ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC - KỸ THUẬT MÁY TÍNH

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

Nhận dạng biểu thức toán học viết tay

Giảng viên hướng dẫn : **TS. Lê Thành Sách**

Nhóm sinh viên thực hiện : **Phan Tấn Phúc - 51303058**

Bùi Khánh Ngọc - 51302567

Tp. Hồ Chí Minh, Tháng 05/2017

Lời cam đoan

Luận văn của nhóm có tham khảo các tài liệu từ nhiều nguồn khác nhau và các nguồn tham khảo này đều được trích dẫn rõ ràng trong phần tài liệu tham khảo. Ngoài những phần được trích dẫn, nhóm xin cam đoan toàn bộ nội dung báo cáo là do các thành viên nhóm tự soạn thảo dựa trên những tìm hiểu và kết quả thực tế do nhóm tạo ra.
Thành viên nhóm sẽ hoàn toàn chịu xử lý theo quy định nếu có bất kỳ sai phạm nào xảy ra liên quan đến những gì nhóm đã cam đoan.

Nhóm sinh viên thực hiện
Phan Tấn Phúc
Bùi Khánh Ngọc

Lời nói đầu

Luận văn tốt nghiệp là thử thách cuối cùng của chặng đường 4.5 năm trên giảng đường đại học của những sinh viên Bách Khoa. Nó khép lại một giai đoạn mà ở đó chứa đựng đầy những nỗ lực, những phấn đấu, cả niềm vui và thỉnh thoảng là những nỗi buồn, sự luyến tiếc. Nhưng cũng từ đây một cánh cửa mới sẽ mở ra- cánh cửa của lao động, của cống hiến.

Để đi đến thời điểm này, nhóm muốn gửi lời cảm ơn chân thành đến Ban giám hiệu và các thầy cô Trường Đại học Bách Khoa Thành phố Hồ Chí Minh, cách riêng cho các thầy cô Khoa Khoa học và Kỹ thuật Máy tính đã chỉ dẫn cho các thành viên nhóm trong suốt những năm học vừa qua.

Trong quá trình thực hiện luận văn, nhóm đã nhận được rất nhiều sự hỗ trợ về mặt chuyên môn cũng như những đóng góp trong vấn đề xây dựng tập dữ liệu cho đề tài từ các anh và các bạn trong và ngoài khoa Khoa học và Kỹ thuật Máy tính. Chính sự giúp sức này là động lực và cũng là sức mạnh giúp nhóm vượt qua những khó khăn trong quá trình thực hiện và đến thời điểm này có thể nói đã hoàn thành thành công. Nhóm xin bày tỏ sự cảm kích và biết ơn chân thành đến những con người này, cách riêng cho Thạc sĩ Huỳnh Chí Kiên- người anh đã luôn hỗ trợ và đưa ra những lời khuyên hữu ích bất kỳ khi nào nhóm gặp khó khăn.

Trên tất cả, lời cảm ơn chân thành nhất và sâu sắc nhất xin được gửi đến thầy hướng dẫn đề tài- Tiến sĩ Lê Thành Sách. Cảm ơn thầy đã luôn theo sát, hỗ trợ cũng như định hướng công việc cho nhóm. Cảm ơn thầy đã luôn tạo ra áp lực để thúc đẩy sinh viên của mình tiến về phía trước và cũng cảm ơn thầy vì những khi áp lực nhất đều được thầy chia sẻ.

Sau cùng, vì những hạn chế về mặt thời gian cũng như khả năng trong cách trình bày và viết báo cáo nên không thể tránh khỏi những thiếu sót, rất mong nhận được sự thông cảm và những ý kiến đóng góp từ quý thầy cô và các bạn để giúp nhóm hoàn thiện hơn.

Chân thành cảm.

Hồ Chí Minh, ngày 08 tháng 12 năm 2017
Nhóm sinh viên thực hiện
Phan Tấn Phúc
Bùi Khánh Ngọc

Tóm tắt luận văn

Luận văn này chỉ ra quá trình thực hiện của nhóm từ tìm hiểu những kiến thức cần thiết đến hiện thực đề tài và đánh giá kết quả đạt được. Bố cục của luận văn gồm 6 chương, không kể các mục lục, phụ lục khác.

Chương 1 là chương giới thiệu tổng quan đề tài. Ở đó sẽ trình bày thế nào là nhận dạng biểu thức toán học, nhu cầu của nó trong xã hội cũng như đưa ra lý do vì sao nhóm chọn đề tài này và sau cùng là quá trình thực hiện của nhóm từ sau giai đoạn thực tập tốt nghiệp.

Chương 2 sẽ đề cập những kiến thức liên quan trực tiếp đến đề tài mà nhóm đã tìm hiểu, cụ thể là kiến trúc mạng SSD.

Chương 3 tóm lược một số công trình liên quan trực tiếp đến đề tài. Trong mục này, nhóm sẽ giới thiệu 4 công trình là cơ sở tham khảo cho quá trình hoàn thiện luận văn.

Chương 4 trình bày về phương pháp đã được sử dụng để hiện thực đề tài luận văn. Phương pháp này được xây dựng dựa trên những kiến thức đã tìm hiểu được trình bày ở chương 2 và những công trình được giới thiệu ở chương 3.

Chương 5 chỉ ra quá trình hiện thực đề tài bao gồm chuẩn bị tập dữ liệu, xây dựng hệ thống và chương trình thử nghiệm. Cụ thể, nhóm sẽ diễn giải về cách cấu hình, các thông số trong từng giai đoạn của quá trình nhận dạng. Tiếp theo, nhóm sẽ giới thiệu cách thức thu thập dữ liệu cùng một số mô tả cho tập dữ liệu hiện tại mà nhóm đang sử dụng. Cuối cùng là kết quả từ chương trình thử nghiệm và những so sánh, đánh giá giữa các mô hình được tạo ra từ việc thay đổi một số yếu tố liên quan đến quá trình nhận dạng.

