẫn: **TS. Nguyễn Thị Mỹ Bình**

Hà Nội, 05/2021

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



**BÁO CÁO TỔNG KẾT
ĐỀ TÀI NGHIÊN CỨU KHOA HỌC CỦA SINH VIÊN
NGHIÊN CỨU CÁC KỸ THUẬT HIỆU QUẢ ĐỂ
NÉN DỮ LIỆU ĐA PHƯƠNG TIỆN**

Sinh viên thực hiện: **1. Phạm Hồng Thái** Nam
2. Đinh Thị Thuyên Nữ
Dân tộc: Kinh
Lớp: 2017DHCNTT05 Năm thứ: 4/4
2017DHCNTT02
Khoa: Công nghệ thông tin
Ngành học: Công nghệ thông tin
Người hướng dẫn: **TS. Nguyễn Thị Mỹ Bình**

Hà Nội, 05/2021

Lời cảm ơn

Lời đầu tiên, nhóm nghiên cứu chúng em xin bày tỏ sự cảm ơn chân thành đối với Cô giáo, TS. Nguyễn Thị Mỹ Bình – giáo viên hướng dẫn trực tiếp em.

Chúng em cũng xin gửi lời cảm ơn tới các thầy cô trong khoa Công nghệ thông tin, trường Đại học Công Nghiệp Hà Nội đã hướng dẫn, chỉ bảo và tạo điều kiện cho em học tập cũng như nghiên cứu trong thời gian qua.

Cảm ơn Câu lạc bộ HIT, Đội Olympic Tin học khoa Công nghệ thông tin đã đồng hành cùng chúng em trong suốt quãng thời gian học tập, làm việc tại trường.

Mặc dù đã cố gắng hoàn thành báo cáo nghiên cứu này nhưng chắc chắn sẽ không tránh khỏi những sai sót, chúng em kính mong nhận được sự thông cảm và chỉ bảo của các thầy cô và các bạn.

Mục lục

MỞ ĐẦU	4
0.1 Lý do chọn đề tài	4
0.2 Mục đích của đề tài	4
0.3 Đối tượng và phạm vi nghiên cứu của đề tài	5
0.4 Bảng các ký hiệu	6
1 Nghiên cứu tổng quan	7
1.1 Các phương pháp nghiên cứu	7
1.2 Ưu nhược điểm của các phương pháp	7
2 Cơ sở lý thuyết	10
2.1 Tổng quan về các kĩ thuật nén mất mát thông tin	10
2.2 Các kĩ thuật nén codec thường được sử dụng	11
2.3 Mạng nơ-ron nhân tạo (Artificial Neural Network - ANN)	14
2.4 Mô hình mạng neural tích chập (Convolutional neural network - CNN) ..	15
2.5 Bộ mã hóa tự động (Autoencoder)	21
2.6 Ngôn ngữ lập trình Python và thư viện PyTorch	27

3 Triển khai chương trình và đánh giá hiệu suất	29
3.1 Triển khai chương trình	29
3.2 Đánh giá hiệu suất nén	37
4 Kết luận và hướng phát triển	39
4.1 Kết luận	39
4.2 Hướng phát triển	39
Tài liệu tham khảo	41

MỞ ĐẦU

0.1. Lý do chọn đề tài

Với sự phát triển không ngừng của khoa học và công nghệ, đặc biệt là các định dạng phương tiện mới công nghệ phần cứng thay đổi, cũng như các yêu cầu và loại nội dung đa dạng tạo ra nhu cầu về các thuật toán nén có giá trị cao hơn các phương pháp nén hiện tại (codec). Chúng em lựa chọn đề tài này nhằm tìm kiếm, xây dựng, triển khai các phương pháp nén mới tận dụng sự mạnh mẽ của phần cứng ngày nay.

Những tiến bộ trong việc đào tạo mạng nơ-ron đã giúp cải thiện hiệu suất trong một số nguồn cung cấp, nhưng mạng nơ-ron vẫn chưa vượt qua các codec hiện có trong việc nén hình ảnh mất dữ liệu. Các kết quả đầu tiên đã đạt được gần đây bằng cách sử dụng bộ mã hóa tự động - đặc biệt là trên các hình ảnh nhỏ - và mạng nơ-ron đã đạt được kết quả tiên tiến nhất trong việc nén hình ảnh không mất dữ liệu.

0.2. Mục đích của đề tài

- Phân tích các kĩ thuật nén cũ (codec)
- Xây dựng, tìm kiếm các kĩ thuật, phương pháp mới sử dụng trong nén dữ liệu đa phương tiện hiệu quả
- Tận dụng sự mạnh mẽ của phần cứng.

0.3. Đối tượng và phạm vi nghiên cứu của đề tài

0.3.1. Đối tượng

- Các kĩ thuật nén thường được sử dụng (DCT, Huffman, kmean, ...)
- Các phương pháp nén được phát triển và thể hiện tính hiệu quả trong thời gian gần đây
- Bộ mã hóa tự động (Autoencoder)
- Các phương pháp phương pháp đánh giá hiệu năng nén
- Lấy ảnh làm đại diện cho dữ liệu, vì tiện lợi cho việc kiểm tra tính tương đối của dữ liệu gốc và giữ liệu sau khi trải qua quá trình nén, giải mã

0.3.2. Phạm vi nghiên cứu

- Nghiên cứu tập trung chủ yếu các kĩ thuật nén có mất mát dữ liệu.
- Tìm kiếm các phương pháp, kĩ thuật nén liên quan đến học máy, học sâu để tận dụng khả năng của phần cứng.

0.4. Bảng các ký hiệu

Các ký hiệu sử dụng trong sách được liệt kê trong Bảng 0.1.

Bảng 0.1: Các quy ước ký hiệu và tên gọi được sử dụng trong báo cáo

Ký hiệu	Ý nghĩa
x, y, N, k	in nghiêng, thường hoặc hoa, là các số vô hướng
\mathbf{x}, \mathbf{y}	in đậm, chữ thường, là các vector
\mathbf{X}, \mathbf{Y}	in đậm, chữ hoa, là các ma trận
\mathbb{R}	tập hợp các số thực
\mathbb{N}	tập hợp các số tự nhiên
\mathbb{C}	tập hợp các số phức
\mathbb{R}^m	tập hợp các vector thực có m phần tử
$\mathbb{R}^{m \times n}$	tập hợp các ma trận thực có m hàng, n cột
\mathbb{S}^n	tập hợp các ma trận vuông đối xứng bậc n
\mathbb{S}_+^n	tập hợp các ma trận nửa xác định dương bậc n
\mathbb{S}_{++}^n	tập hợp các ma trận xác định dương bậc n
\in	phần tử thuộc tập hợp
\exists	tồn tại
\forall	mọi
\triangleq	ký hiệu là/bởi. Ví dụ $a \triangleq f(x)$ nghĩa là “ký hiệu $f(x)$ bởi a ”.
x_i	phần tử thứ i (tính từ 1) của vector \mathbf{x}
$\text{sgn}(x)$	hàm xác định dấu. Bằng 1 nếu $x \geq 0$, bằng -1 nếu $x < 0$.
$\exp(x)$	e^x
$\log(x)$	logarit <i>tự nhiên</i> của số thực dương x
$\underset{x}{\text{argmin}} f(x)$	giá trị của x để hàm $f(x)$ đạt giá trị nhỏ nhất
$\underset{x}{\text{argmax}} f(x)$	giá trị của x để hàm $f(x)$ đạt giá trị lớn nhất
o.w	<i>otherwise</i> – trong các trường hợp còn lại
$\frac{\partial f}{\partial x}$	đạo hàm của hàm số f theo $x \in \mathbb{R}$
$\nabla_{\mathbf{x}} f$	gradient của hàm số f theo \mathbf{x} (\mathbf{x} là vector hoặc ma trận)
$\nabla_{\mathbf{x}}^2 f$	gradient bậc hai của hàm số f theo \mathbf{x} , còn được gọi là <i>Hesse</i>
\odot	Hadamard product (elementwise product). Phép nhân từng phần tử của hai vector hoặc ma trận cùng kích thước.
\propto	tỉ lệ với
$\rule[1.5ex]{0pt}{0pt}$	đường nét liền
$\rule[-0.5ex]{0pt}{0pt}$	đường nét đứt
$\dots\dots$	đường nét chấm (đường chấm chấm)
$\cdots\cdots$	đường chấm gạch
	nền chấm
	nền sọc chéo

Chương 1

Nghiên cứu tổng quan

1.1. Các phương pháp nghiên cứu

Hiện nay nén dữ liệu được chia thành 2 loại:

- Nén không mất mát thông tin
- Nén có mất mát thông tin

1.2. Ưu nhược điểm của các phương pháp

1.2.1. Nén không mất mát thông tin:

Các thuật toán nén không mất dữ liệu thường dựa trên giả thuyết dư thừa trong dữ liệu và thể hiện dữ liệu chính xác hơn mà không mất các thông tin. Nén mà không làm mất dữ liệu là khả thi vì tất cả các dữ liệu thực tế đều có dư thừa. Ví dụ một hình ảnh có thể có các vùng màu sắc không thay đổi trong nhiều pixel. Thay vì ghi nhận từng pixel như đỏ, đỏ, đỏ... dữ liệu có thể được ghi là 279 điểm ảnh đỏ liên tiếp. Đây là một ví dụ về run-length encoding; ngoài ra còn có rất nhiều giải thuật khác.

Dựa theo mức áp dụng thuật toán nén người ta chia nén thành các dạng sau:

- Nén tệp tin: Đây là dạng thức nén truyền thống và thuật toán nén được áp dụng cho từng tệp tin riêng lẻ. Tuy vậy nếu 2 tệp tin giống nhau thì vẫn được nén 2 lần và được ghi 2 lần. Chỉ các byte trùng lặp trong 1 file được loại trừ

để giảm kích thước. Tùy dữ liệu nhưng thông thường khả năng giảm sau khi nén chỉ từ 2-3 lần.

- Loại trừ trùng lặp file: Đây là dạng thức nén mà thuật toán nén được áp dụng cho nhiều tập tin. Các file giống hệt nhau sẽ chỉ được lưu một lần. Ví dụ một thư điện tử có tệp tin đính kèm được gửi cho 1000 người. Chỉ có một bản đính kèm được lưu và vì vậy có thể giảm khá nhiều. Thông thường có thể giảm từ 5-10 lần so với dữ liệu gốc. - Loại trừ trùng lặp ở mức sub-file: Đây là một dạng thức kết hợp cả nén tệp tin và loại trừ trùng lặp

1.2.2. Nén có mất mát dữ liệu:

Nén mất dữ liệu giảm số lượng bit bằng cách xác định các thông tin không cần thiết và loại bỏ chúng.

