TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BẢO MẬT THÔNG TIN**

**QR CODE**

*Người hướng dẫn*: **Cô Huỳnh Ngọc Tú**

*Người thực hiện*: **Nguyễn Duy Hàn Lâm (MSHV: 196005004)**

Khoá  **: 2019**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BẢO MẬT THÔNG TIN**

**QR CODE**

*Người hướng dẫn*: **Cô Huỳnh Ngọc Tú**

*Người thực hiện*: **Nguyễn Duy Hàn Lâm (MSHV: 196005004)**

Khoá  **: 2019**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của cô Huỳnh Ngọc Tú. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 17 tháng 11 năm 2020*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Duy Hàn Lâm*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

MỤC LỤC

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN ii](#_Toc38785913)

[MỤC LỤC 1](#_Toc38785914)

[CHƯƠNG 1: DỊCH MÁY NEURAL 4](#_Toc38785915)

[1.1 Giới thiệu về dịch máy neural 4](#_Toc38785916)

[1.2 Recurrent Neural Network trong mô hình 5](#_Toc38785917)

[1.2.1 Cổng update 7](#_Toc38785918)

[1.2.2 Cổng reset 7](#_Toc38785919)

[1.2.3 Xuất kết quả 8](#_Toc38785920)

[1.3 Convolutional Neural Network trong mô hình 9](#_Toc38785921)

[1.4 Mô hình Encoder-Decoder 11](#_Toc38785922)

[1.4.1 Word embedding 12](#_Toc38785923)

[1.4.2 Encoder 13](#_Toc38785924)

[1.4.3 Decoder 13](#_Toc38785925)

[1.4.4 Linguistic Input Feature 16](#_Toc38785926)

[1.5 Tóm tắt chương 1 18](#_Toc38785927)

[CHƯƠNG 2: THỰC HIỆN MÔ PHỎNG ENCODER-DECODER 19](#_Toc38785928)

[2.1 Chuẩn bị nguồn dữ liệu và thư viện 19](#_Toc38785929)

[2.2 Xử lý dữ liệu và tạo các đối tượng quản lý 19](#_Toc38785930)

[2.3 Xây dựng encoder-decoder 23](#_Toc38785931)

[2.4 Huấn luyện 27](#_Toc38785932)

[2.5 Đánh giá 33](#_Toc38785933)

[2.6 Kiểm thử 35](#_Toc38785934)

[2.7 Chạy chương trình và kết quả 36](#_Toc38785935)

[2.8 Tóm tắt 37](#_Toc38785936)

[TÀI LIỆU THAM KHẢO 39](#_Toc38785937)

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1. Sơ lược cấu tạo hoạt động của RNN [1] 5](#_Toc38785871)

[Hình 1.2. Cấu trúc neural theo GRU [2] 6](#_Toc38785872)

[Hình 1.3. Mô tả cách tính của cổng update [2] 7](#_Toc38785873)

[Hình 1.4. Mô tả cách tính cổng reset [2] 8](#_Toc38785874)

[Hình 1.5. Mô tả cách tính [2] 8](#_Toc38785875)

[Hình 1.6. Cấu trúc của CNN trong encoder [3] 9](#_Toc38785876)

[Hình 1.7. Minh họa cách hoạt động của hàm activation ở mỗi node [3] 10](#_Toc38785877)

[Hình 1.8. Minh họa cho một số cấu trúc của CNN [3] 11](#_Toc38785878)

[Hình 1.9. Mô hình encoder-decoder cơ bản [3] 12](#_Toc38785879)

[Hình 1.10. Cấu trúc của một bộ GRU-RNN encoder [4] 13](#_Toc38785880)

[Hình 1.11. Minh họa cho decoder [4] 14](#_Toc38785881)

[Hình 1.12. Cơ chế attention [5] 15](#_Toc38785882)

[Hình 1.13. Cấu trúc chi tiết của một attention [6] 15](#_Toc38785883)

[Hình 2.1. Các thư viện cần chuẩn bị 19](#_Toc38785884)

[Hình 2.2. Các thư viện phục vụ linguistic input feature 19](#_Toc38785885)

[Hình 2.3. Class quản lý dữ liệu 20](#_Toc38785886)

[Hình 2.4. Xử lý và chuẩn hóa dữ liệu 21](#_Toc38785887)

[Hình 2.5. Hàm đọc dữ liệu 22](#_Toc38785888)

[Hình 2.6. Chuẩn hóa số lượng từ của câu và lemmas hóa dữ liệu 22](#_Toc38785889)

[Hình 2.7. Hàm đọc dữ liệu và thực hiện nạp dữ liệu vào chương trình 23](#_Toc38785890)

[Hình 2.8. Xây dựng bộ encoder 23](#_Toc38785891)

[Hình 2.9. Xây dựng bộ decoder cơ bản 24](#_Toc38785892)

[Hình 2.10. Cấu trúc hoạt động của attention 25](#_Toc38785893)

[Hình 2.11. Class decoder sử dụng attention (hàm khởi tạo của class) 26](#_Toc38785894)

[Hình 2.12. Hàm khởi tạo tiến trình và hàm feedforward 27](#_Toc38785895)

[Hình 2.13. Các hàm tạo vector input và output cho quá trình huấn luyện 28](#_Toc38785896)

[Hình 2.14. Hàm thực hiện huấn luyện 29](#_Toc38785897)

[Hình 2.15. Giai đoạn tiền encoder 29](#_Toc38785898)

[Hình 2.16. Giai đoạn chạy encoder 30](#_Toc38785899)

[Hình 2.17. Giai đoạn hậu encoder và tiền decoder 30](#_Toc38785900)

[Hình 2.18. Giai đoạn chạy decoder 30](#_Toc38785901)

[Hình 2.19. Giai đoạn hậu decoder 31](#_Toc38785902)

[Hình 2.20. Hàm thông báo lượng thời gian chương trình đã và đang chạy 31](#_Toc38785903)

[Hình 2.21. Hàm vẽ biểu đồ thay đổi của giá trị loss 32](#_Toc38785904)

[Hình 2.22. Hàm huấn luyện tổng quát 33](#_Toc38785905)

[Hình 2.23. Hàm thực hiện đánh giá 34](#_Toc38785906)

[Hình 2.24. Hàm thực hiện chọn đánh giá ngẫu nhiên 35](#_Toc38785907)

[Hình 2.25. Các hàm phục vụ trình bày kết quả 35](#_Toc38785908)

[Hình 2.26. Tiến hành huấn luyện mô hình 36](#_Toc38785909)

[Hình 2.27. Kết quả của từng đợt huấn luyện 36](#_Toc38785910)

[Hình 2.28. Chạy thử 1 câu ví dụ 37](#_Toc38785911)

[Hình 2.29. Kết quả chạy thử 1 câu ví dụ 37](#_Toc38785912)

TÓM TẮT NỘI DUNG

QR code là một công nghệ mã vạch của Nhật Bản được phát minh vào năm 1994. Tiền thân của QR code là barcode. Do những hạn chế về lượng dữ liệu mà barcode chỉ có thể được lưu vì vậy QR code ra đời nhằm mở rộng và tăng kích thước lưu trữ dữ liệu phục vụ được nhiều mục đích hơn của đời sống.