Chương 6 đánh giá tổng kết luận văn. Nhóm sẽ nêu lên những điều còn tồn đọng và những điểm nổi bật, cũng như hướng phát triển tiếp theo của đề tài. Ngoài ra, sẽ là một vài dòng đúc kết lại chặng đường thực hiện luận văn tốt nghiệp của các thành viên nhóm: những điều đã học được, cảm xúc khi hoàn thành đề tài luận văn.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh mục từ viết tắt

Thuật ngữ	Giải thích
CNN	Mạng nơron tích chập (Convolutional Neural Network)
NN	Mạng nơron truyền thống
MSE	Mean Square Error

Chương 1 Giới thiệu

1 Giới thiệu đề tài

Khoa học công nghệ phát triển đồng nghĩa với việc xã hội "trở nên tự động hoá" bởi lẽ con người ngày càng có nhu cầu tìm kiếm sự hỗ trợ từ các thiết bị công nghệ và hạn chế dùng sức lực của chính mình. Dựa trên nền tảng công nghệ và nhu cầu của xã hội, hàng loạt những thiết bị, ứng dụng công nghệ ra đời. Chúng phục vụ nhu cầu của con người trong đời sống hằng ngày cũng như trong hầu hết mọi lĩnh vực của xã hội từ giao thông, y tế, thương mại,... đến giáo dục. Trong lĩnh vực y tế, phải kể đến ứng dụng giám sát sức khoẻ người dùng trên những chiếc đồng hồ thông minh của Apple hay SamSung. Với giao thông, những ứng dụng chỉ đường, định vị và giám sát xe đề phòng trộm ngày càng phổ biến và giúp ích thực sự cho con người. Riêng với giáo dục, vấn đề thường gặp đối với các bạn học sinh là giải, trực quan hoá các phương trình, hàm số toán học hay soạn các giáo án chứa nhiều công thức, ký hiệu phức tạp đối với thầy cô. Họ cần một giải pháp nào đó giúp giảm thiểu công sức trong những tình huống như vậy. Giải pháp này cần phải giải quyết những vấn đề cơ bản sau:

- Số hoá các công thức in trong sách hay được viết tay.
- Giải tham khảo một số dạng phương trình.
- Trực quan hoá các hàm số
- Biểu diễn công thức, ký hiệu dưới những lệnh mà các trình soạn thảo toán có thể hiểu được, ví dụ Latex.
- ...

2 Lý do chọn đề tài

Bởi sự cần thiết về một ứng dụng nhận dạng biểu thức toán học đã được trình bày ở mục 1, trên thị trường cũng đã xuất hiện nhiều sản phẩm như vậy, đáng chú ý là PhotoMath[1]. Tuy nhiên, nhóm nhận thấy thách thức nếu phải hiện thực thành công đề tài này. Một số câu hỏi đã được đặt ra:

- Làm sao có thể nhận dạng được các ký hiệu?
- Làm cách nào để nhận dạng cả một biểu thức?
- Làm sao biết được đây là loại biểu thức gì?
- Liệu có chắc chắn bất kỳ những gì mình viết ra đều được hiểu đúng?

[1]Một ứng dụng về nhận dạng và giải các biểu thức toán học nổi bật trên Google Play.

- Nhóm có thể tạo ra được một sản phẩm hoàn thiện như PhotoMath không?

Để tự mình trả lời những câu hỏi đó, nhóm quyết tâm thực hiện đề tài này. Ngoài ra, việc áp dụng kiến thức đã học để tạo ra một sản phẩm vừa cần thiết cho xã hội vừa tự bản thân mình có thể sử dụng được tạo cho nhóm một động lực để tiến hành.

3 Phạm vi đề tài

- Nhận dạng biểu thức toán học viết tay dạng offline[2].
- Chuyển biểu thức từ dạng hình ảnh sang dạng máy có thể hiểu được (Latex).
- Ảnh nhận diện ít nhiễu và không bị xoay quá nhiều.
- Nhận diện được các cấu trúc đơn giản (tích phân, tổng, chỉ số trên, chỉ số dưới, phân số, giới hạn, ...)

4 Quá trình thực hiện

Bước 1: Tìm hiểu những công trình trong nước và nước ngoài liên quan đến vấn đề nhận dạng biểu thức toán học.

- Mục tiêu:
  - Biết được khả năng giải quyết vấn đề của các hệ thống nhận dạng biểu thức toán học đã được xây dựng.
  - Hiểu được phương pháp mà các nhóm tác giả sử dụng để thực hiện công trình của họ.
- Công việc:
  - Tìm kiếm và chọn lọc những bài báo với nội dung nhận dạng biểu thức toán học.
  - Đọc và nghiên cứu các thông tin được cung cấp bởi các bài báo.
  - Tải mã nguồn (nếu có), chạy thử nghiệm.

Bước 2: Chọn ra một phương pháp trong các phương pháp đã đọc để hiện thực.

- Mục tiêu:
  - Xây dựng từ đầu một hệ thống nhận dạng biểu thức toán học theo phương pháp đề xuất.

[2]Nhận dạng từ ảnh chứa biểu thức toán học

  - Kiểm chứng độ chính xác được nêu trong bài báo và thực tế nhóm làm được. Trên cơ sở đó, tiến tới việc cải tiến để thu được kết quả tốt hơn.

- Công việc:
  - Đọc kỹ lại bài báo đã chọn.
  - Tìm hiểu những kiến thức được giới thiệu trong bài báo ở mức độ có thể vận dụng được.
  - Triển khai và hiện thực phương pháp.

Bước 2+[3]: Lựa chọn một phương pháp mới để giải quyết đề tài trên cơ sở mã nguồn mở đã có.

- Mục tiêu:
  - Có được phương án thay thế phương án cũ không khả thi
  - Có được chương trình thử nghiệm.
- Công việc:
  - Tìm và tải bộ mã nguồn liên quan đến vấn đề nhận dạng biểu thức toán học[7].
  - Thiết lập và chạy thử mã nguồn.

Bước 3: Đề xuất phương pháp.

- Mục tiêu:
  - Quyết định phương pháp giải quyết vấn đề của nhóm.
- Công việc:
  - Thay đổi, hoàn thiện mã nguồn trên cơ sở phương pháp nhóm đề xuất.