Chuẩn nén tín hiệu số gồm có các chuẩn sau:

- Chuẩn MJPEG: Đây là một trong những chuẩn cổ nhất mà hiện nay vẫn sử dụng. MJPEG (Morgan JPEG). Chuẩn này hiện chỉ sử dụng trong các thiết bị DVR rẻ tiền, chất lượng thấp. Không những chất lượng hình ảnh kém, tốn tài nguyên xử lý, cần nhiều dung lượng ổ chứa, và còn hay làm lỗi đường truyền.
- Chuẩn MPEG2: Chuẩn MPEG là một chuẩn thông dụng. Đã được sử dụng rộng rãi trong hơn một thập kỉ qua. Tuy nhiên, kích thước file lớn so với những chuẩn mới xuất hiện gần đây, và có thể gây khó khăn cho việc truyền dữ liệu.
- Chuẩn MPEG-4: Mpeg-4 là chuẩn cho các ứng dụng MultiMedia. Mpeg-4 trở thành một tiêu chuẩn cho nén ảnh kỹ thuật truyền hình số, các ứng dụng về đồ họa và Video tương tác hai chiều (Games, Videoconferencing) và các ứng dụng Multimedia tương tác hai chiều (World Wide Web hoặc các ứng dụng nhằm phân phát dữ liệu Video như truyền hình cáp, Internet Video...). Mpeg-4 đã trở thành một tiêu chuẩn công nghệ trong quá trình sản xuất, phân phối và truy cập vào các hệ thống Video. Nó đã góp phần giải quyết vấn đề về dung lượng cho các thiết bị lưu trữ, giải quyết vấn đề về băng thông của đường truyền tín hiệu Video hoặc kết hợp cả hai vấn đề trên.

1.2.3. Kết luận

Bộ mã hóa tự động (Autoencoders) có tiềm năng giải quyết nhu cầu ngày càng tăng về các thuật toán nén mát mót có thể thực hiện được. Tùy thuộc vào tình huống, các bộ mã hóa và giải mã có độ phức tạp tính toán khác nhau được yêu cầu. Khi gửi dữ liệu từ máy chủ đến thiết bị di động, có thể mong muốn ghép nối một bộ mã hóa mạnh mẽ với một bộ giải mã ít phức tạp hơn, nhưng các yêu cầu sẽ bị đảo ngược khi gửi dữ liệu theo hướng khác. Lượng sức mạnh tính toán và băng thông có sẵn cũng thay đổi theo thời gian khi có công nghệ mới. Đối với mục đích lưu trữ, thời gian mã hóa và giải mã ít quan trọng hơn so với các ứng dụng phát trực tuyến. Cuối cùng, các thuật toán nén hiện tại có thể không tối ưu cho các định dạng phương tiện mới như hình ảnh trường ánh sáng, video 360 hoặc nội dung VR

Vậy nên chúng em đã quyết định chọn bộ mã hóa tự động là đối tượng nghiên cứu chính cho chủ đề này

Chương 2

Cơ sở lý thuyết

2.1. Tổng quan về các kỹ thuật nén mất mát thông tin

Trong công nghệ thông tin, nén mất mát thông tin hoặc nén không thể đảo ngược là lớp phương pháp mã hóa dữ liệu sử dụng các phép gần đúng và loại bỏ một phần dữ liệu để thể hiện nội dung. Các kỹ thuật này được sử dụng để giảm kích thước dữ liệu để lưu trữ, xử lý và truyền tải nội dung. Công nghệ nén mất dữ liệu được thiết kế tốt thường làm giảm kích thước tệp đáng kể trước khi người dùng cuối nhận thấy sự xuống cấp.

Nén mất dữ liệu được sử dụng phổ biến nhất để nén dữ liệu đa phương tiện (âm thanh, video và hình ảnh), đặc biệt trong các ứng dụng như phương tiện truyền trực tiếp hoặc điện thoại internet.

Để mô tả định lượng mức độ gần đúng của dữ liệu tái tạo so với dữ liệu gốc, cần có một số hình thức đo độ biến dạng. Vậy nên, trước khi tìm hiểu về các thuật toán nén mất dữ liệu thường sử dụng, chúng ta cần quan tâm đến một số hình thức đo độ biến dạng.

Một phép đo sự biến dạng là 1 đại lượng toán học chỉ mức độ gần đúng so với giá trị ban đầu của nó sử dụng một số tiêu chí biến dạng. Khi nhìn vào dữ liệu đã nén, người ta thường nghĩ về sự khác biệt số giữa dữ liệu gốc và dữ liệu được tái tạo. Tuy nhiên, khi dữ liệu được nén là một ảnh, phép đo như vậy có thể không mang lại một kết quả như mong muốn.

Ví dụ: nếu hình ảnh được tái tạo giống với ảnh gốc ngoại trừ việc nó bị lệch sang phải bởi một đường quét dọc, thì một người quan sát bình thường sẽ gặp khó khăn trong việc phân biệt nó với ảnh gốc và do đó sẽ kết luận rằng sự biến dạng nhỏ. Tuy nhiên, khi tính toán được thực hiện bằng số, chúng ta nhận thấy sự biến dạng lớn, do những thay đổi lớn trong các pixel riêng lẻ của ảnh được tái tạo. Vấn đề là chúng ta cần một phép đo về sự biến đổi cảm giác, chứ không phải một phương pháp số học ngờ nghênh.

Trong số rất nhiều phép đo biến dạng cảm giác đã được tìm ra, chúng ta trình bày ba phép đo phổ biến nhất được sử dụng trong nén hình ảnh. Nếu chúng ta quan tâm đến sự khác nhau pixel trung bình, thì sai số bình phương(MSE- Mean square error) σ^2 thường được sử dụng:

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N (X_n - Y_n)^2 \quad (2.1)$$

Với X_n, Y_n , N lần lượt là chuỗi dữ liệu vào, chuỗi dữ liệu phục hồi, độ dài của chuỗi dữ liệu

2.2. Các kĩ thuật nén codec thường được sử dụng

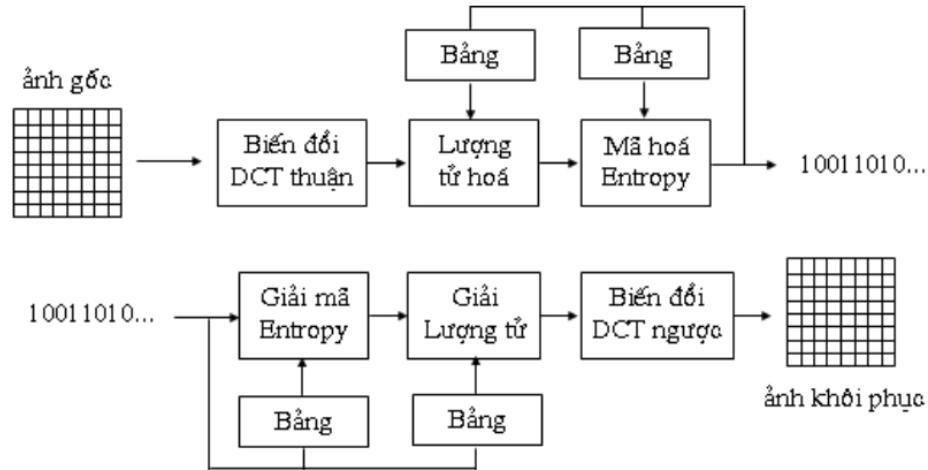
Mã hóa biến đổi(Transform coding), một số hình thức nén mất dữ liệu có thể được coi là ứng dụng của mã hóa biến đổi , là một kiểu nén dữ liệu được sử dụng cho hình ảnh kỹ thuật số tín hiệu , âm thanh kỹ thuật số và video kỹ thuật số .

Việc chuyển đổi thường được sử dụng để cho phép lượng tử hóa tốt hơn (nhắm mục tiêu hơn). Kiến thức về ứng dụng được sử dụng để chọn thông tin cần loại bỏ, do đó làm giảm băng thông của nó. Thông tin còn lại sau đó có thể được nén thông qua nhiều phương pháp. Khi đầu ra được giải mã, kết quả có thể không giống với đầu vào ban đầu, nhưng được mong đợi là đủ gần với mục đích của ứng dụng.

2.2.1. Phương pháp biến đổi Cosin rời rạc

Biến đổi Cosin rời rạc (Discrete Cosine Transform - DCT) là một kỹ thuật mã hóa biến đổi được sử dụng rộng rãi, có khả năng thực hiện mối tương quan của tín hiệu đầu vào theo cách độc lập với dữ liệu.

Nguyên tắc chính của phương pháp mã hóa này là biến đổi các giá trị pixel của ảnh trong miền không gian sang một tập các giá trị khác trong miền tần số, sao cho các hệ số trong tập giá trị mới này có tương quan giữa các điểm ảnh gần nhau nhỏ hơn.



Hình 2.1. Sơ đồ mã hóa, giải mã sử dụng DCT

Giải thích DCT

Cho một hàm $f(i,j)$ với 2 biến số nguyên i và j (là thành phần của ảnh), DCT 2 chiều biến đổi nó sang một hàm mới $F(u,v)$ với u, v chạy trên cùng một phạm vi với i và j .

Công thức chung cho phép biến đổi là:

$$\mathbf{F}_{(u,v)} = \frac{C(u)}{\sqrt{N/2}} \frac{C(v)}{\sqrt{N/2}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} \cos \frac{(2j+1)v\pi}{2N} \mathbf{f}_{(i,j)}$$

Trong đó $C(u)$ và $C(v)$ được xác định bởi:

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}} & u = 0 \\ 1 & u > 0 \end{cases}.$$

Biến đổi DCT nghịch:

$$\tilde{\mathbf{f}}_{(i,j)} = \frac{C(u)}{\sqrt{N/2}} \frac{C(v)}{\sqrt{N/2}} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} \cos \frac{(2j+1)v\pi}{2N} \mathbf{F}_{(u,v)}$$

Trong tiêu chuẩn nén ảnh JPEG, một khối ảnh được xác định có kích thước $N = 8$. Các phép biến đổi 2D có thể áp dụng cho các tín hiệu 2D, chẳng hạn như ảnh kỹ thuật số.

