TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**THỊ GIÁC MÁY TÍNH**

**FACE RECOGNITION ALGORITHM BASED ON VGG NETWORK**

*Người hướng dẫn*: **Thầy Phạm Văn Huy**

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

**Đào Việt Duy (MSSV:)**

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**



**THỊ GIÁC MÁY TÍNH**

**FACE RECOGNITION ALGORITHM BASED ON VGG NETWORK**

*Người hướng dẫn*: **Thầy Phạm Văn Huy**

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

**Đào Việt Duy (MSSV:)**

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 thầy Phạm Văn Huy. 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 13 tháng 10 năm 2020*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Duy Hàn Lâm*

*Đào Việt Duy*

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

Là một công nghệ nhận dạng sinh trắc học, nhận dạng khuôn mặt là một trong những chủ đề nóng trong các lĩnh vực nhận dạng mẫu, xử lý hình ảnh, thị giác máy, mạng thần kinh và khoa học nhận thức. Đồng thời, hiện nay việc nhận dạng khuôn mặt là công nghệ sinh trắc học có độ ổn định cao, độ chính xác cao và dễ dàng được chấp nhận và có triển vọng ứng dụng rộng rãi trong các lĩnh vực xác thực danh tính, giám sát bảo mật và tương tác giữa người và máy tính.

Nhận dạng khuôn mặt sử dụng các thuật toán nhận dạng tương quan để nhận dạng hoặc phân biệt khuôn mặt, dựa trên các đặc điểm hình ảnh khuôn mặt được trích xuất. Chọn phương pháp trích xuất đặc điểm khuôn mặt thích hợp và chiến lược đối sánh là chìa khóa của quá trình này. Việc xây dựng khung nhận dạng đi đôi với cách thức của tính năng trích xuất khuôn mặt.

Để thực việc trích xuất khuôn mặt ta có rất nhiều giải thuật, trong đó bài viết này xin giới thiệu về mô hình VGG16. Mô hình sẽ sử dụng các mạng neural CNN để trích xuất thông tin từ gương mặt.

Để giảm độ phức tạp của bài toán, tài liệu sẽ sử dụng giải thuật Principal Component Analysis (PCA) để giảm chiều của dữ liệu.

Và cuối cùng sẽ thực hiện việc phân loại bằng giải thuật SVM.

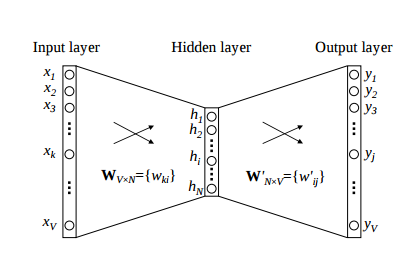
Chương 1: CONVOLUTIONAL NEURAL NETWORK

Mô hình VGG16 mà tài liệu sẽ giới thiệu ở các chương sau sẽ sử dụng nhiều lớp của mô hình Convolutional Neural Network (CNN). Chính vì thế ta cần nên tìm hiểu thế nào là 1 mô hình CNN cơ bản.

1.1 Giói thiệu mô hình

Mô hình CBOW thuộc prediction-based embedding, Mục tiêu của mô hình là xây dựng cố gắng dự đoán từ một từ được chọn dựa theo các từ xung quanh, ngữ cảnh của chúng.

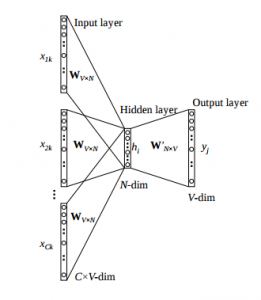
CBOW hoạt động dựa trên cách thức là nó sẽ dự đoán xác suất của một từ được đưa ra theo ngữ cảnh, ngữ cảnh đó có thể gồm một hay nhiều từ quanh từ được chọn, với input là một hoặc nhiều One-hot vector của các từ ngữ cảnh có chiều dài V (với V là độ lớn của từ điển), output sẽ là một vector xác suất cũng với chiều dài V của từ liên quan hoặc còn thiếu, Hidden Layer có chiều dài N, N cũng chính là độ lớn của vector từ biểu thị. Dưới đây là mô hình CBOW với ngữ cảnh là 1 từ đơn:



Ma trận số hóa các từ sẽ được đưa vào mô hình mạng neural 3 lớp với lớp output đầu ra là lớp softmax được để tổng các xác suất đầu ra tầng ẩn thành 1.

Mô hình thuật toán có thể được miêu tả như sau:

* Input được vector hóa one-hot encoded
* Giữa các lớp chỉ sử dụng hàm kích hoạt tuyến tính.
* Input được nhân trọng số tới hàm ẩn điều chỉnh trong số tính toán và từ tầng ẩn tiếp tục nhân trọng số rồi đưa vào hàm softmax đưa ra output là 1 vector xác suất, output này sẽ được so sánh với output mong muốn và tính toán độ lỗi
* Dựa vào độ lỗi này mà mạng neuron sẽ lan truyền ngược trở lại để cập nhật các giá trị của các ma trận trọng số.
* Trọng số giữa lớp ẩn và lớp đầu ra được lấy làm đại diện vectơ từ của từ.



1.2 Ưu điểm của CBOW

So với các phương pháp xác định xác suất ngữ cảnh của từ, CBOW cho ra kết quả tính toán rất tốt.

Không tốn nhiều bộ nhớ so với phương pháp ma trận ngữ cảnh.