Bước 4: Xây dựng tập dữ liệu

- Mục tiêu:
  - Có được dữ liệu phục vụ quá trình huấn luyện và kiểm thử hệ thống của nhóm.
- Công việc:

[3] 2+ là phương án thay thế của bước 2. Vì trong quá trình thực hiện bước 2, nhóm không thu được kết quả như mong đợi nên dẫn tới việc chọn một cách tiếp cận khác để giải quyết vấn đề.

- Liên hệ nhóm sinh viên Khoá 2011[4] để nhận được toàn bộ tập dữ liệu của họ bao gồm ảnh của các ký tự toán học.
- Chuẩn bị mẫu thu và bổ sung thêm ảnh cho một số loại ký tự mới.
- Xây dựng bộ ảnh chứa đựng các biểu thức toán học từ đơn giản đến phức tạp.

Bước 5: Cải tiến phương pháp

- Mục tiêu:
  - Hoàn thiện hệ thống nhận dạng biểu thức toán học.
- Công việc:
  - Kết hợp kiến thức và kết quả thử nghiệm trên tập dữ liệu để tìm ra phương pháp cải tiến.
  - Xây dựng các phiên bản thử nghiệm khác nhau dựa trên các yếu tố có thể cải tiến.

Bước 6: Xây dựng chương trình demo và đánh giá kết quả

- Mục tiêu:
  - Có kết quả đánh giá và điều chỉnh phương pháp sao cho phù hợp.
  - Có chương trình demo cho luận văn.
- Công việc:
  - Xây dựng bản demo.
  - Xem xét ưu, nhược điểm của các cải tiến dựa trên đánh giá các độ đo.
  - Cấu hình và lựa chọn phương án cải tiến khả thi.

[4]Nhóm sinh viên này đã từng hiện thực đề tài nhận dạng biểu thức toán học

Chương 2 Kiến thức đã tìm hiểu

1 Mạng nơ-ron tích chập (CNN)

2 Mạng Single Shot Multibox Detector (SSD)

SSD[1] như là một mạng cải tiến phương pháp phát hiện bằng cửa sổ trượt[5]. Thay vì sử dụng một (một số) cửa sổ có kích thước cố định, thì SSD[1] sinh ra một số lượng hữu hạn các ô chuẩn [6] để rồi từ các ô chuẩn đó để hệ thống có thể xác định vị trí các ký tự cần nhận diện cho quá trình huấn luyện, qua đó mạng cần phải học cách dự đoán cả kích thước của các ô bọc quanh ký tự thay vì chỉ chấp nhận kích thước cho trước. SSD[1] cũng sử dụng các lớp tích chập (hay cụ thể hơn là các mạng nơ-ron tích chập) để trích đặc trưng, tạo tiền đề cho việc phát hiện và phân loại ký tự.

Nhóm xin phép được tách quá trình huấn luyện thành ba giai đoạn: mã hóa [7], phát hiện - phân loại và tính giá trị lỗi. Và song song với huấn luyện, quá trình kiểm tra, kiểm định, chạy thực tiễn cũng được chia thành ba giai đoạn: trích đặc trưng - phân loại và giải mã [8]

2.1 Bộ mã hóa (Encoder)

Bộ mã hóa chỉ được sử dụng trong quá trình huấn luyện nhằm mã hóa, chuyển đổi từ "đáp án" [9] thô (được lưu trong tệp tin txt hoặc xml) sang "đáp án" mà mạng SSD[1] có thể hiểu được.

Sinh ô chuẩn

Để mã hóa "đáp án", bộ mã hóa trước hết có nhiệm vụ sinh ra nhiều ô chuẩn có nhiều kích thước khác nhau, để làm được điều này, mạng cần phải được cung cấp một số thông số:

- Danh sách kích thước của các feature map (Phần này sẽ được giải thích rõ hơn ở mục sau): Mỗi feature map trong danh sách tương ứng với một mức kích thước[10] trong việc phát hiện ảnh. Trong một mức kích thước, tất cả các ô chuẩn đều có kích thước như nhau và cách đều nhau, mỗi pixel trong feature map thể hiện một (một số ô chuẩn), vì vậy, ta có thể tính được danh sách khoảng cách giữa trọng tâm giữa các ô chuẩn bằng cách lấy kích thước ảnh chia cho kích thước của feature map ở mức kích thước tương ứng. Tất cả các dữ liệu liên quan đến kích thước sau đó sẽ được chuyển về tập giá trị $\{x \in R | x \in [0, 1]\}$

[5] Thuật ngữ tiếng Anh: Sliding Window
[6] Thuật ngữ tiếng Anh: Default box
[7] Thuật ngữ tiếng Anh: Encoder
[8] Thuật ngữ tiếng Anh: Decoder
[9] Thuật ngữ tiếng Anh: Ground truth
[10] Thuật ngữ tiếng Anh: Scale

- Danh sách các tỉ lệ diện mạo[11] ứng với mỗi mức kích thước. Mạng SSD[1] dựa vào các tỉ lệ diện mạo đó để tạo ra các ô chuẩn có hình dạng khác nhau tại cùng một vị trí (trọng tâm của các ô chuẩn có cùng một vị trí).
- Kích thước nhỏ nhất và lớn nhất của các ô chuẩn.

Từ những thông số trên, các ô chuẩn sẽ được sinh ra theo các bước sau:
Trước hết, bộ phận này tiến hành sinh ra kích thước của ổ chuẩn ứng với mỗi mức kích thước tương ứng bằng công thức:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1)$$

Trong đó s_{min} và s_{max} là kích thước nhỏ nhất và lớn nhất của ô chuẩn, m là số lượng feature map. Việc chọn các thông số này khá quan trọng vì nó sẽ ảnh hưởng đến các ký tự có thể nhận diện được sau này, các kích thước cần được chọn sao cho không quá lớn cũng như không quá nhỏ đối với những đối tượng được kỳ vọng nhận diện được.