Tài liệu sẽ giới thiệu về barcode, sau đó sẽ giới thiệu về QR code và cuối cùng trình bày một số rủi ro có thể gặp phải khi sử dụng QR code trong cuộc sống.

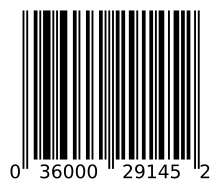
Chương 1: QR CODE

QR code xuất phát từ tiền thân là barcode. Vì thế đầu tiên, ta tìm hiểu sơ thế nào là barcode.

1.1 Barcode

Barcode là một phương pháp biểu diễn dữ liệu ở dạng trực quan mà ở dạng này máy vẫn có thể đọc được. Dữ liệu sẽ được biểu diễn thành các đường vạch có độ dài, độ rộng khác nhau và được in song song nhau. Những mã vạch này được sắp xếp thành 1 hàng hay 1 chiều.

Mã vạch được phát minh bởi Norman Joseph Woodland và Bernard Silver và được cấp bằng sáng chế tại Hoa Kỳ vào năm 1951.



1. Minh họa về barcode

Ta có thể thấy qua hình 1.1, dữ liệu “036000291452” được biểu diễn (mã hóa) thành các mã vạch có độ rộng khác nhau và được xếp thành ngay 1 hàng.

Về mặt vật lý, mã vạch được tạo thành từ một loạt các đường có độ rộng khác nhau và tương ứng với các kí tự số và chữ khác nhau, sau đó có thể đọc được bằng các máy quét mã vạch laser.

Code 128 là một trong những phương pháp thường dùng, rất hiệu quả trong việc mã hóa các dữ liệu về chữ và số. Code 128 bao gồm 2 giai đoạn: một là kiểm tra giá trị checksum và hai là kiểm tra các byte mã hóa.

Các thành phần của code 128 gồm các thành phần sau:

* Vùng yên tĩnh (“Quiet zone”) là vùng trống nằm ở phía trước và sau của các thanh barcode; vùng trống này sẽ cho phép máy quét thiết lập các giá trị cơ bản như màu sắc và độ phản chiếu của vật liệu mà chúng sẽ đọc. Những con số này được sử dụng để giúp máy quét xác định được đâu là các khoảng trắng (space) và đâu là các mã vạch (bar).
* Theo sau quiet zone sẽ là vùng start code. Start code cho chương trình biết bắt đầu quá trình mã hóa cũng như khởi tạo chế độ mã hóa tương ứng. Start code sẽ là một ký tự trong 3 kí tự “A” (giá trị tương ứng là 103), “B” (giá trị tương ứng là 104), “C” (giá trị tương ứng là 105) đại diện cho 3 kiểu mã hóa khác nhau.
* Theo sau start code sẽ là phần dữ liệu sẽ được mã hóa. Dữ liệu sẽ được mã hóa đưa về dạng nhị phân các số 0 và 1. Phần dữ liệu ứng với 1 sẽ được kẻ thành 1 vạch (bar) và phần dữ liệu 0 sẽ không được kẻ vạch.
* Tiếp theo sẽ là phần checksum. Checksum là một giá trị dung để xác minh xem dữ liệu vừa được encode có được chính xác hay không.
* Cuối cùng là phần stop code cho biết kết thúc phiên mã hóa.

Ví dụ ta có dữ liệu cần được barcode hóa là “CSE370” với kí tự bắt đầu mã hóa là “A” (103). Ta sẽ tiến hành như sau:

|  |  |  |
| --- | --- | --- |
| Ký tự | Giá trị | Mã nhị phân tương ứng |
| A (START\_A) | 103 | 11010000100 |
| C (35) | 35 x 1 = 35 | 10001000110 |
| S (51) | 51 x 2 = 102 | 11011101000 |
| E (37) | 37 x 3 = 111 | 10001101000 |
| 3 (19) | 19 x 4 = 76 | 11001011100 |
| 7 (23) | 23 x 5 = 115 | 11101101110 |
| 0 (16) | 16 x 6 = 96 | 10011101100 |
| Checksum | 535 mod 103 = 20 | 11001001110 |
| STOP\_CODE |  | 1100011101011 |

Cuối cùng ta sẽ thay các kí tự 1 thành các 1 vạch mảnh (bar). Ta sẽ được barcode như hình sau:



1. Kết quả barcode sau khi thực hiện

Ta có thể thấy kết quả của barcode thật chất là dữ liệu dưới dạng mã nhị phân, khi quét ứng với mỗi vạch mảnh ta sẽ được các kí tự 1 còn phần trắng sẽ là kí tự 0. Vì vậy sau khi quét hoàn chỉnh ta sẽ được một chuỗi các giá trị 0 và 1. Từ đó máy sẽ truy ngược lại các đoạn mã nhị phân về lại các giá trị dữ liệu thường bằng cách tra bảng. Cuối cùng dựa vào START\_CODE cũng như CHECKSUM ta sẽ tính lại được giá trị tổng ban đầu của dữ liệu để xác minh xem việc đọc barcode có đúng hay không.

1.2 QR code

Kĩ thuật barcode đã giúp con người có thể lưu trữ thông tin thành dạng mà máy từ, laser có thể đọc được. Lượng dữ liệu càng nhiều thì số lượng vạch cũng như phạm vi cần để viết barcode sẽ càng dài. Dữ liệu càng nhiều, barcode sẽ càng dài cần yêu cầu đầu đọc có khả năng quét phạm vi càng lớn. Việc hạn chế này về đầu đọc khiến ta cần phải cải tiến cách mã hóa barcode sao cho vừa có thể chứa được nhiều dữ liệu đồng thời đầu đọc vẫn có thể đọc và tiếp thu được tốt.