2.2.2. Biến đổi Karhunen-Loeve

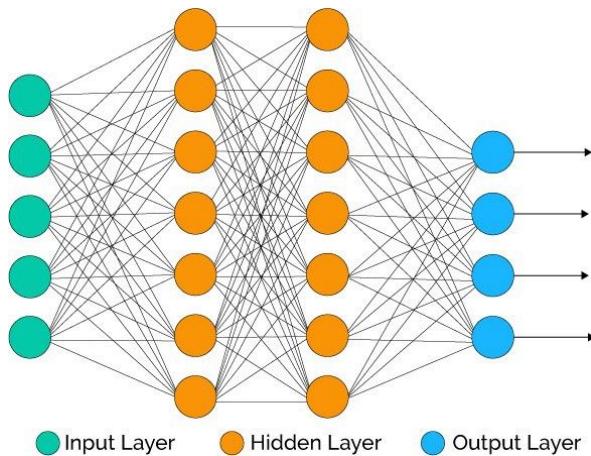
Phép biến đổi Karhunen-Loeve (KLT) là một phép biến đổi tuyến tính có thể đảo ngược khai thác các đặc tính thống kê của biểu diễn vectơ. Thuộc tính chính của nó là nó trang trí tối ưu đầu vào. Để làm như vậy, nó phù hợp với một ellipsoid n chiều xung quanh dữ liệu (trung bình đã trừ). Trục ellipsoid chính là hướng thay đổi chính của dữ liệu.

Hãy nghĩ về một điều xì gà không may bị giãm phái. Dữ liệu xì gà bao gồm một đám mây điểm trong không gian 3 cung cấp tọa độ vị trí của các điểm đo được trong điều xì gà. Trục dài của điều xì gà sẽ được xác định bởi một chương trình thống kê là trực KLT đầu tiên.

Trục quan trọng thứ hai là trực nằm ngang qua điều xì gà bếp, vuông góc với trực thứ nhất. Trục thứ ba là trực giao với cả hai và theo phương thẳng đứng, mỏng. Một chương trình thành phần KLT chỉ thực hiện phân tích này.

2.3. Mạng nơ-ron nhân tạo (Artificial Neural Network - ANN)

Artificial neural network (ANN) có thể coi là một tập hợp của các lớp. Các lớp ẩn được tạo ra từ perceptron đơn lẻ sẽ được kết hợp với nhau. Các lớp ẩn sử dụng các hàm kích hoạt phi tuyến ánh xạ các lớp đầu vào thành các lớp đầu ra trong một không gian có số chiều thấp hơn và được gọi chung là mạng nơ-ron nhân tạo. ANN có thể hiểu là một hàm biến đổi (ánh xạ) từ đầu vào đến đầu ra. Ánh xạ này được tính bằng cách thêm và các ma trận trọng số của các đầu vào với các độ lệch biases. Các giá trị của ma trận trọng số và giá trị của độ sai lệch biases tương ứng với kiến trúc được gọi là mô hình hay model.



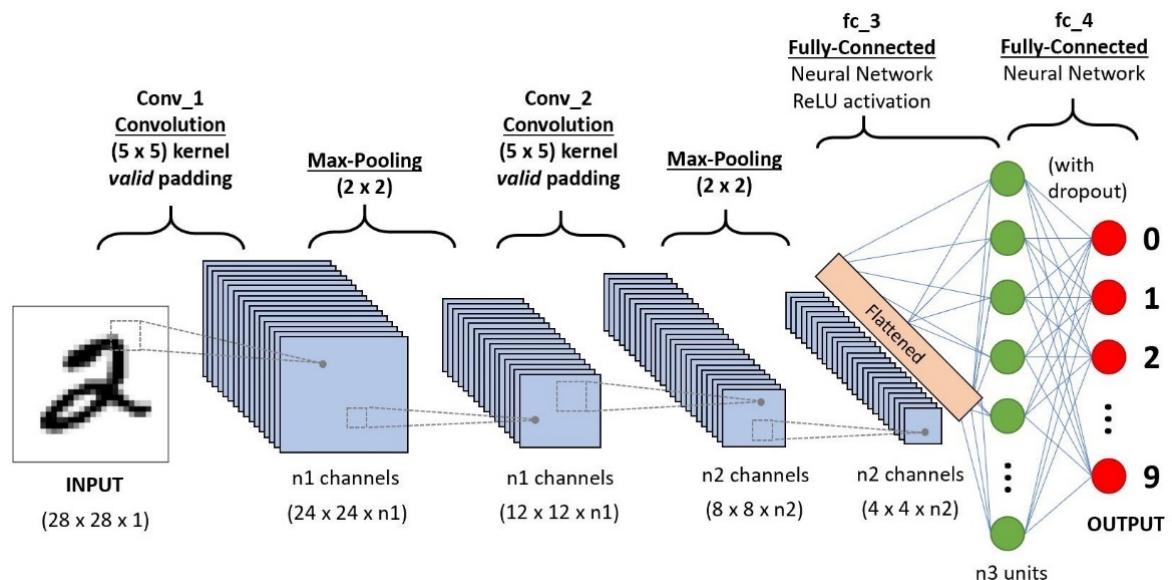
Hình 2.2. Ví dụ mạng nơ-ron nhân tạo đơn giản

Quá trình huấn luyện mô hình là quá trình xác định các giá trị của các trọng số và hệ số tự do. Các giá trị của chúng được khởi tạo với các giá trị ngẫu nhiên khi bắt đầu huấn luyện. Muốn huấn luyện chúng ta cần phải định nghĩa bằng hàm lỗi của mạng. Lúc này chúng ta sẽ nhắc đến hai giá trị đó là ground truth - tức là giá trị thực tế có trong tập dữ liệu huấn luyện và predict output là giá trị mà mô hình dự đoán. Lỗi được tính bằng cách sử dụng hàm loss định nghĩa dựa trên sự khác nhau giữa hai giá trị này. Có nhiều kiểu định nghĩa hàm loss khác nhau mà chúng ta sẽ thực hiện trong series này. Làm đến phần nào chúng ta sẽ nói rõ hơn ở phần đó. Dựa trên giá trị của hàm đánh giá măt mát được tính toán, các trọng số trọng số được điều chỉnh tại mỗi bước trong quá trình huấn luyện. Quá trình huấn luyện thực chất là tìm ra các đặc trưng features trong dữ liệu đầu vào. Các features này là một đại diện tốt hơn so với dữ liệu thô.

2.4. Mô hình mạng neural tích chập (Convolutional neural network - CNN)

Mạng neural tích chập (CNN) là một trong những mô hình học sâu[2] tiên tiến phổ biến nhất và có ảnh hưởng nhất với cộng đồng thị giác máy tính (Computer vision). CNN thường được dùng trong các bài toán nhận dạng ảnh, phân tích ảnh, xử lý ngôn ngữ tự nhiên dưới dạng ảnh các bước sóng. Và hầu hết đều cho hiệu quả tốt đến rất tốt.

CNN là một kiến trúc mạng neural sinh ra để xử lý các dữ liệu phi cấu trúc dạng ảnh. Có 2 loại lớp chính trong CNN: lớp tích chập (Convolutional layer) và lớp gộp (Pooling layer)



Hình 2.3. CNN cho bài toán nhận diện chữ số.

2.4.1. Lớp tích chập

Lớp tích chập là lớp quan trọng nhất và thường cũng là lớp đầu tiên của của mô hình CNN. Lớp này có chức năng chính là phát hiện các đặc trưng có tính không gian hiệu quả. Trong tầng này có 4 đối tượng chính là: ma trận đầu vào, bộ lọc (filters) và trường thu cảm, bản đồ đặc trưng (feature map). Lớp tích chập nhận đầu vào là một ma trận 3 chiều và một bộ lọc cần phải học. Bộ lọc này sẽ trượt qua từng vị trí trên

bức ảnh để tính tích chập (convolution) giữa bộ lọc và phần tương ứng trên bức ảnh. Phần tương ứng này trên bức ảnh gọi là trường thực cảm (receptive field), tức là vùng mà một neuron có thể nhìn thấy để đưa ra quyết định, và mà trận cho ra bởi quá trình này được gọi là bản đồ đặc trưng (feature map).

Để hình dung, có thể tưởng tượng, bộ filters giống như các tháp canh trong nhà tù quét lần lượt qua không gian xung quanh để tìm kiếm tên tù nhân bỏ trốn. Khi phát hiện tên tù nhân bỏ trốn, thì chuông báo động sẽ reo lên, giống như các bộ lọc tìm kiếm được đặc trưng nhất định thì tích chập đó sẽ cho giá trị tương ứng.

a. Lớp tích chập được coi như xác định đặc trưng

- Lớp tích chập có chức năng chính là phát hiện đặc trưng cụ thể của bức ảnh. Những đặc trưng này bao gồm đặc trưng cơ bản là góc, cạnh, màu sắc, hoặc đặc trưng phức tạp hơn như texture của ảnh. Vì bộ lọc quét qua toàn bộ bức ảnh, nên những đặc trưng này có thể nằm ở vị trí bất kỳ trong bức ảnh, cho dù ảnh bị xoay trái/phải thì những đặc trưng này vẫn bị phát hiện.
- Ở minh họa dưới, có một bộ lọc 5x5 dùng để phát hiện góc/cạnh, với bộ lọc này chỉ có giá trị mờ tại các điểm tương ứng một góc cong.

0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0

Hình 2.4. Bộ lọc phát hiện cạnh

- Dùng bộ lọc ở trên trước qua ảnh của nhân vật Olaf trong bộ phim Frozen. Chúng ta thấy rằng, chỉ ở những vị trí trên bức ảnh có dạng góc như đặc trưng ở bộ lọc thì mới có giá trị lớn trên bản đồ đặc trưng, những vị trí còn lại sẽ cho giá trị thấp hơn. Điều này có nghĩa là, bộ lọc đã phát hiện thành công một dạng

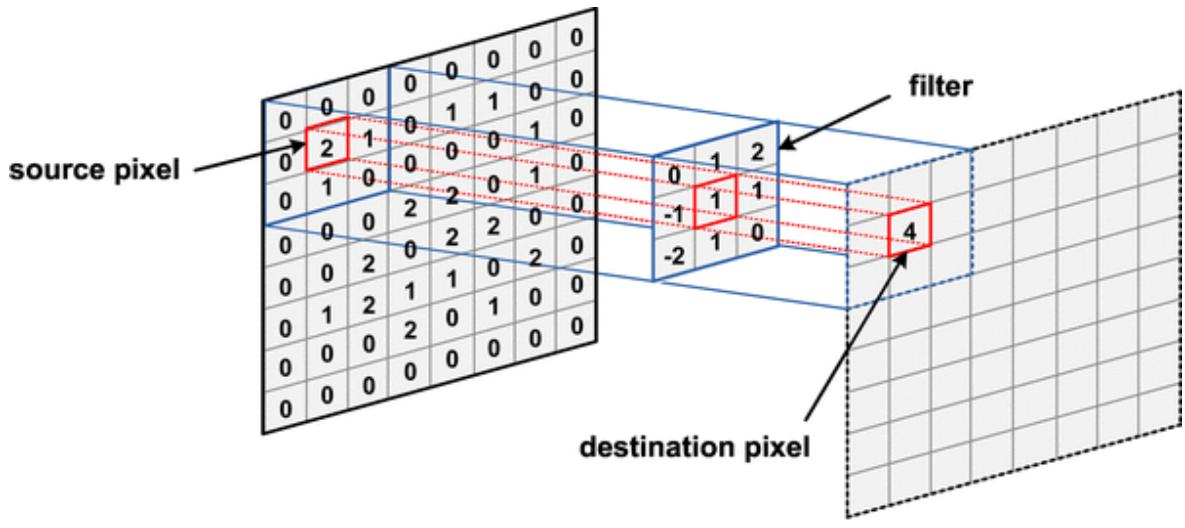
góc/cạnh trên dự liệu đầu vào. Tập hơn nhiều bộ lọc sẽ cho phép các bạn phát hiện được nhiều loại đặc trưng khác nhau, và giúp định danh được đối tượng.