Ứng với mỗi mức kích thước:

- Mỗi ô chuẩn đều được liên kết với một điểm ảnh trên feature map, ta cũng đã tính được danh sách khoảng cách trọng tâm theo đề cập ở trên. Từ đó ta dễ dàng tính được tọa độ của các trọng tâm của các ô chuẩn trong ảnh theo công thức:

$$(x, y) = \left(\left(i + \frac{1}{2}\right) \times step, \left(j + \frac{1}{2}\right) \times step\right)$$

  Trong đó: x, y là tọa độ của các trọng tâm, i, j là các số nguyên dương nằm trong khoảng từ 0 đến kích thước feature map đang xét và step là khoảng cách giữa hai trọng tâm liền kề trong mức kích thước đang xét.

- Sau khi có được tọa độ của các trọng tâm, bộ phận mã hóa tiến hành sinh các ô chuẩn ứng với mỗi vị trí trọng tâm:
  - Tạo một ô chuẩn có kích thước s_k là kích thước ứng với mức kích thươc k hiện tại.
  - Tạo một ô chuẩn có kích thước bằng $\sqrt{s_k \times s_{k+1}}$.
  - Với mỗi phần tử trong danh sách tỉ lệ diện mạo tương ứng, ta tạo một ô chuẩn có kích thước chiều rộng: $w = s_k \times \sqrt{aspect\_ratio}$ và chiều cao bằng $h = s_k \div \sqrt{aspect\_ratio}$ với $aspect\_ratio$ là tỉ lệ diện mạo đang xét.

Như vậy, với mỗi vị trí được chọn để đặt trong tâm các ô chuẩn, bộ phận mã hóa sẽ sinh ra $2 + n$ ô chuẩn với n là số tỉ lệ diện mạo ứng với mức kích thước hiện tại. Kết quả của quá trình sinh ô chuẩn là một tập hợp nhiều vector. Mỗi vector ứng với một ô chuẩn mang thông tin tọa độ trọng tâm, chiều dài, chiều rộng của ô chuẩn.

[11] Thuật ngữ tiếng Anh: Aspect ratio

Kết hợp[12]

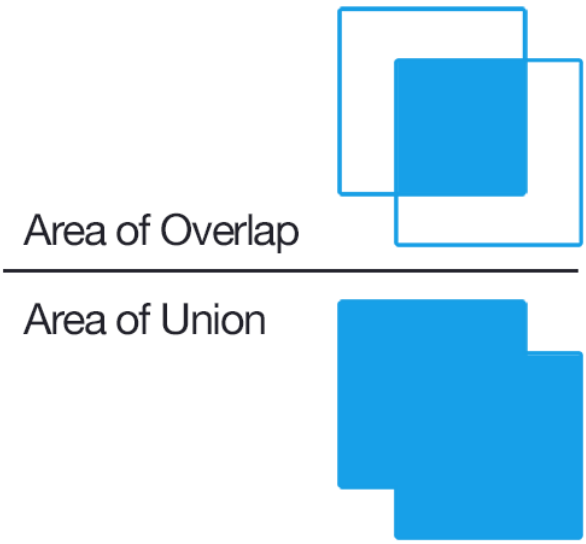
Sau khi hoàn tất sinh ô chuẩn, bộ phận encoder có nhiệm vụ kết hợp dữ liệu thô là các ô bọc [13] từ "đáp án"[14] qua các ô chuẩn vừa được tạo. Kết quả là ta sẽ thu được một tập nhiều vector, mỗi vector đại diện cho mỗi ô chuẩn chứa thông tin gồm tọa độ trọng tâm, kích thước của ô chuẩn và nhãn của ô chuẩn đó là gì (có thể là phần nền - không có gì cả, hoặc một ký tự nào đó cần được nhận diện).

Việc kết hợp được thực hiện bằng cách so sánh từng ô chuẩn với từng ô bọc trong "đáp án" bằng cách tính chỉ số Jaccard, nếu chỉ số này thỏa điều kiện thì ô chuẩn đó được gán nội dung là kí tự chứa trong ô bọc phù hợp nhất, nếu không thì ô chuẩn được gán nội dung là "nền".

Chỉ số Jaccard giữa hai ô bọc được tính bằng công thức:

$$J(A, B) = \frac{A \cap B}{A \cup B}$$

Trong đó, A, B lần lượt là phần ảnh mà ô bọc A và ô bọc B chiếm. Công thức này mang ý nghĩa hai ô bọc có kích thước càng tương đồng và phần trùng nhau càng nhiều thì có chỉ số Jaccard càng gần giá trị 1.



Hình 1: *Mô phỏng chỉ số Jaccard*

Các ô bọc này có bản chất như một "hệ quy chiếu" cho các ô bọc, sau quá trình kết hợp, các ô bọc "đáp án" được tính độ lệch so với các ô chuẩn tương ứng và khi huấn luyện,

[12] Thuật ngữ tiếng Anh: Matching
[13] Thuật ngữ tiếng Anh: Bounding box
[14] Thuật ngữ tiếng Anh: Ground truth

mạng SSD[1] có nhiệm vụ dự đoán các ô bọc theo các độ lệch đó. Giá trị hàm lỗi được tính dựa trên độ lêch của ô bọc "đáp án" và độ lệch của ô bọc dự đoán được. Cách tính độ lệch sẽ được đề cập rõ hơn ở phần tính giá trị lỗi.

2.2 Bộ phát hiện - phân loại

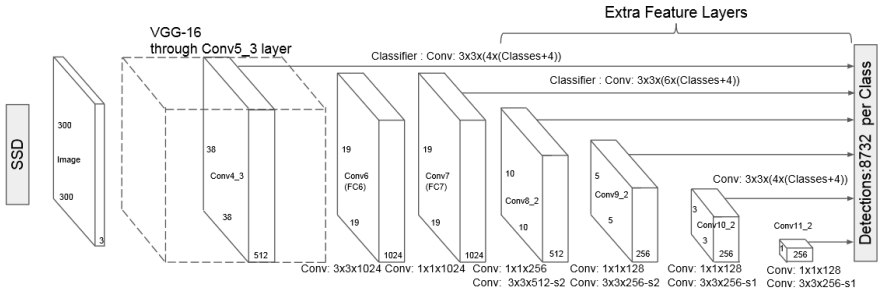
Đây là bộ phận có nhiệm vụ trích đặc trưng từ ảnh thô đầu vào, tạo ra các feature map để đưa vào quá trình phân loại ký tự.