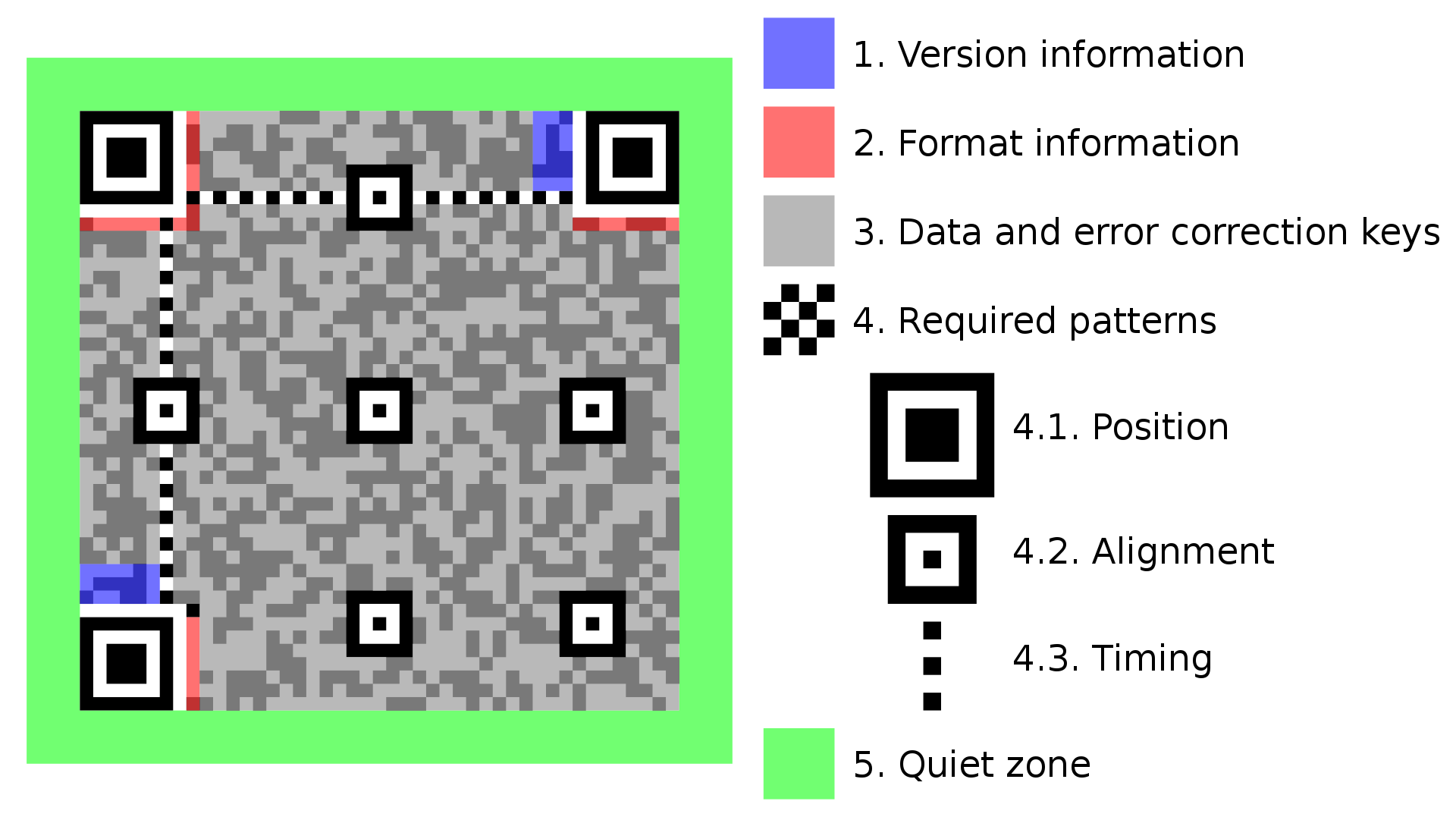
Sau nhiều ngày đêm nghiên cứu tìm hiểu khuôn mẫu cho dữ liệu, Mashiro Hara và nhóm cộng sự đã tìm ra được một khuôn mẫu barcode mới mà không những cho phép tăng lượng dữ liệu có thể lưu trữ được mà mẫu barcode mới này còn nhỏ gọn và dễ dàng cho việc đọc và quét của đầu đọc.

Mẫu barcode này được gọi là QR code (Quick Response code) vì khả năng đọc và phản hồi nhanh của đầu đọc khi làm việc trên các barcode này.

Mẫu QR code này cho phép lưu trữ dữ liệu thành 2 chiều ngang và dọc thay vì chỉ một chiều ngang như barcode. Điều này giúp thỏa mãn được nhu cầu nhiều dữ liệu hơn của người dùng.

1.3 Cấu tạo của QR code

Mô hình cấu tạo của 1 QR code cơ bản như sau:

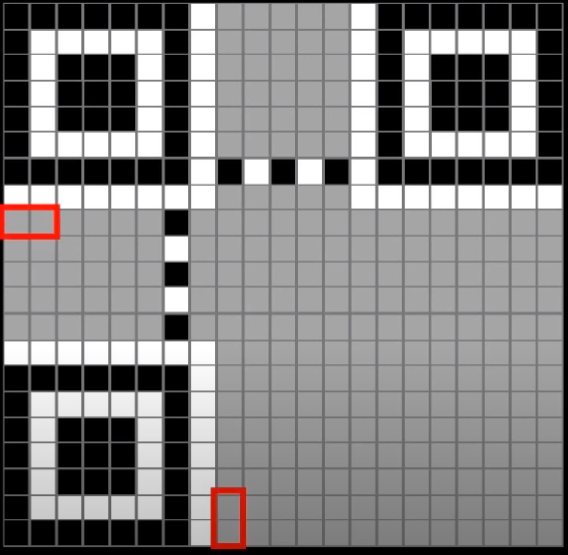


1. Cấu tạo của 1 QR code [1]

1 QR code cơ bản theo hình 1.3 sẽ gồm 5 phân vùng chính:

* Phân vùng cho biết phiên bản (version) mà QR code sẽ được sử dụng (phần tô màu xanh tím trong hình 1.3).
* Phân vùng định hình cho biết các tham số cấu hình của QR code sẽ sử dụng (phần tô màu hồng trong hình 1.3).
* Phần vùng lưu trữ dữ liệu và các giá trị hiệu chỉnh sai số (phần tô màu xám trong hình 1.3).
* Phân vùng các cấu trúc đặc biệt: các khối giúp định hình (position) vị trí cho QR code (các khối 4.1 trong hình 1.3), các khối tạo sự liên kết (alignment) cho QR code (các khối 4.2 trong hình 1.3). Các khối position và alignment giúp đầu đọc định hình được hình dáng, vị trí, chiều của QR code. Nối giữa các khối position và alignment sẽ là các hàng timing. Trên các hàng timing, các pixel sẽ được tô đen trắng xen kẽ nhau. Trong 1 QR code timing line sẽ giúp đầu đọc xác nhận lại version mà QR code đang sử dụng cũng như tốc độ mà đầu đọc sẽ đọc các bit.
* Quiet zone: tương tự với barcode, quiet zone giúp đầu đọc phân biệt ranh giới giữa phạm vi giữa QR code với các vật thể khác. Cũng như là phần trống để đầu đọc có thể chuẩn bị các tham số phụ.

1.3.1 Phân vùng cấu hình



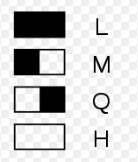
1. Lưu trữ giá trị khả năng phục hồi dữ liệu

Trong phân vùng cấu hình của QR code sẽ dàng riêng ra 4 pixel (2 pixel ở vùng cấu hình trên và 2 pixel ở vùng cấu hình dưới) để thông báo mức độ data có thể phục hồi được.