1.3 Nhược điểm của CBOW

Xác suất ngữ cảnh một từ CBOW cho ra có thể là trung bình của 2 kết quả. Ví dụ Apple có thể là trái táo vừa là tên công ty, CBOW sẽ đưa ra xác suất ngữ cảnh nằm giữa táo và công ty. Dẫn tới việc các từ giống nhau nhưng nghĩa khác nhau vẫn chỉ được biểu diễn bằng 1 vec-tơ từ duy nhất.

Train mô hình CBOW từ đầu rất khó cho ra kết quả tốt nếu không được tối ưu hóa đúng cách.

CHƯƠNG 2: MÔ HÌNH VGG

Ở chương trước ta đã tìm hiểu xong thế nào là 1 mô hình CNN cơ bản, chương này ta sẽ vận dụng mô hình CNN để đi xây dựng mô hình VGG.

2.1 Giới thiệu sơ lược về mô hình

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.

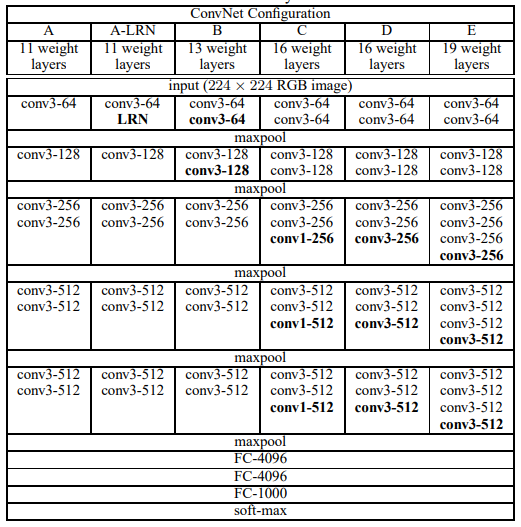
Tất cả các lớp ẩn đều được xử lý phi tuyến tính (ReLU (Krizhevsky et al., 2012))) trước khi xuất kết quả ở đầu ra.

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.

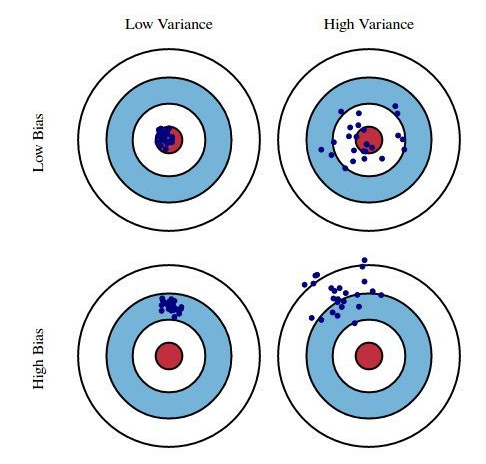
Trong chương này ta sẽ dùng kết quả của mô hình VGG ở chương trước để tiến hành quá trình phân loại và dán nhãn cho bức ảnh.

* 1. Bias và variance

Trong tính toán ta có khái niệm bias và variance.

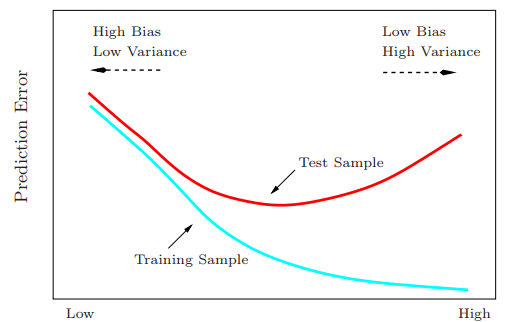
Bias là giá trị biểu thị độ chênh lệch giữa giá trị mong đợi với giá trị dự đoán. Gọi T là giá trị thống kê của tham số . Ta có , trong đó biểu thị cho giá trị mong đợi của T, biểu thị cho bias của .

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:

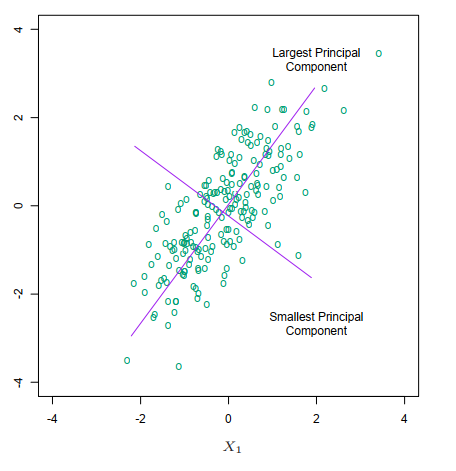
* U và V là 2 ma trận trực giao (orthogonal matrix).
* D là ma trận đường chéo có kích thước là . Với các giá trị đường chéo được gọi là các giá trị dị biến của X. Nếu có một giá trị trong D, thì X sẽ được coi là dị biến (singular).

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:



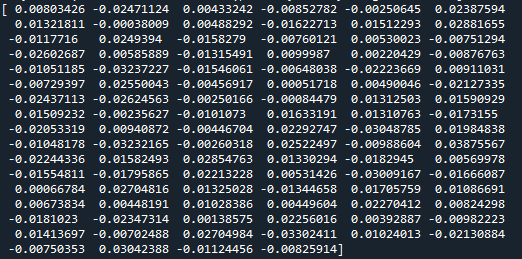
1. Xây dựng và huấn luyện mô hình biểu diễn từ

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. Karen Simonyan, Andrew Zisserman, 2015, Very Deep Convolutional Networks For Large-Scale Image Recognition.
2. Trevor Hastie, Robert Tibshirani, Jerome Friedman, 2008, The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Second Edition).
3. [Hongling](https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling#Derivation-of-Cost-Function) Chen, Chen Haoyu, 2019, Face Recognition Algorithm Based on VGG Network Model and SVM.