Trích đặc trưng (Hay mạng cơ sở [15])

Mạng cơ sở của SSD[1] là một mạng nơron tích chập bất kỳ, có thể là VGG[8] hoặc lenet[9], ... Hầu hết những mạng này có mục đích phân loại[16], vì vậy ở phía cuối mạng thường có các lớp liên kết đầy đủ nhằm giảm kích thước feature map và tạo ra dữ liệu đầu ra có kích thước phù hợp (bằng với số nhãn cần dự đoán). Khi kết hợp với mạng SSD[1], các lớp liên kết đầy đủ này sẽ được thay thế bằng các lớp tích chập và song hành là các bước trích ra feature map để đưa vào lớp Multibox [10], chi tiết phần này sẽ được để cập ở mục tiếp theo.

Phân loại

Bộ phận phân loại có nhiệm vụ đưa ra dự đoán về vị trí và nhãn của các ký tự có trong ảnh. Để làm được điều đó, bộ phân loại phải sinh ra các feature map ứng với từng mức kích thước khác nhau, các feature map đó được đưa vào lớp Multibox để sinh ra các vị trí dự đoán và nhãn tương ứng. Các feature map được lấy sau một (một số) lớp tích chập. Danh sách feature map có được ở giai đoạn sinh ô chuẩn chính là được lấy từ đây.



Hình 2: *Cấu tạo của một mạng SSD[1]*

Sau khi có được các feature map, lớp Multibox sẽ tiến hành đưa ra dự đoán về ô bọc và nhãn gắn với ô bọc đó. Mỗi feature map sử dụng một tập hợp các lớp tích chập sẽ cho

[15] Thuật ngữ tiếng Anh: Base Network
[16] Thuật ngữ tiếng Anh: Classification

ra một số lượng dự đoán nhất định. Với mỗi vị trí trên feature map được kernel trượt qua, thì một dự đoán về ký tự được sinh ra. Kernel có kích thước:

$$3 \times 3 \times (n \times (classes + 4))$$

Với n là số lượng tỉ lệ diện mạo với fearture map tương ứng và $classes$ là số lượng nhãn cần nhận diện, số 4 trong công thức đại diện cho 4 thông số về vị trí của ô bọc (tọa độ trọng tâm và kích thước). Do kích thước của các kernel không đồng nhất (vì số lượng tỉ lệ diện mạo ứng với từng feature map có thể khác nhau), nên để tạo ra dữ liệu đầu ra phù hợp với các ô chuẩn đã tạo từ trước thì các vector dự đoán sẽ được sắp xếp lại sao cho dữ liệu đầu ra chỉ có $classes + 4$ kênh và vị trí của các vector phải tương ứng với vị trí các ô chuẩn đã sinh từ trước.

2.3 Tính giá trị lỗi

Giá trị lỗi được tính bằng tổng có trọng số giữa lỗi về vị trí[17] và lỗi về độ tin cậy[18]

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

Trong đó:

- x biểu thị các phép kết hợp, cụ thể x_{ij}^p biểu thị phép kết hợp giữa ô chuẩn thứ i với ô bọc thứ j đối với nhãn p. Vì vậy x_{ij}^p có tập giá trị $\{0, 1\}$.
- c biểu thị nhãn kỳ vọng cho phép kết hợp đang xét.
- l biểu thị ô bọc dự đoán.
- g biểu thị ô bọc "đáp án".

Hàm lỗi về vị trí có thể được biểu diễn bằng công thức:

$$L_{loc}(x, l, g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx, cy, w, g\}} x_{ij}^k smooth_{L1}(l_i^m - \hat{g}_j^m)$$

Trong đó:

- N là số lượng phép kết hợp giữa ô chuẩn và ô bọc có ý nghĩa (nhãn của phép kết hợp không phải là "nền". Nếu như $N = 0$ thì ta đặt giá trị lỗi bằng 0.
- Pos là tập hợp các phép kết hợp có ý nghĩa.

[17] Thuật ngữ tiếng Anh: Localization Loss
[18] Thuật ngữ tiếng Anh: Confidence Loss

- $smooth_{L1}$ (theo [11]) là một hàm tính lỗi được định nghĩa là

$$L(x) = \begin{cases} 0.5x^2 & if|x| < 1 \\ |x| - 0.5 & otherwise \end{cases}$$

- $\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w$ Với d biểu thị cho ô chuẩn.
- $\hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$
- $\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right)$
- $\hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$

Bốn công thức $\hat{g}$ trên chính là độ lệch giữa ô đang xét và ô chuẩn, mà chúng tôi đã đề cập ở mục trước.

Vậy ta có thể thấy khi tính giá trại lỗi về vị trí, ta tính dựa trên sai lệch so với ô chuẩn mà ô bọc "đáp án" ban đầu kết hợp được thay vì tính lỗi trực tiếp.

hàm lỗi về độ tin cậy có thể được tính theo công thức:

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} x_{ij}^p \log(\hat{c}_i^p) - \sum_{i \in Neg} \log(\hat{c}_i^0)$$

Với $\hat{c}_i^p = \frac{\exp c_i^p}{\sum_p \exp c)i^p}$

2.4 Bộ giải mã (Decoder)

Bản chất của bộ giải mã khá đơn giản, nó có chức năng chuyển dữ liệu các ô bọc mà mạng dự đoán (tức là những ô bọc được tính kích thước và tọa độ theo ô chuẩn mà ô bọc đó được kết hợp vào) sang dữ liệu tọa độ của ảnh (có gốc tọa độ nằm ở góc trên bên phải và tọa độ nằm trong miền $[0, 1]$. Dữ liệu này cần thiết để trực quan hóa các ký tự mà hệ thống dự đoán và để đưa vào bộ phân tích cú pháp để sinh mã Latex.