Có 4 mức độ hồi phục dữ liệu của QR code:

* QR code chỉ có thể hồi phục 7% dữ liệu (Low – L).
* QR code chỉ có thể hồi phục 15% dữ liệu (Medium – M).
* QR code chỉ có thể hồi phục 25% dữ liệu (Quartile – Q).
* QR code chỉ có thể hồi phục 30% dữ liệu (High – H).

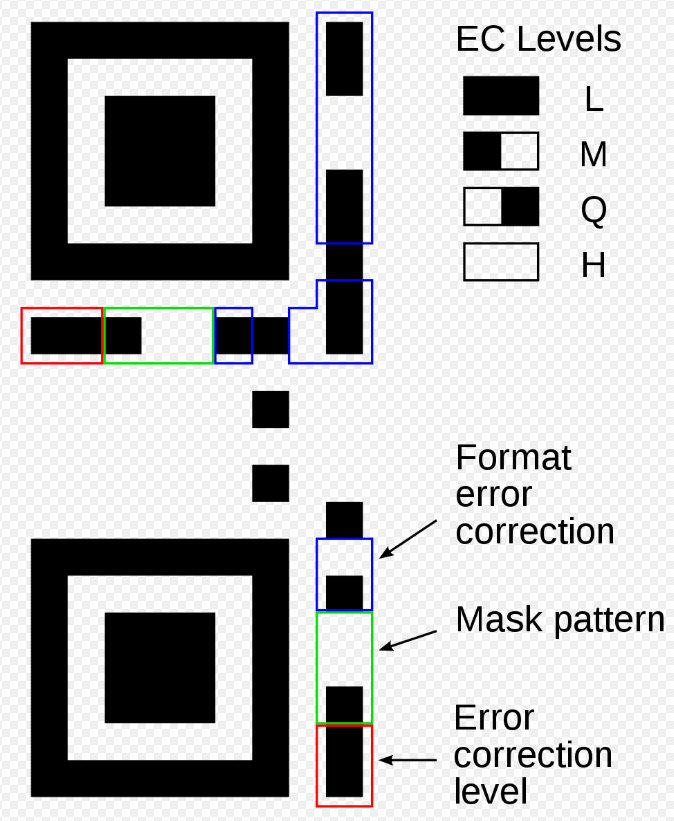
Cách tô pixel tương ứng dành cho các mức phục hồi dữ liệu được mô ta như sau:



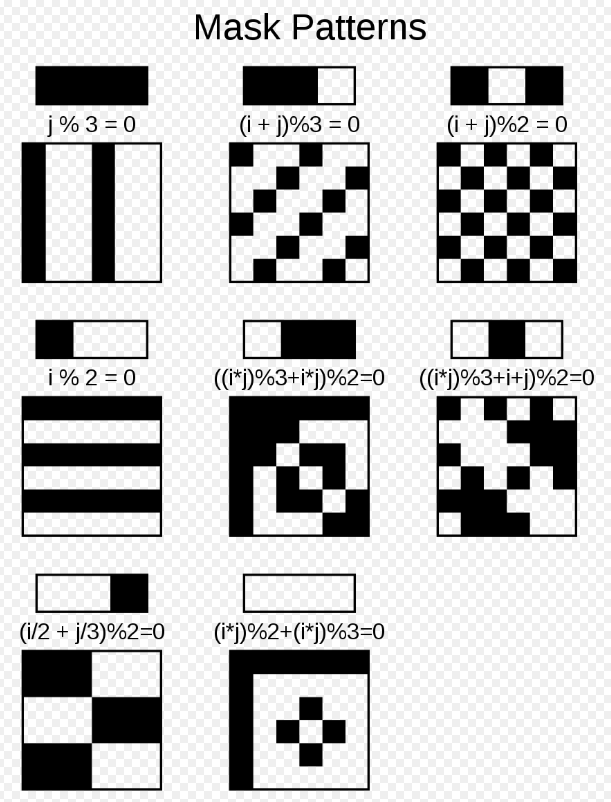
1. Các mức phục hồi dữ liệu [1]

* Mức L ta sẽ tô cả 4 ô màu đen.
* Mức M ta sẽ tô đen ở ô bên trái của 2 ô vùng trên và ở ô bên dưới của 2 ô vùng dưới.
* Mức Q ta sẽ tô đen ở ô bên phải của 2 ô vùng trên và ở ô bên trên của 2 ô vùng dưới.
* Mức H ta sẽ để trắng cả 4 ô.

Trong phân vùng cấu hình ta sẽ dùng tiếp 6 pixel (tô màu xanh lá trong hình 1.6) để cấu hình cho giá trị mask pattern. Mask pattern cho biết dữ liệu sẽ được phân phối như thế nào trong QR code. Có tất cả 8 kiểu phân phối của dữ liệu trong QR code. Ứng với mỗi kiểu phân phối dữ liệu sẽ có 8 kiểu mask patter khác nhau.



1. Mô tả phân vùng cấu hình của QR code [1]



1. Các loại phân phối dữ liệu của QR code [1]

Trong hình 1.7 mô tả các loại phân phối dữ liệu thường gặp ở QR code. Ví dụ ứng với mask pattern là (j % 3 = 0) QR code sẽ được phân như sau: trong 3 hàng dữ liệu thì sẽ có 1 hàng là ranh giới, 2 hàng còn lại để phân bổ dữ liệu vào. Mask pattern giúp đầu đọc biết được đâu là ranh giới và đâu là phần dữ liệu.

Phần còn lại của phân vùng cấu hình sẽ được dùng để cấu hình cho việc hiệu chỉnh sai số.

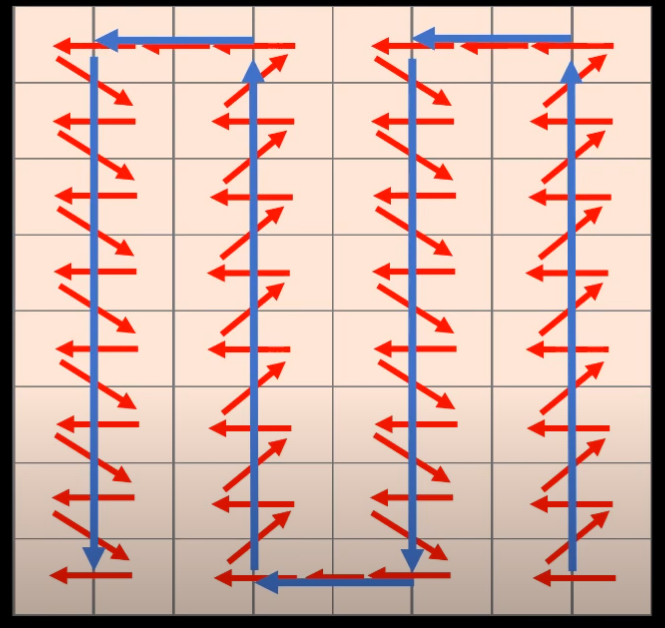
1.3.2 Hiệu chỉnh sai số

QR code sử dụng thuật toán hiệu chỉnh sai số Reed-Solomon trong trường hữu hạn . Các phần tử được mã hóa dưới dạng các byte 8 bit. 1 byte có giá trị là được dùng để mã hóa cho 1 giá trị với thỏa mãn biểu thức .