Hình 2.5. Bộ lọc phát hiện cạnh

b. Các tham số: Kích thước bộ lọc, bước nhảy, lè

- Kích thước bộ lọc là một trong những tham số quan trọng nhất của lớp tích chập. Kích thước này tỉ lệ thuận với số tham số cần học tại mỗi lớp tích chập và là tham số quyết định trường thu cảm của tầng này. Kích thước phổ biến nhất của bộ lọc là 3x3.
- Kích thước bộ lọc nhỏ được ưu tiên lựa chọn thay kích thước lớn vì những lý do sau đây:
 - Cho phép nhìn được các vùng nhỏ
 - Trích rút được những đặc trưng có tính cục bộ cao
 - Phát hiện đặc trưng nhỏ



Hình 2.6. Cách hoạt động của bộ lọc (filter)

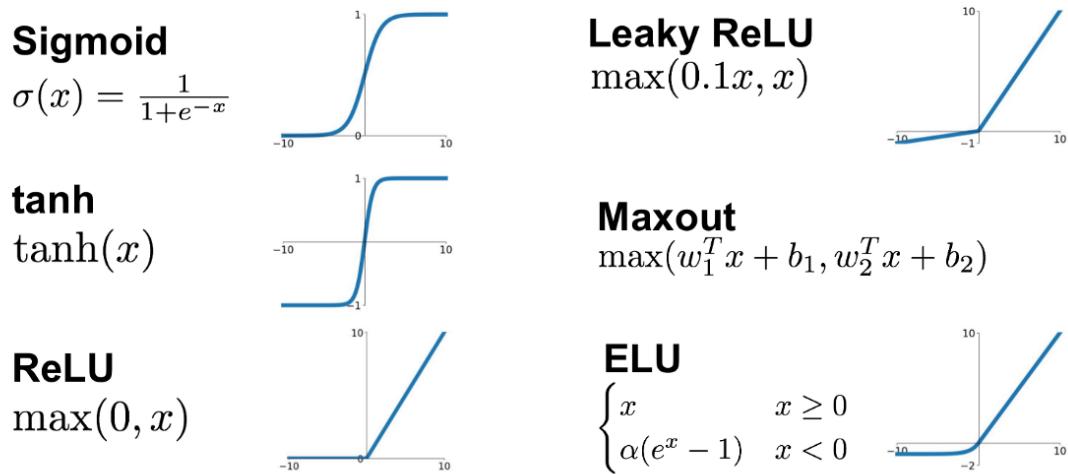
- Đặc trưng được trích rút sẽ nhiều, đa dạng
- Giảm kích thước ảnh chật, cho phép mạng sâu hơn
- Chia sẻ trọng số tốt hơn
- Kích thước của bộ lọc sẽ là những số lẻ để kết quả của phép tích chập sẽ nằm ở giữa ma trận

2.4.2. Lớp phi tuyến

Để các mô hình học sâu tìm được mối quan hệ phức tạp giữa các đặc trưng, cũng như tìm được những đặc trưng quan trọng (là sự kết hợp phi tuyến giữa các đặc trưng cơ bản khác) thì các mối quan hệ đó khó có thể được biểu diễn dưới các hàm tuyến tính mà cần sự kết hợp phi tuyến tính, chính vì vậy các hàm kích hoạt phi tuyến ra đời nhằm phá vỡ sự tuyến tính của giữa các đặc trưng từ đó tìm ra các đặc trưng mới quan trọng hơn.

Hiện nay hàm kích hoạt được sử dụng phổ biến nhất là hàm ReLU (Rectified Linear Units). Hàm ReLU được ưa chuộng vì tính đơn giản và cho kết quả tốt hơn ReLU cũng như những hàm kích hoạt khác, được đặt ngay sau tầng convolution, ReLU sẽ gán những giá trị âm bằng 0 và giữ nguyên giá trị của đầu vào khi lớn hơn 0.

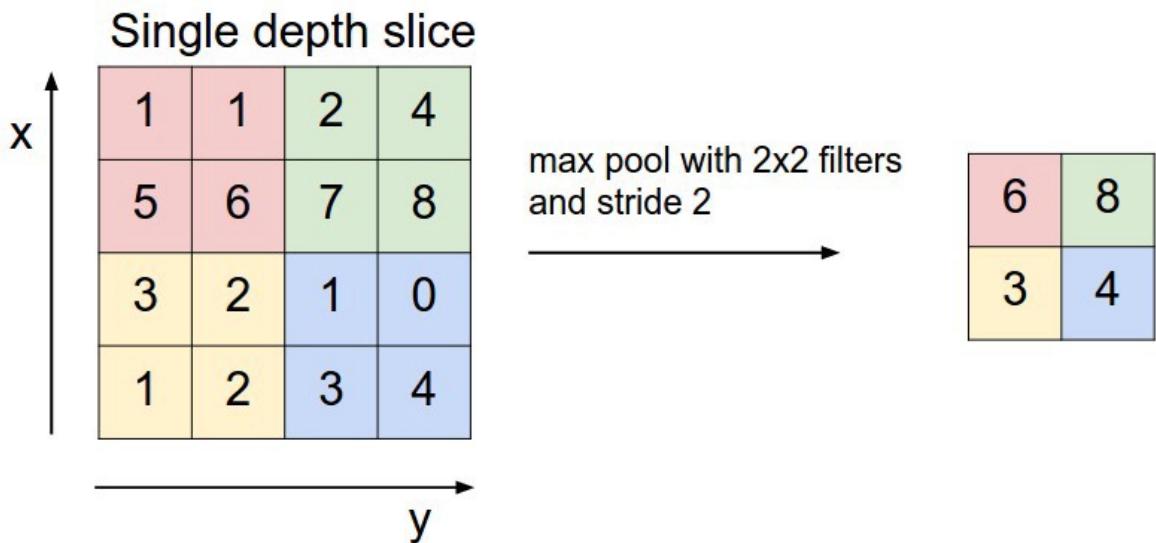
ReLU cũng có một số vấn đề tiềm ẩn như không có đạo hàm tại điểm 0, giá trị của hàm ReLU có thể lớn đến vô cùng và nếu chúng ta không khởi tạo trọng số cẩn thận, hoặc khởi tạo tỉ lệ học (learning rate) quá lớn thì những neuron ở tầng này sẽ rơi vào trạng thái chết, tức là luôn có giá trị < 0 .



Hình 2.7. Một số hàm kích hoạt thường được sử dụng

2.4.3. Lớp gộp

Sau hàm kích hoạt, thông thường chúng ta sử dụng lớp gộp. Một số loại lớp gộp phổ biến như là max-pooling, average pooling, với chức năng chính là giảm chiều của tầng trước đó. Với một lớp gộp có kích thước 2×2 , các bạn cần phải trượt bộ lọc 2×2 này trên những vùng ảnh có kích thước tương tự rồi sau đó tính giá trị lớn nhất, hay trung bình cho vùng ảnh đó.



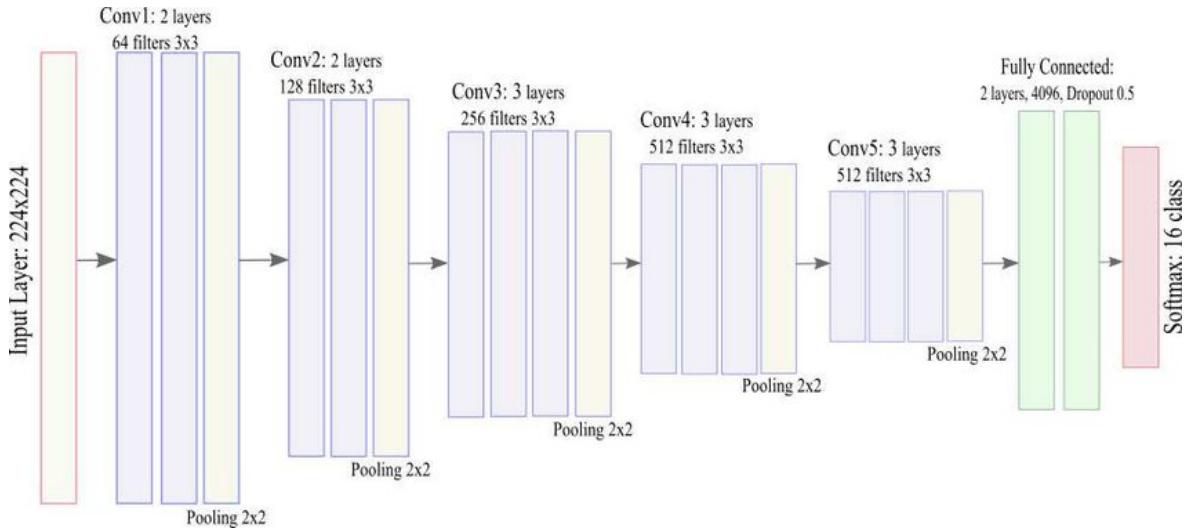
Hình 2.8. Ví dụ về max-pooling

Ý tưởng đằng sau lớp gộp là vị trí tuyết đối của những đặc trưng trong không gian ảnh không còn cần thiết, thay vào đó vị trí tương đối giữ các đặc trưng đã đủ để phân loại đối tượng. Hơn nữa, giảm tầng pooling có khả năng giảm chiều cực kì nhiều, làm hạn chế overfit, và giảm thời gian huấn luyện tốt.

2.4.4. Lớp kết nối đầy đủ

Lớp cuối cùng của mô hình CNN trong bài toán phân loại ảnh là lớp kết nối đầy đủ. Lớp này có chức năng chuyển ma trận đặc trưng ở tầng trước thành các vector chứa xác suất của các đối tượng cần được dự đoán.