2.5 Một số vấn đề khác trong mạng SSD[1]

Tăng cường dữ liệu[19]

Ngay trước quá trình kết hợp, ảnh và các ô bọc được đi qua một bước gọi là cắt ngẫu nhiên[20]. Ở bước này, ảnh được cắt đi một số phần ngẫu nhiên (có tác dụng giống như phóng to (zoom) ảnh. Điều này giúp làm đa dạng dữ liệu huấn luyện và làm giảm bớt hiện tượng overfit.

[19] Data Augumentation
[20] Thuật ngữ tiếng Anh: Random Crop

Cân bằng nhãn[21]

Ngay sau bước kết hợp, hầu hết các ô

[21] Hard Negative Mining

Chương 3 Công trình liên quan

Là một phần quan trọng của hệ thống **Nhận dạng ký tự thuộc thị giác**[22], nhận dạng **biểu thức toán học**[23] đã được nghiên cứu trong hơn nửa thế kỷ qua. Trên hành trình đó, rất nhiều công trình nhằm giải quyết vấn đề này đã được công bố. Trong chương này, nhóm sẽ trình bày tóm lược về một số công trình tiêu biểu, là những điểm tham khảo cho hệ thống nhận dạng biểu thức toán học mà nhóm sẽ hiện thực sau này và nhằm giúp cho bạn có một cái nhìn toàn cảnh về lĩnh vực này.

1 Cái nhìn toàn cảnh

Trong suốt hơn 50 năm hành trình, rất nhiều phương pháp nhận diện biểu thức toán học khác nhau đã được đề xuất, tuy vậy, bài toán này có thể được phân ra thành một số hướng tiếp cận khác nhau:

- Nhận diện theo hướng phân vùng[24]. Ở hướng này, bài toán được chia ra thành nhiều bài toán nhỏ hơn thông qua việc phân chia vùng ảnh biểu thức toán học một cách có quy tắc. Một số ví dụ cho phương pháp này như giải thuật X-Y cut [12] hay dùng phương pháp projection profiles [13]. Đối với phương pháp này, bài toán sẽ trở nên vô cùng khó khăn với những ký tự dính liền nhau hoặc với những bài toán có dấu căn.

- Nhận diện dựa trên cấu trúc cây hoặc đồ thị. Ở hướng này, dựa vào vị trí các ký tự cũng như cấu trúc tổng thể của biểu thức, biểu thức toán học được biểu diễn theo cấu trúc cây hoặc đồ thị. Một số ví dụ cho phương pháp này như Tapia and Rojas Đã đề xuất một phương pháp nhận diện dựa trên cây trải dài[25] và ký tự chủ đạo[26], Zanibbi cho ra đời phương pháp nhận diện bằng một chuỗi các bước biến đổi cây[5]. Phương pháp này khá hoàn thiện nhưng vẫn có một số vướng mắc như việc nhận diện ký tự một cách phi ngữ cảnh vẫn mang đến sự thiếu tự nhiên, hay phương pháp đọc từ trái sang phải vẫn để lại nhiều lỗ hổng trong việc nhận diện biểu thức.

- Nhận diện dựa trên ngữ pháp toán học. Ở hướng này, ngữ pháp được đưa vào quá trình nhận diện, ví dụ như sử dụng ngữ pháp để hậu xử lý, loại bỏ, hiệu chỉnh các ý tự nhận diện sai hoặc sử dụng các mô hình ngữ pháp để dự đoán ký tự tiếp theo trong biểu thức.

[22] Thuật ngữ tiếng Anh: Optical Character Recognition, viết tắt OCR.
[23] Thuật ngữ tiếng Anh: Mathematical Expression, viết tắt ME.
[24] Thuật ngữ tiếng Anh: Segmentation.
[25] Thuật ngữ tiếng Anh: Spanning tree.
[26] Thuật ngữ tiếng Anh: Dominate symbol.

2 WHOOOOOO [2] : Watch, Attend and Parse: An End-to-end Neural Network Based Approach to Handwritten Mathematical Expression Recognition

Khác với những bài báo đi trước sử dụng phương pháp phân vùng hay dựa trên ngữ pháp, bài báo này sử dụng phương pháp mang tên Watch, Attend, Parse dựa trên bài toán chú thích cho ảnh [27]. Hệ thống này là sự kết hợp giữa CNN, RNN và một hệ thống ANN (Cụ thể là GRU[14][28]), CNN đặc trưng cho Watcher sẽ trích đặc trưng ảnh, ANN đặc trưng cho Attend mang nhiệm vụ điều hướng cho mạng biết được vị trí nào sẽ tập trung vào và RNN đặc trưng cho Parser mang nhiệm vụ sinh ra chuỗi Latex chính là đầu ra của hệ thống

Cấu trúc mạng này được chia thành ba phần:

2.1 Watcher

Bộ phận này có bản chất là một hệ thống mạng nơ-ron tích chập đầy đủ (FCN)[29] có cấu tạo là các lớp tích chập và các lớp pooling xếp chồng lên nhau. Bộ phận này nhận đầu vào là ảnh cần nhận diện và đầu ra là các vector đặc trưng ứng với mỗi pixel của ảnh.

2.2 Attend

Bộ phận này hoạt động giống như một hệ thống tổng hợp thông tin, nó có chức năng lấy dữ liệu ký tự vừa được dự đoán từ Parser (sẽ được giải thích bên dưới), từ các vector đặc trưng đã có được từ Watcher và ghi nhận các vị trí đã được xử lý trong ảnh, từ đó dự đoán vị trí tiếp theo để xử lý.