1.3.3 Mã hóa

Sau khi hoàn tất quá trình cấu hình, ta sẽ tiến hành chia các phần còn lại của QR code thành các mảnh bit lưu trữ thông tin.

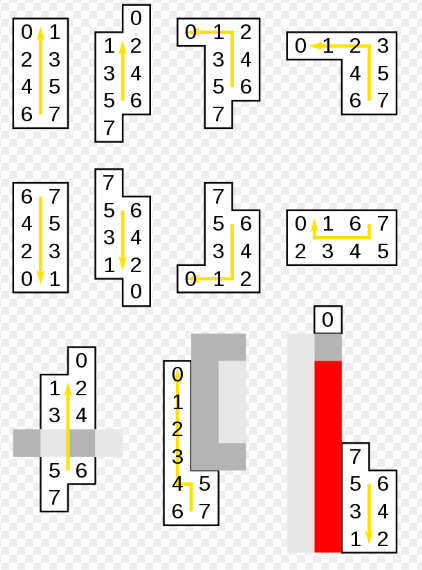
Thông tin trong các mảnh này sẽ được ghi theo đường zigzag. Bắt đầu từ ô bên phải dưới cùng, sau đó đến ô bên trái, xong rồi chéo lên trên ô bên phải rồi di chuyển sang ô bên trái. Để dễ hiểu ta có thể theo xem ví dụ minh họa sau:



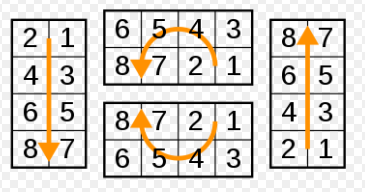
1. Chiều mã hóa của thông tin trong QR code

QR code mã hóa thông tin thành các bit nhị phân. Sau đó sẽ ghi các bit này theo chiều mã hóa được mô tả như hình 1.8. các bit sẽ được ghi theo chiều zigzag, bắt đầu từ ô dưới cùng bên phải, sau đó đến ô bên trái kế bên và bắt chéo lên lại ở ô bên phải. Khi đến hàng cuối cùng của phân vùng ghi thông tin, các bit sẽ bắt đầu ghi từ ô kế tiếp của ô bên trái, xong rồi ghi tiếp sang 1 ô nữa, xong bắt chéo xuống ô bên phải.

Hình 1.9 và 1.10 sau miêu tả các kiểu ghi bit thường gặp ở QR code:



1. Thứ tự ghi các bit mã hóa của dữ liệu trong các trường hợp đặc biệt [1]



1. Thứ tự ghi các bit mã hóa của dữ liệu trong QR code

Sau khi hoàn tất các mảnh mã hóa, ta sẽ ráp các mảnh này sao cho khớp vào mô hình. Cuối cùng ta đã được 1 QR code hoàn chỉnh.

Trong QR code sẽ có 2 loại mảnh bit chính: một là các mảnh bit chứa thông tin chính và hai là loại mảnh bit chứa các giá trị để hiệu chỉnh sai số của mã hóa theo thuật toán tương ứng. Tùy vào cấu trúc cũng như phiên bản của QR code mà các loại mảnh này sẽ phân bố theo các kiểu nhất định.

1.4 Tổng kết chương 1

Sau chương 1 ta đã tìm hiểu nguồn gốc, cấu tạo hoạt động của barcode. Từ đó hiểu được những mặt hạn chế của barcode.

Do nhu cầu tăng thêm lượng dữ trự về mặt dữ liệu cũng như hạn chế về thiết bị đọc nên barcode mở rộng từ lưu trữ dữ liệu từ 1 chiều sang thành 2 chiều. barcode này được gọi là QR code.

Do được mở rộng phạm vi lưu trữ nên QR code có khả năng lưu trữ dữ liệu nhiều hơn.

Bên cạnh đó, việc lưu trữ các bit hiệu chỉnh sai số song song với bit dữ liệu chính, QR code có khả năng phục hồi được dữ liệu nhanh hơn và có thời gian đáp ứng tốt.

CHƯƠNG 2: QR CODE VÀ ỨNG DỤNG

Ở chương trước ta đã tìm hiểu cấu tạo cũng như hoạt động của QR code. Trong chương này ta sẽ đi tìm hiểu các ứng dụng của QR code và thực hiện xây dựng 1 ứng dụng nhỏ về QR code.

2.1 Ứng dụng của QR code

Với việc CNN ngày càng trở thành một giải thuật thông dụng trong lĩnh vực thị giác máy tính, một số cải tiến đã được thực hiện để cải thiện độ chính xác của kiến trúc CNN ban đầu của Krizhevsky et al. (2012). Các cải tiến tốt nhất (Zeiler & Fergus, 2013; Sermanet và cộng sự, 2014) trong cuộc thi ILSVRC-2013 đã sử dụng các kĩ thuật như sử dụng window size nhỏ hơn hay sử dụng bước trượt (stride) cho window size nhỏ hơn của lớp phức hợp đầu tiên. Một dòng cải tiến khác nghiên cưu về mật độ của ảnh sẽ ảnh hưởng như thế nào đến quá trình đào tạo và kiểm thử của các mạng neural (Sermanet và cộng sự, 2014; Howard, 2014).

Thay vì chú trọng cải tiến các kĩ thuật để giúp việc nhận dạng tốt hơn thì VGG sẽ nghiên cứu về chiều sâu của mô hình. Để đạt được mục tiêu này, mô hình VGG sẽ chập nhiều lớp mô hình CNN lại với nhau để củng cố về chiều sâu. Bên cạnh đó các lớp CNN của VGG sẽ sử dụng các window có kích thước nhỏ 3 x 3 để trích xuất ảnh.

VGG không chỉ đạt được độ chính xác cao đối với các tác vụ phân loại mà còn có thể áp dụng cho các bộ dữ liệu nhận dạng hình ảnh khác nhau mà vẫn đạt được hiệu suất tốt.