Quá trình huấn luyện mô hình CNN cho bài toán phân loại ảnh cũng tương tự như huấn luyện các mô hình khác. Cần có hàm đánh giá măt măt để tính sai số giữa dự đoán của mô hình và nhãn chính xác, để sử dụng cơ chế của thuật toán lan truyền ngược (backpropagation) cho quá trình cập nhật trọng số.



Hình 2.9. Một CNN đơn giản với đầy đủ các lớp

2.5. Bộ mã hóa tự động (Autoencoder)

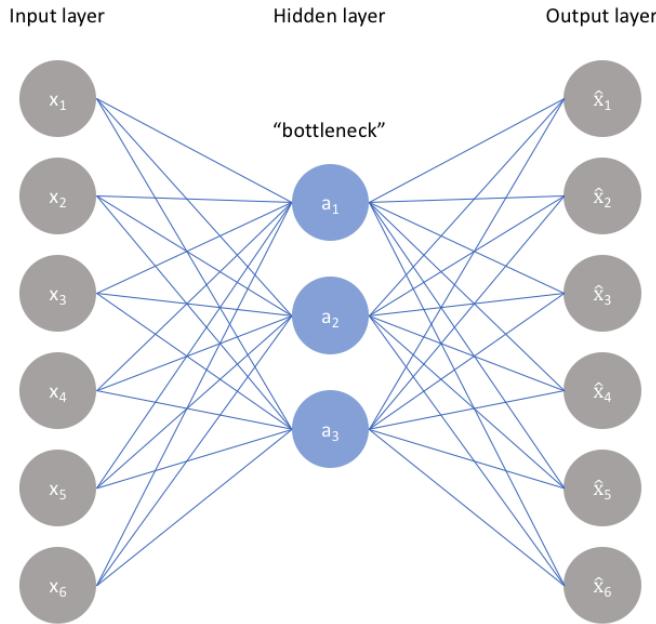
Bộ mã hóa tự động là một kỹ thuật học tập không có giám sát, trong đó chúng em tận dụng mạng nơ-ron cho nhiệm vụ của học để biểu diễn các tập giá trị dưới dạng nén và học cách để giải mã dữ liệu từ dạng nén.

Cụ thể, chúng em sẽ thiết kế một kiến trúc mạng nơ-ron nhân tạo sau đó áp đặt một nút thắt cổ chai trong mạng - điều này đại diện cho sự nén lại một cách tự động. Mạng này sẽ phải biểu diễn tri thức đầu vào dưới dạng các biểu diễn trong ít chiều không gian hơn, đây chính là biểu diễn nén của đầu vào.

Nếu các tính năng đầu vào từng độc lập của nhau, việc nén này và tái tạo sau đó sẽ là một nhiệm vụ rất khó khăn. Tuy nhiên, nếu một số loại cấu trúc tồn tại trong dữ liệu (ví dụ như: mối tương quan giữa các tính năng đầu vào), cấu trúc này có thể được học và do đó được tận dụng khi buộc đầu vào thông qua nút thắt cổ chai của mạng.

Bộ mã hóa tự động lý tưởng cân bằng những điều sau đây:

- Nhạy cảm với các yếu tố đầu vào đủ để xây dựng lại một cách chính xác.
- Đủ nhạy cảm với các đầu vào mà mô hình không chỉ đơn giản là ghi nhớ hoặc trang bị quá nhiều dữ liệu đào tạo.



Hình 2.10. Bộ mã hóa tự động.

Sự đánh đổi này buộc mô hình chỉ duy trì các biến thể trong dữ liệu cần thiết để cấu trúc lại đầu vào mà không giữ lại các phần dư thừa trong đầu vào. Đối với hầu hết các trường hợp, điều này liên quan đến việc xây dựng một hàm mất mát trong đó phải thỏa mãn mô hình của chúng ta nhạy cảm với các yếu tố đầu vào (ví dụ: xây dựng lại 1 hàm mất mát $\mathcal{L}(x, \hat{x})$ và thêm một chính quy hóa)

$$\mathcal{L}(x, \hat{x}) + \text{regularizer} \quad (2.2)$$

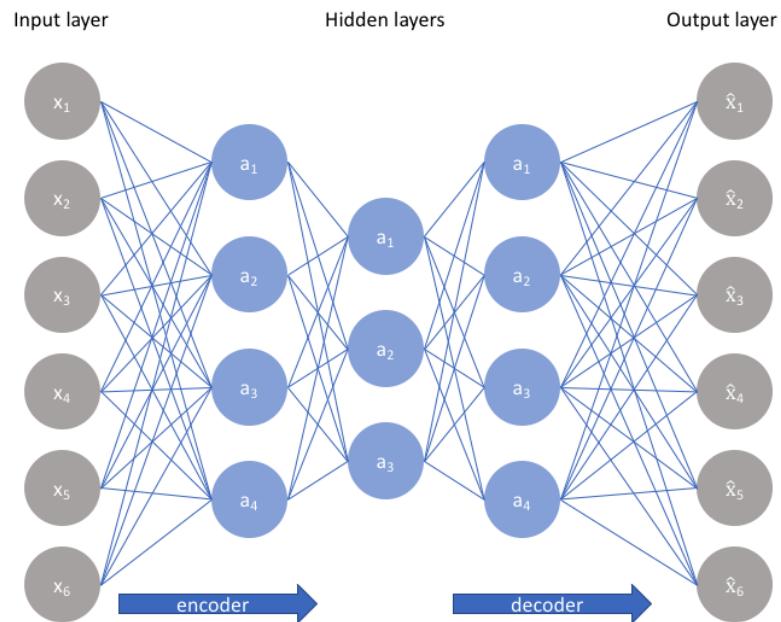
Thông thường, sẽ có thêm một tham số tỷ lệ trước thuật ngữ chính quy để chúng ta có thể điều chỉnh sự cân bằng giữa hai mục tiêu.

Dưới đây chúng em sẽ trình bày về một số kiến trúc của bộ mã hóa tự động tiêu chuẩn để áp đặt 2 ràng buộc này và điều chỉnh sự cân bằng.

2.5.1. Bộ mã hóa tự động chưa hoàn chỉnh

Kiến trúc đơn giản nhất để xây dựng bộ mã hóa tự động là hạn chế số lượng nút hiện diện trong (các) lớp ẩn của mạng, hạn chế lượng thông tin có thể truyền qua mạng. Bằng cách sử dụng các hình phạt mạng theo lỗi xây dựng lại, mô hình của chúng tôi có thể tìm hiểu các thuộc tính quan trọng nhất của dữ liệu đầu vào và cách tái tạo tốt

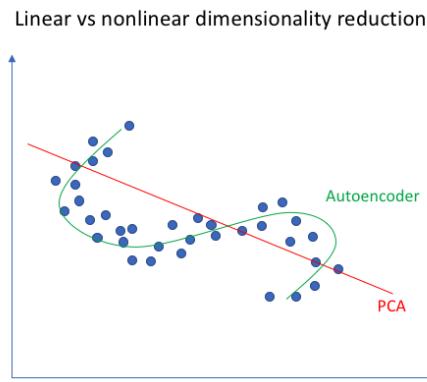
nhất dữ liệu đầu vào ban đầu từ trạng thái "được mã hóa". Lý tưởng nhất là bảng mã này sẽ tìm hiểu và mô tả các thuộc tính tiềm ẩn của dữ liệu đầu vào.



Hình 2.11. Mô tả mô hình bộ mã hóa tự động chưa hoàn chỉnh.

Bởi vì mạng nơ-ron có khả năng học các mối quan hệ phi tuyến, điều này có thể được coi là một sự tổng quát hóa (phi tuyến) mạnh mẽ hơn của PCA (kỹ thuật giảm chiều dữ liệu tuyến tính)

Trong khi PCA cố gắng khám phá một siêu phẳng có chiều thấp hơn mô tả dữ liệu ban đầu, thì các bộ mã hóa tự động có khả năng học các đa tạp phi tuyến (đa tạp được định nghĩa theo thuật ngữ đơn giản là liên tục, không giao nhau bề mặt). Sự khác biệt giữa hai cách tiếp cận này được hình dung bên dưới.



Hình 2.12. Mô tả mô hình bộ mã hóa tự động chưa hoàn chỉnh.

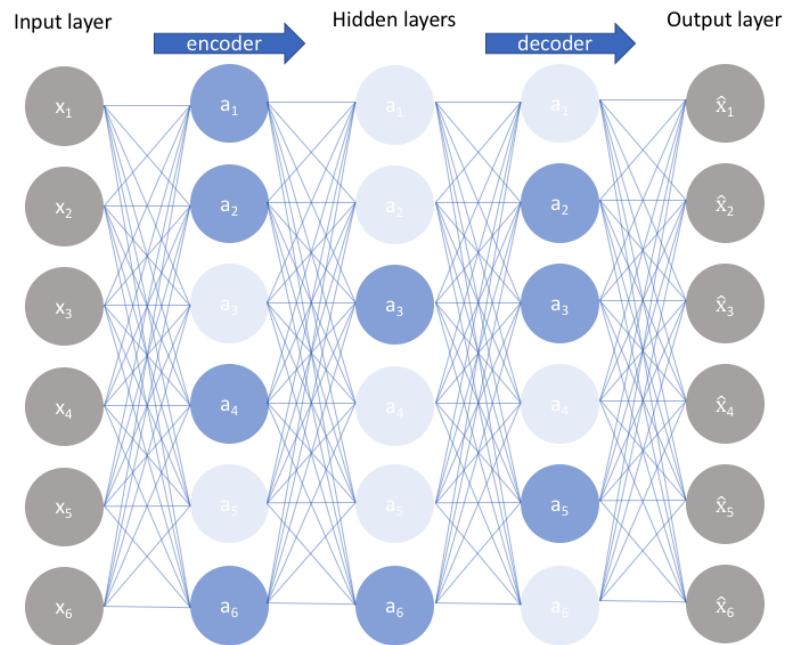
Một bộ mã hóa tự động chưa hoàn chỉnh không có thuật ngữ chính quy rõ ràng - chúng tôi chỉ đào tạo mô hình của mình theo sự mất mát khi xây dựng lại. Do đó, cách duy nhất của chúng tôi để đảm bảo rằng mô hình không ghi nhớ dữ liệu đầu vào là đảm bảo rằng chúng tôi đã hạn chế đủ số lượng các nút trong (các) lớp ẩn.

Đối với các bộ mã hóa tự động sâu, chúng ta cũng phải lưu ý về dung lượng của bộ mã hóa và các kiểu máy giải mã của chúng ta. Ngay cả khi "lớp nút cổ chai" chỉ là một nút ẩn, mô hình của chúng tôi vẫn có thể ghi nhớ dữ liệu huấn luyện với điều kiện là các mô hình bộ mã hóa và giải mã có đủ khả năng để học một số chức năng tùy ý có thể ánh xạ dữ liệu thành một chỉ mục.