2.3 Parser

Bản chất của bộ phận này là một mạng GRU[14] nhận dữ liệu đầu vào từ hai bộ phận còn lại, từ đó sinh ra chuỗi Latex

3 Wenhao He[3]: Context-aware Recognition

Một quá trình nhận dạng biểu thức toán học chuẩn bao gồm 2 giai đoạn: giai đoạn đầu tiên là phân tách và nhận dạng ký hiệu[30] , giai đoạn thứ hai là phân tích cấu trúc[31]. Nhiều công trình nghiên cứu trước đây về nhận dạng biểu thức toán học cũng thực hiện theo phương pháp như vậy. Nhược điểm của những phương pháp dạng này là thực hiện hai quá

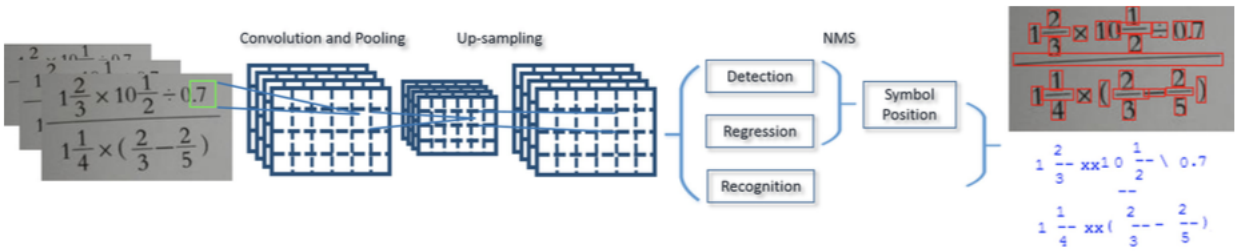
[27] Thuật ngữ tiếng anh: Image Captioning
[28] Viết tắt: Gated recurrent unit
[29] Thuật ngữ tiếng anh: Fully Convolution Network
[30] symbol segmentation and recognition
[31] structure analysis

trình trên một cách độc lập, do đó thông tin cấu trúc[32] của biểu thức không được đưa vào quá trình nhận dạng. Như vậy sẽ dẫn đến lỗi[3].

WenHao He cùng các cộng sự của ông đã đề xuất một phương pháp dựa trên CNN, kết hợp với cách học đa nhiệm vụ[33], cố gắng kết hợp hai giai đoạn chuẩn của quá trình nhận dạng biểu thức toán lại với nhau.

Phương pháp mà các ông đưa ra đảm bảo thông tin cấu trúc của biểu thức được đưa vào quá trình nhận dạng và thể hiện trong các ma trận đặc trưng[34][9].
Đối với các phương pháp trước, biểu thức phải được phân tách thành những ký hiệu rồi từng ký hiệu này mới được đưa qua bộ nhận dạng. Một số phương pháp phân tách ký hiệu thường được sử dụng như: phân tích thành phần liên thông[35], cắt dựa trên các phép chiếu[36][15]. Tuy nhiên với phương pháp mới này, cả ảnh của biểu thức toán học được đưa qua bộ nhận dạng, chính vì vậy mà thông tin cấu trúc của biểu thức được bảo toàn.

Dưới đây là mô hình học của phương pháp này:



Hình 3: *Mô hình học được đề xuất [3].*

Ảnh đầu vào sẽ qua một số lớp tích chập, down-sampling và up-sampling để tạo ra một feature map. Feature map này sẽ được gửi đến ba nhiệm vụ, mỗi nhiệm sẽ tạo ra 1 feature map có cùng kích thước với feature map đầu vào.
Giả sử một điểm i được cho đặt tại toạ độ (w_i, h_i) của feature map được tạo ra bởi các nhiệm vụ.

- **Nhiệm vụ phát hiện** (Dectection task) sẽ cho ra một con số s thể hiện độ tin cậy rằng một ký hiệu được đặt tại i.
- **Nhiệm vụ hồi quy** (Regression task) cho ra một vec-tor 4 chiều x_1, y_1, x_2, y_2 thể hiện thông tin về bounding box của ký hiệu được đặt tại i.
- **Nhiệm vụ nhận dạng** (Recognition task) gán nhãn cho ký hiệu đặt tại i cùng với xác suất của nhãn đó.

[32] structure information hay context information
[33] multi-task learning
[34] feature map
[35] connected components
[36] projection cutting

Như vậy nhiệm vụ phát hiện và hồi quy được thiết kế để định vị trí của ký hiệu trong biểu thức toán học, nhiệm vụ nhận dạng quyết định xem đó là ký hiệu gì.

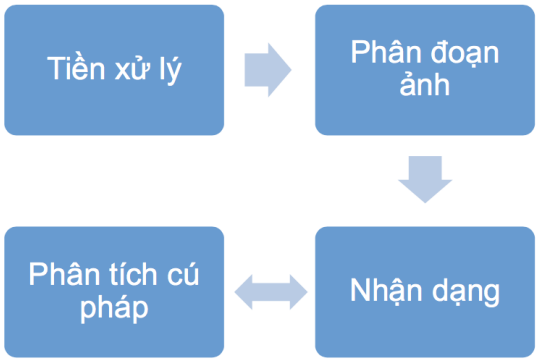
Phương pháp nhận dạng như trên có thể giải quyết cả những trường hợp là thách thức đối với các phương pháp phân tách và nhận dạng ký hiệu trước đây, cụ thể đó là vấn đề gom nhóm ký hiệu đối với các ký hiệu nhiều phần nhỏ, tách những ký hiệu bị dính nhau.

4 QAK[4]

QAK là tên dự án được thực hiện bởi nhóm sinh viên khoá 2011 cũng về đề tài nhận dạng biểu thức toán học.

Phương pháp nhóm sinh viên này xây dựng dựa trên nghiên cứu chính từ hai công trình của Aderson[16] và Zanibbi[5]. Do đó, qua trình nhận dạng biểu thức toán học vẫn đi theo 2 bước chuẩn đó là phân tách ký hiệu và phân tích cấu trúc như đã trình bày ở mục 3.

Dưới đây là mô hình phương pháp mà nhóm đã đề xuất:



Hình 4: *Quy trình nhận dạng.*

- Trong bước tiền xử lý, nhóm áp dụng một số kỹ thuật trong xử lý ảnh như: loại nhiễu, ảnh nhị phân,... để tăng cường chất lượng ảnh, hỗ trợ cho bước phân đoạn.
- Trong bước phân đoạn ảnh, nhóm dùng kỹ thuật chính là cắt theo hình chiếu[15] và phân tích thành phần liên thông[15] cho toán tử lấy căn. Mục tiêu của giai đoạn này là phân tách ảnh chứa biểu thức toán học ban đầu ra thành các mảnh ảnh, mỗi mảnh chỉ chứa 1 ký hiệu toán học. Ngoài ra, ở bước này một cấu trúc cây được xây dựng lưu thông tin của các mảnh ảnh, hỗ trợ cho quá trình phân tích ngữ pháp sau này.
- Ở bước nhận dạng, nhóm sinh viên khoá 2011 đã sử dụng một kiến trúc mạng tương tự Lenet-5[9].
- Với phân tích cú pháp, nhóm sử dụng tập luật **văn phạm phi ngữ cảnh** [37] do chính

[37] Thuật ngữ tiếng Anh: Context-free grammar

nhóm đề xuất để giới hạn kết quả đầu ra của quá trình nhận dạng, từ đó tăng khả năng nhận dạng đúng biểu thức.