2.2 Cấu trúc của mô hình VGG

Trong quá trình đào tạo, đầu vào cho CNN của ta sẽ là hình ảnh RGB 224 × 224 có kích thước cố định. Quá trình tiền xử lý duy nhất mà ta sẽ thực hiện đó là lấy từng pixel trừ đi giá trị RGB trung bình (được tính trên tập huấn luyện). Sau đó tập các pixel này sẽ được chuyển qua một chồng các lớp CNN, trong đó ta sẽ sử dụng các bộ lọc có trường tiếp nhận (window) rất nhỏ: 3 × 3 (là kích thước nhỏ nhất để nắm bắt các thay đổi từ trái sang phải, từ trên xuống dưới, trung tâm của bức ảnh). Đầu vào của mỗi lớp, ta sẽ sử dụng bộ lọc CNN có window có kích thước là 1 × 1 như một phép biến đổi tuyến tính của các kênh đầu vào. Kết quả của mỗi lần trượt window sẽ được cố định thành 1 pixel; các kết quả trượt đầu ra của CNN sẽ phải đảm bảo đúng với kích thước đã được khai báo, hay nói cách khác rõ ràng hơn đó là 1 giá trị pixel ở đầu ra sẽ tương ứng với 1 lần trượt của window 3 × 3. Sau khi kết thúc 1 khối các lớp CNN ta sẽ tiến hành 1 lớp tổng hợp (pooling layer). Có tổng cộng năm lớp tổng hợp tối đa (max-pooling layer), theo sau một khối các lớp CNN (không phải tất cả các khối CNN đều được theo sau đó bởi các max-pooling layer). Các max-pooling layer sẽ được thực hiện trên các window có kích thước (size) 2 × 2 pixel, và bước trượt ở mỗi lần trượt là 2.

Một chồng các lớp chập (có độ sâu khác nhau trong các kiến trúc khác nhau) được theo sau bởi ba lớp kết nối đầy đủ (Full-Connected (FC) layer): hai lớp đầu tiên có 4096 kênh ở mỗi lớp, lớp thứ ba sẽ thực hiện phân loại ILSVRC 1000 chiều do đó nó sẽ chứa 1000 kênh (một cho mỗi đối tượng label). Lớp cuối cùng là lớp soft-max. Cấu hình của các FC lớp là giống nhau trong tất cả các mô hình mạng VGG.

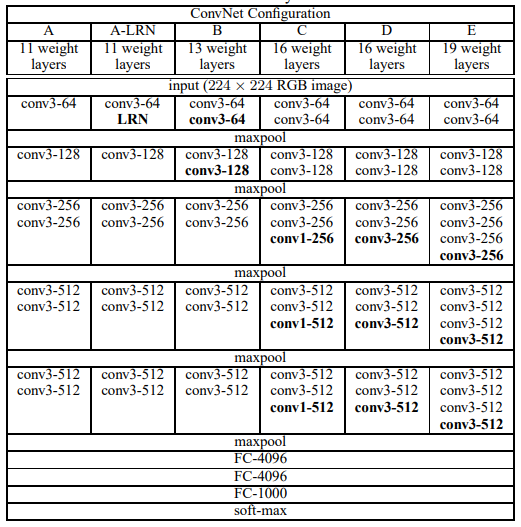
Tất cả các lớp ẩn đều được xử lý phi tuyến tính (ReLU (Krizhevsky et al., 2012))) trước khi xuất kết quả ở đầu ra.

2.3 Các loại mô hình VGG

Dựa trên số lớp CNN và FC được sử dụng ta có các loại mô hình VGG sau:

* VGG11: 11 lớp (8 lớp CNN, 3 lớp FC).
* VGG13: 13 lớp (10 lớp CNN, 3 lớp FC).
* VGG16: 16 lớp (13 lớp CNN, 3 lớp FC).
* VGG19: 19 lớp (16 lớp CNN, 3 lớp FC).

Sau đây là hình tổng quát thông tin của các loại của mô hình VGG:



1. Thông tin về từng loại mô hình VGG [1]

Trong hình 2.1, các loại mô hình sẽ được đánh số từ A, B cho đến E ứng với việc tăng dần số lượng lớp CNN của mô hình. Ngoài ra con số sau chữ “conv” biểu thị window size mà lớp CNN đó sẽ sử dụng. Ứng với “conv3” sẽ là window size 3x3. Tương tự, ứng với “conv1” sẽ là window size 1x1. Các con số 64, 128, 256, 512 biểu số lượng bộ lọc (filter) mà lớp CNN sẽ sử dụng. LRN trong hình 2.1 biểu thị cho chuẩn hóa Local Response (Local Response Normalisation do Krizhevsky và cộng sự khởi xướng vào năm 2012) sẽ được áp dụng vào mô hình.

2.4 Một số thông tin bổ sung

Một chồng hai lớp CNN 3×3 (không sử dụng pooling) tương đương với 1 lớp CNN 5 × 5; ba lớp CNN 3 × 3 như vậy tương đương với 1 lớp CNN 7 × 7. Vì vậy, tại sao ta lại không thay một chồng ba lớp CNN 3 × 3 bằng một lớp 7 × 7? Đầu tiên, việc ta kết hợp ba lớp rectification layers phi tuyến tính thay vì một lớp duy nhất sẽ giúp việc nhận dạng trở nên tốt hơn. Thứ hai, chúng ta sẽ giúp làm giảm được số lượng tham số: giả sử rằng cả đầu vào và đầu ra của một khối 3 lớp CNN 3 × 3 có C kênh ở mỗi lớp, số lượng trọng số sẽ là ; thay vì, một lớp CNN 7 × 7 sẽ yêu cầu các tham số, tức nhiều hơn 81% số lượng tham số. Do mô hình CNN 7x7 có nhiều tham số nên dẫn tới việc tính toán phải nhiều hơn cũng như ảnh hưởng đến độ chính xác, vì thế CNN 7x7 buộc chúng phải có sự phân rã chiều dữ liệu thông qua các bộ lọc 3 × 3 (việc phân rã này sẽ không có các lớp phi tuyến tính xen vào giữa).

Việc kết hợp các lớp CNN 1 × 1 (mô hình C, trong hình 2.1) là một cách để tăng tính quyết định mà không ảnh hưởng đến các trường tiếp nhận của các lớp CNN. Mặc dù trong trường hợp sử dụng các lớp CNN 1 × 1 về cơ bản là một phép chiếu tuyến tính lên không gian có cùng chiều (số lượng kênh đầu vào và đầu ra là như nhau).

2.5 Tóm tắt chương 2

Sau chương 2 ta đã tìm hiểu thế nào là một mô hình VGG cơ bản. Cấu trúc của VGG gồm 2 phần là các lớp CNN và các lớp pooling.

Các lớp CNN gồm 2 loại là CNN dùng window size 3x3 và CNN dùng window size 1x1. Nếu phân theo số lượng bộ lọc (filter) thì các lớp CNN được phân thành 4 loại: CNN dùng 64 filter, CNN dùng 128 filter, CNN dùng 256 filter và CNN dùng 512 filter.

Các window của lớp CNN có bước trượt (stride) là 1. Và window của lớp max pooling có stride là 2.

Ngoài ra ta còn biết các loại mô hình khác nhau của VGG. Loại 11 lớp (8 lớp CNN, 3 lớp max-pooling), loại 13 (10 lớp CNN, 3 lớp max-pooling), loại 16 (13 lớp CNN, 3 lớp max-pooling), loại 19 (16 lớp CNN, 3 lớp max-pooling).