2.5.2. Bộ mã hóa tự động thưa thớt

Các bộ mã hóa tự động thưa thớt cung cấp cho chúng ta một phương pháp thay thế để giới thiệu một nút thắt cổ chai thông tin mà không yêu cầu giảm số lượng nút ở các lớp ẩn của chúng ta. Thay vào đó, chúng tôi sẽ xây dựng hàm mất mát của chúng tôi để chúng tôi xử phạt các hàm kích hoạt trong một lớp. Đối với bất kỳ quan sát

nhất định nào, chúng ta sẽ khuyến khích mạng của mình học cách mã hóa và giải mã chỉ dựa vào việc kích hoạt một số lượng nhỏ neuron.



Hình 2.13. Mô tả mô hình bộ mã hóa tự động chưa hoàn chỉnh.

Một bộ mã hóa tự động thừa thớt chung được hiển thị bên trên nơi độ mờ của một nút tương ứng với mức độ kích hoạt. Điều quan trọng cần lưu ý là các nút riêng lẻ của một mô hình được đào tạo kích hoạt phụ thuộc vào dữ liệu, các đầu vào khác nhau sẽ dẫn đến việc kích hoạt các nút khác nhau thông qua mạng.

Một kết quả của thực tế này là chúng tôi cho phép mạng của mình nhạy cảm với các nút lớp ẩn riêng lẻ đối với các thuộc tính cụ thể của dữ liệu đầu vào. Trong khi một bộ mã hóa tự động chưa hoàn chỉnh sẽ sử dụng toàn bộ mạng cho mỗi lần quan sát, một bộ mã hóa tự động thừa thớt sẽ buộc phải kích hoạt có chọn lọc các vùng của mạng tùy thuộc vào dữ liệu đầu vào. Do đó, chúng tôi đã giới hạn khả năng ghi nhớ dữ liệu đầu vào của mạng mà không giới hạn khả năng mạng trích xuất các tính năng từ dữ liệu.

Điều này cho phép chúng tôi xem xét biểu diễn trạng thái tiềm ẩn và quy định của mạng một cách riêng biệt, do đó chúng tôi có thể chọn biểu diễn trạng thái tiềm ẩn (tức là kích thước mã hóa) phù hợp với những gì có ý nghĩa với ngữ cảnh của dữ liệu trong khi áp đặt chính quy bởi ràng buộc thừa thớt.

Có hai cách chính mà chúng ta có thể áp đặt hạn chế thừa thớt này; cả hai đều liên quan đến việc đo lường các kích hoạt lớp ẩn cho mỗi lô đào tạo và thêm một số thuật ngữ vào hàm mất mát để xử phạt các kích hoạt quá mức. Các cách này là:

- **L1 Regularization:** Chúng ta có thể thêm một thuật ngữ vào hàm mất mát để phạt giá trị tuyệt đối của vector kích hoạt a trong lớp h với quan sát i , được chia tỉ lệ bằng 1 tham số điều chỉnh λ

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i |a_i^{(h)}| \quad (2.3)$$

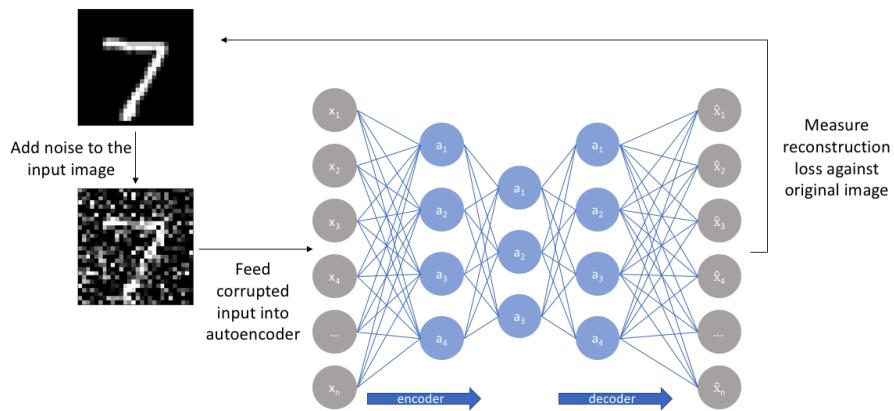
- **KL-phân kỳ** Về bản chất, KL-phân kỳ là thước đo sự khác biệt giữa hai phân phối xác suất. Chúng ta có thể xác định tham số thừa thớt p biểu thị kích hoạt trung bình của một nơ-ron trên một tập hợp các mẫu. Kỳ vọng này có thể được tính là:

$$\hat{\rho}_j = \frac{1}{m} \sum_i [a_i^{(h)}(x)] \quad (2.4)$$

trong đó chỉ số con i biểu thị nơ-ron cụ thể trong lớp h , tính tổng các kích hoạt cho các quan sát huấn luyện m được ký hiệu riêng lẻ là x .

2.5.3. Bộ mã hóa tự động giảm nhiễu

Như ở phần giới thiệu, bộ mã hóa tự động chính là một mạng nơ-ron được đào tạo trong đó đầu vào giống hệt đầu ra và mô hình có nghiệm vụ tái tạo đầu vào càng chặt chẽ càng tốt khi chuyển qua thông tin ở lớp thắt cổ chai. Một cách tiếp cận khác hướng tới việc phát triển một mô hình tổng quát hóa là làm nhiễu một chút dữ liệu đầu vào nhưng vẫn duy trì dữ liệu không bị gián đoạn làm đầu ra mục tiêu của mô hình.



Hình 2.14. Mô tả mô hình bộ mã hóa tự động chưa hoàn chỉnh.

Với cách tiếp cận này, mô hình không thể đơn giản phát triển một ánh xạ ghi nhớ dữ liệu đào tạo vì đầu vào và đầu ra mục tiêu không còn giống nhau. Thay vào đó, mô hình học một trường vectơ để ánh xạ dữ liệu đầu vào tới một đa tạp có chiều thấp hơn (nhớ lại từ hình ảnh trước đây của tôi rằng một đa tạp mô tả vùng mật độ cao nơi dữ liệu đầu vào tập trung).

2.6. Ngôn ngữ lập trình Python và thư viện PyTorch

2.6.1. Ngôn ngữ lập trình Python

Python là ngôn ngữ lập trình có mục đích chung được bắt đầu bởi Guido van Rossum, nó trở nên rất phổ biến rất nhanh trong thời gian gần đây, chủ yếu vì tính đơn giản và khả năng đọc mã của nó. Nó cho phép lập trình viên thể hiện ý tưởng trong ít dòng mã hơn mà không làm giảm khả năng đọc.

So với các ngôn ngữ như C/C++, Python chậm hơn. Điều đó nói rằng, Python có thể dễ dàng được mở rộng với C/C++, cho phép chúng ta viết mã chuyên sâu tính

toán trong C/C++ và tạo các trình bao bọc Python có thể được sử dụng làm mô-đun Python. Điều này mang lại cho chúng ta hai lợi thế: thứ nhất, mã nhanh như mã C/C++ gốc (vì đây là mã C++ thực tế hoạt động ở chế độ nền) và thứ hai, mã dễ dàng hơn trong Python so với C/C++. OpenCV - Python là một trình bao bọc Python để thực hiện OpenCV C++ ban đầu.

2.6.2. Thư viện PyTorch

PyTorch là một thư viện hỗ trợ tạo ra các mô hình mạng nơ-ron nhân tạo và sử dụng chúng trong các ứng dụng khác nhau. Trên thực tế PyTorch chính là một gói hỗ trợ tính toán khoa học (scientific computing) như tài liệu chính thức của PyTorch đã đề cập

PyTorch, tương tự như Python, nó được thiết kế tập trung vào tính dễ sử dụng và thậm chí người dùng có kiến thức lập trình rất cơ bản cũng có thể sử dụng nó trong các dự án có liên quan đến học sâu.

Chương 3

Triển khai chương trình và đánh giá hiệu suất

3.1. Triển khai chương trình

Về cơ bản để triển khai một bộ mã hóa tự động cho bài toán nén có các bước chính sau:

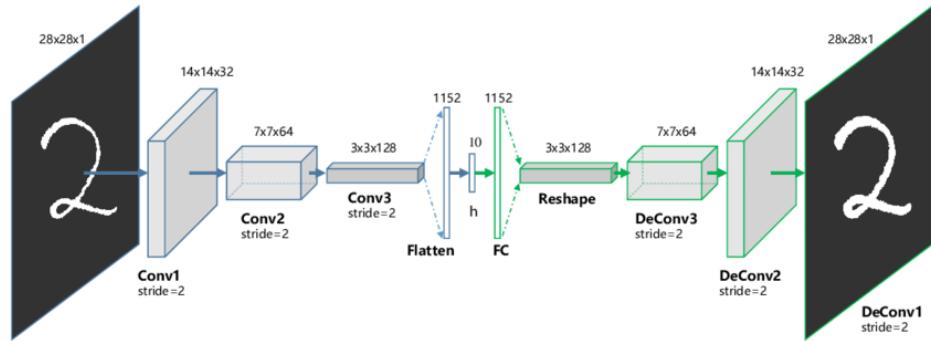
- Thiết kế các lớp mạng
- Lựa chọn hàm đánh giá măt măt
- Lựa chọn thuật toán tối ưu
- Huấn luyện mô hình với tập dữ liệu cần nén
- Thực hiện nén, giải nén

Hoạt động nén được chia thành 2 quy trình:

- Huấn luyện mô hình (đại diện thao tác nén)
- Thực hiện nén, giải nén tập dữ liệu đã huấn luyện

3.1.1. Thiết kế các lớp mạng

Kết cấu của mạng này được thiết kế với 50 lớp chia thành 2 phần, với các lớp tính chập đan xen lớp kích hoạt



Hình 3.1. Mô tả mô hình bộ mã hóa tự động đơn giản.

Thông kê số lượng tham số trong mạng, tương đương với số lượng các phép tính thực hiện khi mỗi ảnh được truyền qua mạng.