Nếu gộp nhiều lớp CNN có window size nhỏ thành 1 lớp CNN có window size sẽ ảnh hưởng xấu tới việc tính toán và độ chính xác do có nhiều tham số hơn cần xử lý.

CHƯƠNG 3: TIẾN HÀNH BÀI TOÁN PHÂN LOẠI

Trong chương trước ta đã tìm hiều những lý thuyết trọng tâm của mô hình VGG dùng để nhận dạng.

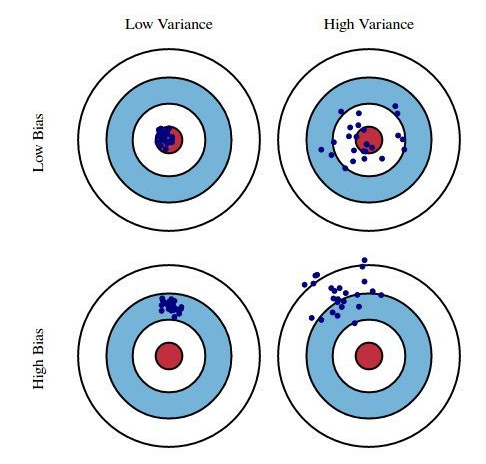
Trong chương này ta sẽ dùng kết quả của mô hình VGG ở chương trước để tiến hành quá trình phân loại và dán nhãn cho bức ảnh.

* 1. Bias và variance

Trong tính toán ta có khái niệm bias và variance.

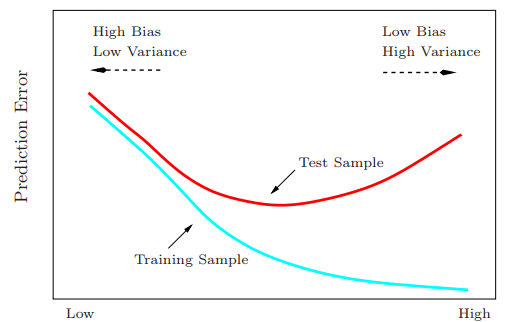
Bias là giá trị biểu thị độ chênh lệch giữa giá trị mong đợi với giá trị dự đoán. Gọi T là giá trị thống kê của tham số . Ta có , trong đó biểu thị cho giá trị mong đợi của T, biểu thị cho bias của .

Var (Variance) biểu thị cho mức độ phân tán của các giá trị tính toán được từ mô hình.



1. Minh họa về bias và variance [2]

Rõ ràng, cả sai lệch (bias) và phương sai (variance) đều có thể gây hại cho hiệu suất dự đoán của mô hình nếu chúng quá lớn. Điều này đặc biệt sẽ có thể xảy ra trong các mô hình có nhiều đặc tính (feature), hoặc các đặc tính của dữ liệu có mức độ liên quan với nhau cao.



1. Sự ảnh hưởng của bias và variance đến training và testing [2]

Phương pháp phân mảnh và chuẩn hóa là giải pháp có thể sẽ giúp làm giảm được phương sai của các giá trị dự đoán, nó sẽ làm tăng sai số của quá trình huấn luyện nhưng thay vào đó giá trị sai số tổng thể lại có thể được giảm xuống.

Một trong nhưng phương pháp phân mảnh (shrinkage method) mà tài liệu sẽ trình bày được gọi là Principal Component Analysis (PCA).

* 1. Principle Component Analysis (PCA)

Trong nhiều tình huống, đầu vào (input) của chúng ta thường sẽ có một số lượng lớn các thuộc tính (chiều) có tính tương quan với nhau. PCA lúc đầu sẽ tạo ra một số lượng nhỏ các tổ hợp tuyến tính của các thuộc tính đầu vào , và sau đó được thế bằng làm input cho quá trình hồi quy (regression).

Một kĩ thuật thường được sử dụng trong ridge regression là SVD (Singular Value Decomposition). Ma trận X khi áp dụng SVD sẽ có dạng:

Trong đó:

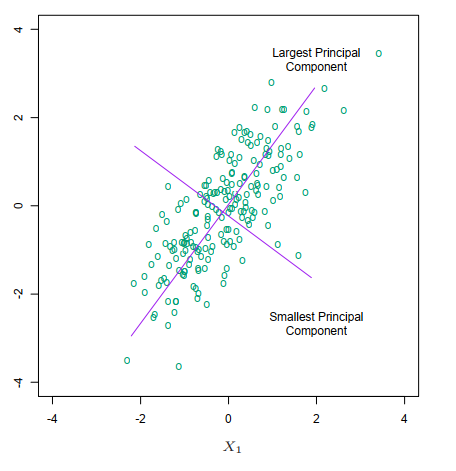
* U và V là 2 ma trận trực giao (orthogonal matrix).
* D là ma trận đường chéo có kích thước là . Với các giá trị đường chéo được gọi là các giá trị dị biến của X. Nếu có một giá trị trong D, thì X sẽ được coi là dị biến (singular).

Để thuận tiện ta sẽ thực hiện một vài phép tính ma trận ta hay gặp:

Do U là ma trận trực giao nên ta được . Ta được:

Do D là ma trận đường chéo có kích thước là nên . Do đó ta được:

Biểu thức (1) hay được gọi là eigen decomposition. Các eigen-vector (vector riêng) trong V còn được gọi là các thành phần chính (principal component) biểu thị cho các hướng của ma trận X. Trong đó hướng có thuộc tính có phương sai mẫu lớn nhất trong số tất cả các tổ hợp tuyến tính chuẩn hóa của X.



1. Minh họa về các principle component của một vài điểm dữ liệu [2]

Trong hình 3.3, ta có thể thấy principle component lớn nhất có hướng về phía có phương sai lớn nhất và principle component nhỏ nhất là hướng giúp làm giảm thiểu phương sai đó. Ridge regression sẽ chiếu y lên các thành phần này, và sau đó thu nhỏ hệ số của các thành phần có phương sai thấp hơn các thành phần có phương sai cao.

PCA tạo các thuộc tính bằng cách cho , và sau đó thực hiện bài toán hồi quy y trên tập input với M là một số nhỏ hơn hoặc bằng số chiều p của dữ liệu. Vì là trực giao, nên hồi quy này chỉ là tổng của các hồi quy đơn biến:

Với:

Là tỉ lệ của tích vô hướng giữa và y với tích vô hướng của với .

Từ đây ridge regression sẽ thu nhỏ các hệ số của các principle component, kích thước được thu nhỏ sẽ tùy thuộc vào kích thước của trị riêng tương ứng; PCA loại bỏ p – M các thành phần có giá trị riêng nhỏ nhất.

Sau khi thu nhỏ chiều của dữ liệu để nhằm tăng độ chính xác cho tổng thể bài toán, cuối cùng ta sẽ thực hiện việc phân loại (dán nhãn) bằng Support Vector Machine (SVM).

* 1. Support Vector Machine (SVM)

1. Khoảng cách từ một điểm tới siêu phẳng

Trong khôn gian, khoảng cách từ một điểm tới siêu phẳng (hyperlane) được tính bằng:

1. Bài toán phân chia 2 class

Giả sử rằng có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, tồn tại một siêu phẳng phân chia chính xác hai classes đó. Hãy tìm một siêu mặt phẳng phân chia hai classes đó, tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại.

Câu hỏi đặt ra là: trong vô số các mặt phân chia đó, đâu là mặt phân chia tốt ưu?

Chúng ta cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class tới đường phân chia là như nhau. Khoảng cách như nhau này được gọi là margin (lề).

Như vậy là ta đã giải quyết được 1 câu hỏi, nên câu hỏi tiếp theo là: vậy ta làm thế nào để margin này là tối ưu?

SVM sẽ giúp chúng ta trả lời được câu hỏi này.

1. Bài toán SVM

Ta có tập dữ liệu input gồm các dữ liệu có dạng . Với là dữ liệu đối tượng input (có d thuộc tính (chiều)), và là nhãn của nó.

Gọi là mặt phân cách giữa 2 class. Tạm phân 2 class thành một class nằm phía dương () và một class nằm phía âm (). Ta có:

Do ta tạm phân thành và nên luôn cùng dấu với . Dẫn tới cùng dấu với . Do đó .

Margin sẽ được tính:

Bài toán SVM mà ta cần tìm đó là tìm giá trị và sao cho margin là lớn nhất.

Ta giả sử với những điểm gần mặt phân chia nhất, dẫn tới với mọi ta có: .

Biểu thức (3) trở thành:

Ta có thể viết lại thành:

1. Hiệu chỉnh w và b

Với biểu thức (4) ta đã đưa bài toán về Lagrange.

Với

Từ đây ta hiệu chỉnh w và b bằng đạo hàm của hàm lagrange theo w và b:

Thay (6), (7) vào (5), ta được:

Đặt

Biểu thức (8) được viết lại thành:

Đặt , ta có:

Biểu thức (9) là một hàm concave.

Gọi tập và là số phần tử của S, với mỗi , ta có:

CHƯƠNG 4: FASTTEXT

4.1 Giới thiệu về fastText

fastText là một thư viện mã nguồn mở được sử dụng để huấn luyện các bài toán liên quan đến các vấn đề về xử lý phân loại văn bản cũng như biểu diễn từ. fastText được xây dựng bởi nhóm nghiên cứu mã nguồn mở của Facebook.

fastText là một thư viện hiệu quả trong xử lý các bài toán về word representation hay các bài toán về phân loại câu. fastText được viết bằng C ++ và hỗ trợ đa xử lý trong quá trình huấn luyện mô hình. fastText cho phép ta huấn luyện theo kiểu giám sát (supervise) hoặc không giám sát (unsupervised) của các từ và câu. Các biểu diễn này có thể được sử dụng cho nhiều ứng dụng.

FastText hỗ trợ đào tạo mô hình CBOW và mô hình Skip-gram bằng phương thức negative sampling, softmax hoặc phân cấp softmax.

Mỗi từ được biểu diễn dưới dạng một túi các cụm n-gram ký tự, ví dụ, đối với từ “matter”, với n = 3, sẽ gồm các biểu diễn cho các cụm: <ma, mat, att, tte, ter, er>. Từ khóa <and> sẽ được thêm dưới dạng ký hiệu ranh giới để phân biệt giữa biểu diễn của các cụm n-gram với biểu diễn từ từ của chính nó. Vì vậy, ví dụ, nếu cụm mat nếu tồn tại trong bộ từ điển như là 1 từ thông thường, thì biểu diễn của “matter” sẽ chính là biểu diễn của từ <mat>. Điều này vừa giúp có thể duy trì nghĩa của các từ ngắn đồng thời có thể tận dụng để biểu diễn cho các từ dài hơn. Do đó, điều này cũng cho phép ta nắm bắt được ý nghĩa của các yếu tố tiền tố hoặc hậu tố của các từ.

Độ dài của n-gram ta sử dụng có thể được kiểm soát bằng các cờ -minn và -maxn cho số lượng ký tự tối thiểu và tối đa cho 1 cụm n-gram. Chúng kiểm soát phạm vi của các giá trị để có sinh các cụm n-gram phù hợp. Mô hình vẫn được coi là mô hình túi của các từ vì ngoài việc quét khung ngữ cảnh của các từ thì khung ngữ cảnh sẽ được áp dụng riêng cho từng cụm được phân ra từ đang xét, thứ tự của các cụm này không quan trọng. Ta có thể tắt hoàn toàn chế độ n-gram này bằng cách đặt cả hai biến -minn và -maxn thành 0. Điều này có thể hữu ích khi các từ trong mô hình của ta không phải là các từ dùng để biểu diễn cho các ngôn ngữ đặc thù và trong các trường hợp này việc n-gram các cụm ký tự sẽ không có nghĩa. Trường hợp sử dụng phổ biến nhất là khi bạn sử dụng các id của các từ để làm đại diện thay cho từ.

4.2 Yêu cầu về hệ thống để chạy fastText

fastText xây dựng trên các bản phân phối Mac OS và Linux hiện đại. Vì nó sử dụng các tính năng của C++ 11, nó yêu cầu một trình biên dịch có hỗ trợ C++ 11 tốt. Bao gồm:

* Gcc-4.6.3 hoặc các phiên bản mới hơn.
* Hoặc clang-3.3 hoặc các phiên bản mới hơn.

4.3 Cài đặt fastText để sử dụng trên python

Sau khi ta đã tải fastText về máy như hướng dẫn ở mục 4.2 thì ta tiếp tục sẽ phải chạy một số lệnh để có thể sử dụng fastText trên python.

Ta cài đặt fastText cho python bằng lệnh:



1. Cài đặt fastText cho python

Như vậy là python đã có thể sử dụng được thư viện fastText.

4.4 Xử lý biểu diễn từ bằng fastText

Đầu tiên ta chuẩn bị thư viện:



1. Chuẩn bị thư viện fasttext

Xây dựng và huấn luyện mô hình:



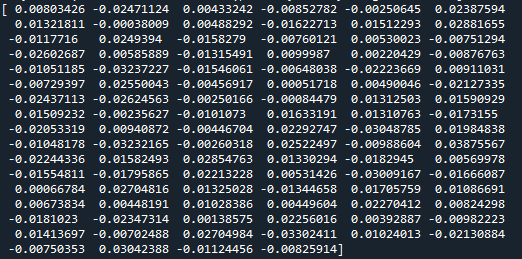
1. Xây dựng và huấn luyện mô hình biểu diễn từ

Ta thử biểu diễn 1 từ bằng fasttext:



1. Biểu diễn từ bằng fasttext

Kết quả biểu diễn của từ minh họa:



1. Kết quả của biểu diễn từ

4.5 Ứng dụng

TÀI LIỆU THAM KHẢO

**Tiếng Anh:**

1. <https://en.wikipedia.org/wiki/QR_code>.