3.1.2. Lựa chọn thuật toán tối ưu

Các thuật toán tối ưu trong học máy đã phát triển mạnh mẽ từ lâu và đã đạt đến mức hoàn thiện. Hiện nay có một số thuật toán tối ưu nổi bật như sau :

- Adam [4] là một trong những trình tối ưu hóa phổ biến nhất còn được gọi là Ước tính thời điểm thích ứng, nó kết hợp các thuộc tính tốt của Adadelta và trình tối ưu hóa RMSprop thành một và do đó có xu hướng làm tốt hơn cho hầu hết các vấn đề.
- Adagrad [1] (viết tắt của adaptive gradient) xử phạt tốc độ học tập đối với các tham số được cập nhật thường xuyên, thay vào đó, nó cung cấp tốc độ học tập nhiều hơn cho các tham số thừa thớt, các tham số không được cập nhật thường xuyên.
- Stochastic gradient descent [3] là cực kỳ cơ bản và hiếm khi được sử dụng bây giờ. Một vấn đề là với tỷ lệ học tập trên toàn thế giới liên quan đến một tương đương. Do đó, nó không hoạt động tốt khi các thông số ở nhiều thang đo vì tốc độ học cà phê sẽ làm cho quá trình đào tạo chậm lại trong khi tốc độ học

tập quá lớn có thể gây ra dao động. Ngoài ra, dốc Stochastic đi xuống thường gặp khó khăn khi thoát khỏi các điểm yên ngựa. Adagrad, Adadelta, RMSprop và ADAM thường xử lý các điểm yên ngựa tốt hơn. SGD với xung lượng đưa ra một số tốc độ tối ưu hóa và cũng giúp thoát cực tiểu cục bộ tốt hơn.

Nhưng sau khi thực hiện những thử nghiệm nhỏ đơn giản, thì chúng em quyết định lựa chọn thuật toán tối ưu Adam, là thuật toán đạt độ mượt mà thấp và tốc độ hội tụ để hướng đến nghiệm tối ưu của bài toán tốt.

3.1.3. Lựa chọn hàm đánh giá măt măt

Cũng giống như các thuật toán tối ưu, các hàm đánh giá măt măt cũng đã được phát triển và hoàn thiện từ lâu, có một số hàm đánh giá nổi bật như sau:

- Cross Entropy: Suy hao chéo entropy hay còn gọi là măt log, đo lường hiệu suất của mô hình phân loại có đầu ra là giá trị xác suất từ 0 đến 1. Măt entropy chéo tăng lên khi xác suất dự đoán khác với nhãn thực tế. Vì vậy, dự đoán xác suất bằng 0,12 khi nhãn quan sát thực tế là 1 sẽ không tốt và dẫn đến giá trị tổn thất cao. Một mô hình hoàn hảo sẽ có lỗ nhặt ký bằng 0.
- Hinge - Thường dùng trong các bài toán phân loại
- Huber - Thường được sử dụng để hồi quy. Nó ít nhạy cảm hơn với các ngoại lệ so với MSE vì nó coi lỗi là hình vuông chỉ trong một khoảng thời gian.
- MAE, MSE - Là phép tính chuẩn 1, chuẩn 2 thường xuất hiện rất nhiều trong các bài toán học máy vì tính đơn giản mà hiệu quả mà nó mang lại.

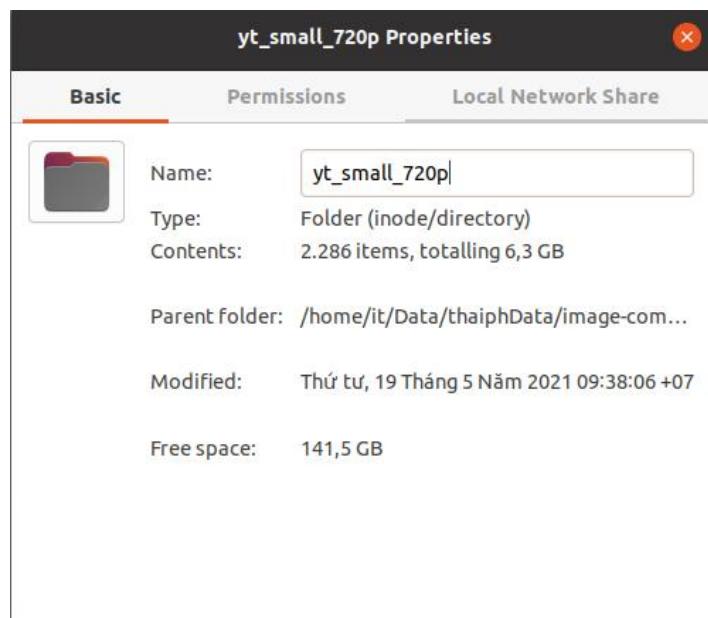
Vì đây là bài toán so sánh độ tương đồng giữa ảnh nên chúng em lựa chọn MSE làm phương pháp đánh giá măt măt cho mô hình

3.1.4. Huấn luyện mô hình với tập dữ liệu cần nén

Tập dữ liệu sử dụng

Ở đây, để tăng tính ngẫu nhiên chúng em sử dụng 1 tập dữ liệu ảnh màu với các kích thước giống nhau (768x1280) được cung cấp từ một thư viện lấy các ảnh ngẫu

nhiên từ các video trên nền tảng Youtube. Với số lượng là 2286 ảnh chất lượng cao, tổng dung lượng lưu trữ là 6.3 GB



Hình 3.2. Tập dữ liệu sử dụng

Các tham số huấn luyện mô hình

Khi huấn luyện một mô hình mạng nơ-ron nhận tạo điển hình có các tham số như là : số kỷ nguyên (epoch), Kích thước lô (batch size), số lân lặp lại (Iterations)

Đối với mô hình này chúng ta phải lựa chọn thêm 1 số tham số giống như số luồng (thread), tỷ lệ học (learning rate) để mô hình trở lên linh hoạt hơn

Với GPU Tesla T4 16GB, được cung cấp bởi Google Colab và dữ liệu 2286 ảnh chúng em lựa chọn các tham số kích thước lô là 16, số kỷ nguyên là 3, tỉ lệ học tập là 0,0001, số luồng chạy là 5

```
configs > {} train.yaml
1 | num_epochs: 3
2 | batch_size: 16
3 | learning_rate: 0.0001
4 | resume: false
5 | checkpoint: null
6 | start_epoch: 1
7 | exp_name: training
8 | batch_every: 1
9 | save_every: 10
10 | epoch_every: 1
11 | shuffle: true
12 | dataset_path: datasets/training/yt_small_720p
13 | num_workers: 5
14 | device: cuda
```

Hình 3.3. Các tham số được lựa chọn

Quá trình huấn luyện mô hình

```
=====
Total params: 2,241,859
Trainable params: 2,241,859
Non-trainable params: 0
-----
Input size (MB): 0.19
Forward/backward pass size (MB): 133.33
Params size (MB): 8.55
Estimated Total Size (MB): 142.07
-----
```

Hình 3.4. Thông tin tổng quan của mạng

Mạng nơ-ron có tổng số tham số khoảng 2 triệu tham số với 50 lớp mạng chủ yếu là các lớp tích chập, phi tuyến.

Tổng thời gian khi huấn luyện mô hình là khoảng 30 phút, điều này đại diện cho thời gian nén mặc dù vẫn chưa thực sự tối ưu về mặt thời gian, nhưng chúng em tin rằng khi phần cứng mạnh mẽ hơn thì thời gian nén cũng sẽ giảm đi đáng kể.

```
[ ] !python3 src/train.py --config configs/train.yaml
[283][21-05-21 03:50:41.975 @ train] INFO: training: experiment training
2021-05-21 03:50:43.042837: I tensorflow/stream_executor/platform/default/dso_loader.cc:49] Succ
[283][21-05-21 03:50:44.398 @ train] INFO: started tensorboard writer
[283][21-05-21 03:50:47.670 @ train] INFO: loaded model on cuda
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserWarning: This Dat
cpuset checked)
[283][21-05-21 03:50:55.509 @ train] INFO: loaded dataset from /content/gdrive/MyDrive/image-com
epoch_idx : 1/4
1->[283][21-05-21 03:51:15.689 @ train] DEBUG: [ 1/ 3][ 1/ 143] avg_loss: 0.10297835
2->[283][21-05-21 03:51:25.358 @ train] DEBUG: [ 1/ 3][ 2/ 143] avg_loss: 0.01997992
3->[283][21-05-21 03:51:34.921 @ train] DEBUG: [ 1/ 3][ 3/ 143] avg_loss: 0.01628421
4->[283][21-05-21 03:51:44.499 @ train] DEBUG: [ 1/ 3][ 4/ 143] avg_loss: 0.00896005
5->[283][21-05-21 03:51:54.142 @ train] DEBUG: [ 1/ 3][ 5/ 143] avg_loss: 0.01329732
6->[283][21-05-21 03:52:03.768 @ train] DEBUG: [ 1/ 3][ 6/ 143] avg_loss: 0.00931335
7->[283][21-05-21 03:52:13.526 @ train] DEBUG: [ 1/ 3][ 7/ 143] avg_loss: 0.00742883
```

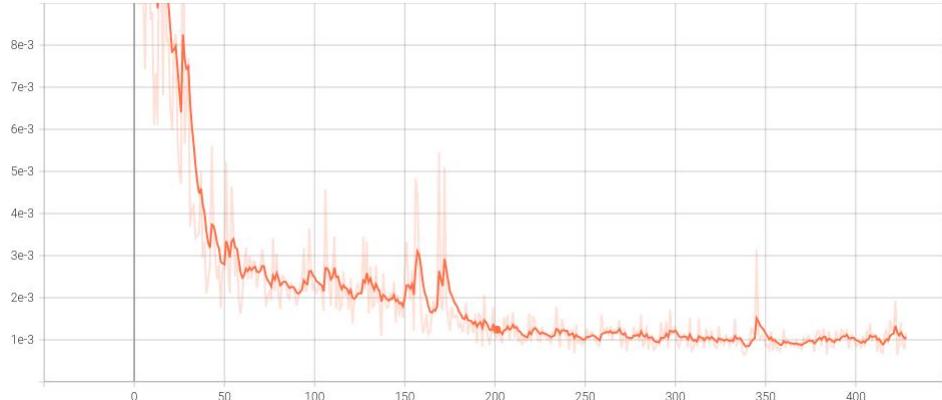
Hình 3.5. Khởi đầu quá trình huấn luyện

Có thể thấy rằng tham số avg-loss (đánh giá sứ mệt mát) từ lúc bắt đầu huấn luyện cho đến khi kết thúc đã giảm đi rất đáng kể cho thấy tính đúng đắn của mô hình, dự báo rằng mô hình sẽ đạt kết quả thử nghiệm tốt.

```
139->[283][21-05-21 05:06:01.042 @ train] DEBUG: [ 3/ 3][ 139/ 143] avg_loss: 0.00086924
140->[283][21-05-21 05:06:11.469 @ train] DEBUG: [ 3/ 3][ 140/ 143] avg_loss: 0.00140475
141->[283][21-05-21 05:06:22.938 @ train] DEBUG: [ 3/ 3][ 141/ 143] avg_loss: 0.00086202
142->[283][21-05-21 05:06:33.376 @ train] DEBUG: [ 3/ 3][ 142/ 143] avg_loss: 0.00079229
143->[283][21-05-21 05:06:42.146 @ train] DEBUG: [ 3/ 3][ 143/ 143] avg_loss: 0.00116826
[283][21-05-21 05:06:42.312 @ train] INFO: Epoch avg = 0.00102917
```

Hình 3.6. Kết thúc quá trình huấn luyện

Mô hình hóa quá trình huấn luyện sử dụng tensorboard : (Trục ngang là số lần lặp lại, trục đứng là đánh giá mệt mát)



Hình 3.7. Quá trình giảm của mệt mát theo số lần lặp lại các lô

3.1.5. Thực hiện nén, giải nén

- Đầu vào của chương trình nén là ảnh với các thông số 768x1280x3 (lần lượt là chiều rộng, chiều cao, số kênh màu)
- Sau khi ảnh được đưa vào bộ nén sẽ được nén thành dạng 6x10x32x32x32

- Khi đưa file nén vào bộ giải nén ảnh sẽ được khôi phục lại kích thước 768x1280x3

Nén

Vì mạng nơ-ron chỉ nhận những đầu vào có kích thước nhỏ nên ảnh đầu vào sẽ được chia làm 60 phần bằng nhau tức kích thước ban đầu là 768x1280x3 sẽ được chia thành 60x128x128x3

Và 60 hình nhỏ này sẽ lần lượt chạy qua mạng và sẽ được lặp đi lặp lại một số lần trong quá trình học của mạng để tạo ra các khối được nén lại. Các khối này có kích thước 32*32*32.

ZeroPad2d-1	[-1, 3, 131, 131]	0
Conv2d-2	[-1, 64, 64, 64]	4,864
LeakyReLU-3	[-1, 64, 64, 64]	0
ZeroPad2d-4	[-1, 64, 67, 67]	0
Conv2d-5	[-1, 128, 32, 32]	204,928
LeakyReLU-6	[-1, 128, 32, 32]	0
ZeroPad2d-7	[-1, 128, 34, 34]	0
Conv2d-8	[-1, 128, 32, 32]	147,584
LeakyReLU-9	[-1, 128, 32, 32]	0
ZeroPad2d-10	[-1, 128, 34, 34]	0
Conv2d-11	[-1, 128, 32, 32]	147,584
ZeroPad2d-12	[-1, 128, 34, 34]	0
Conv2d-13	[-1, 128, 32, 32]	147,584
LeakyReLU-14	[-1, 128, 32, 32]	0
ZeroPad2d-15	[-1, 128, 34, 34]	0
Conv2d-16	[-1, 128, 32, 32]	147,584
ZeroPad2d-17	[-1, 128, 34, 34]	0
Conv2d-18	[-1, 128, 32, 32]	147,584
LeakyReLU-19	[-1, 128, 32, 32]	0
ZeroPad2d-20	[-1, 128, 34, 34]	0
Conv2d-21	[-1, 128, 32, 32]	147,584
Conv2d-22	[-1, 32, 32, 32]	102,432
Tanh-23	[-1, 32, 32, 32]	0

Hình 3.8. Các lớp mạng tham gia vào quá trình nén

Giải nén

Khi các khối nén 32x32x32 đã được học xong thì chúng ta lưu nó lại như là thành phần chính của file nén.

Một thành phần nữa không thể thiếu đó chính là các bộ trọng số đại diện cho tri thức của mạng sau khi đã học nén và giải nén.

Khi thực hiện giải nén, chúng ta lại đưa file nén gồm nhiều các khối 32x32x32 vào mô hình giảm mã lần lượt để mô hình có thể giải mã giống như những gì nó đã học

được trong quá trình huấn luyện để được các đầu ra mong muốn là các khối 128x128x3 . Và các khối này lại được sắp xếp để khôi phục ảnh với kích thước ban đầu 768x1280x3

Tanh-23	[-1, 32, 32, 32]	0
Conv2d-24	[-1, 64, 30, 30]	18,496
LeakyReLU-25	[-1, 64, 30, 30]	0
ZeroPad2d-26	[-1, 64, 32, 32]	0
ConvTranspose2d-27	[-1, 128, 64, 64]	32,896
ZeroPad2d-28	[-1, 128, 66, 66]	0
Conv2d-29	[-1, 128, 64, 64]	147,584
LeakyReLU-30	[-1, 128, 64, 64]	0
ZeroPad2d-31	[-1, 128, 66, 66]	0
Conv2d-32	[-1, 128, 64, 64]	147,584
ZeroPad2d-33	[-1, 128, 66, 66]	0
Conv2d-34	[-1, 128, 64, 64]	147,584
LeakyReLU-35	[-1, 128, 64, 64]	0
ZeroPad2d-36	[-1, 128, 66, 66]	0
Conv2d-37	[-1, 128, 64, 64]	147,584
ZeroPad2d-38	[-1, 128, 66, 66]	0
Conv2d-39	[-1, 128, 64, 64]	147,584
LeakyReLU-40	[-1, 128, 64, 64]	0
ZeroPad2d-41	[-1, 128, 66, 66]	0
Conv2d-42	[-1, 128, 64, 64]	147,584
Conv2d-43	[-1, 32, 62, 62]	36,896
LeakyReLU-44	[-1, 32, 62, 62]	0
ZeroPad2d-45	[-1, 32, 64, 64]	0
ConvTranspose2d-46	[-1, 256, 128, 128]	33,024
Conv2d-47	[-1, 16, 126, 126]	36,880
LeakyReLU-48	[-1, 16, 126, 126]	0
ReflectionPad2d-49	[-1, 16, 130, 130]	0
Conv2d-50	[-1, 3, 128, 128]	435
Tanh-51	[-1, 3, 128, 128]	0

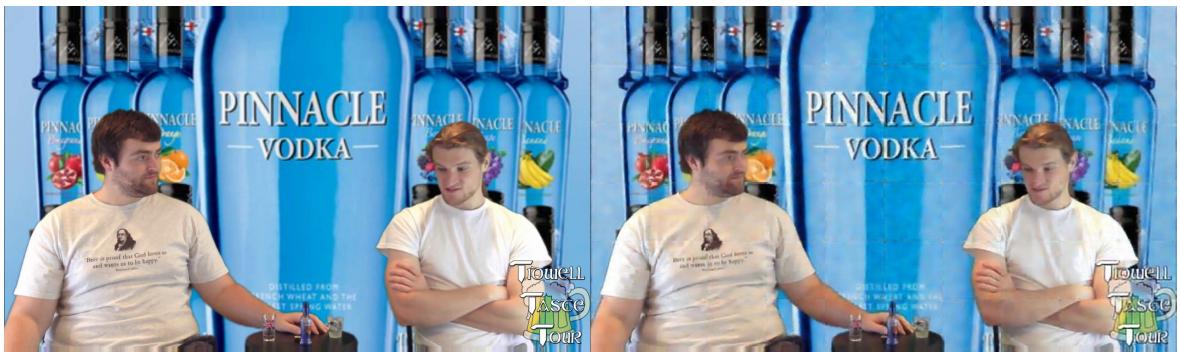
Hình 3.9. Các lớp mạng tham gia vào quá trình giải nén

3.2. Đánh giá hiệu suất nén

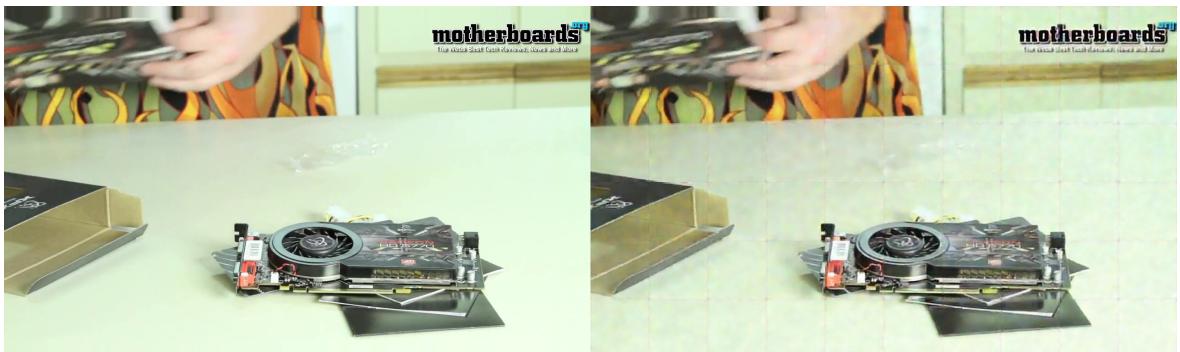
Sau khi thực hiện các quá trình nén và giải nén thì thu được kết quả như sau:



Hình 3.10. Ảnh khi qua kỷ nguyên đầu tiên



Hình 3.11. Ảnh khi qua kỷ nguyên thứ 250



Hình 3.12. Ảnh khi qua kỷ nguyên thứ 380



Hình 3.13. Ảnh khi qua kỷ nguyên cuối cùng

Chương 4

Kết luận và hướng phát triển⁷

4.1. Kết luận

Như vậy, với một mạng nơ-ron không quá phức tạp khoảng 2 triệu tham số cùng 50 lớp, chúng em đã hoàn thành huấn luyện một công cụ nén ảnh trên một máy tính phần cứng bình thường và GPU tốt do Google cung cấp miễn phí. Từ kết quả cho thấy hiệu năng nén được coi là chấp nhận được.

Nhưng vẫn còn một số phần ảnh chưa được hoàn chỉnh, do ghép nối các khối lại với nhau không có sự liên kết. Để khắc phục tình trạng này chúng em đề xuất sử dụng một số phương pháp nội suy để làm mờ các đường nối này.

4.2. Hướng phát triển

Dựa trên những cơ sở sẵn có này công cụ có thể được cải tiến trong tương lai bằng những phương pháp sau:

- Cải thiện thời gian chạy, huấn luyện của mô hình, nâng cấp lên có thể chạy trong thời gian ít hơn nếu có phần cứng tốt hơn
- Để cải thiện độ chính xác cho phần giải mã, sử dụng các phương pháp nội suy.
- Thử nghiệm với nhiều mô hình được huấn luyện trước và thuật toán huấn luyện khác nhau cho bộ dữ liệu của hệ thống.
- Thêm nhiều lớp mạng, để mạng nơ-ron lớn hơn, thông minh hơn.

Không chỉ dừng lại ở việc nén dữ liệu ảnh, mà còn trên nhiều loại dữ liệu khác, . . .

Tài liệu tham khảo

- [1] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.
- [2] R. J.Hassan and A. M. Abdulazeez. Deep Learning Convolutional Neural Network for Face Recognition: A Review. *International Journal of Science and Business*, 5(2):114–127, 2021.
- [3] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980, Dec. 2014.
- [4] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv e-prints*, page arXiv:1212.5701, Dec. 2012.