Chương 4 Mô hình đề xuất

1 Tổng quan

Nhóm quyết định sử dụng mạng SSD[1]để phát hiện các ký tự trong ảnh và sau đó sử dụng bộ phân tích cú pháp DRACULAE[5] để sinh ra chuỗi Latex.

2 Phát hiện ký tự

2.1 Mạng cơ sở

Về cấu trúc của SSD[1], nhóm đã sử dụng mạng VGG16[6] cho phần mạng cơ sở. Cấu trúc của mạng VGG16[6] gồm tổ hợp các lớp tích chập và lớp pooling xếp chồng lên nhau, ở cuối mạng có các lớp liên kết đầy đủ [38] để giảm kích thước tensor và cuối cùng là xuất ra vector có số chiều phù hợp (có số chiều bằng với số lớp đối tượng cần dự đoán).

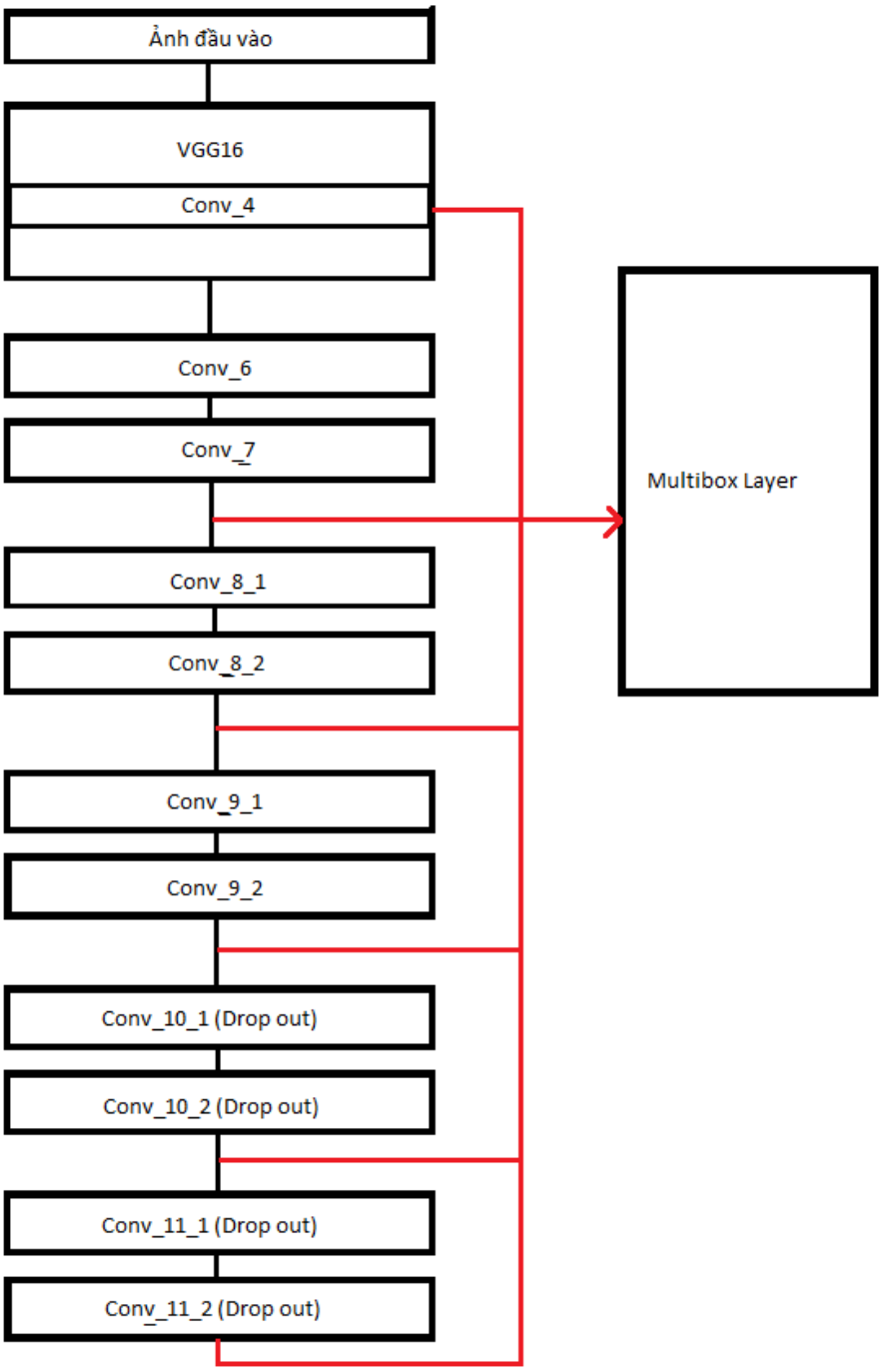
Hình 5: *Cấu tạo mạng VGG16[6]*

Như hình vẽ bên trên, các lớp tích chập và lớp pooling được gom lại thành các cụm, sau hai đến ba lớp tích chập là một lớp pooling. Số kênh của tensor chạy trong mạng được nâng dần từ 64 lên 512 về phía cuối mạng. Khi đưa mạng cơ sở vào mạng SSD[1], các lớp liên kết đầy đủ sẽ được bỏ đi và bộ phận phân loại sẽ được chèn vào vị trí tương ứng. Khi đó, ta cần phải chuyển model của mạng VGG[6] sang mạng SSD[1], phần này sẽ được đề cập chi tiết hơn trong phần hiện thực.

2.2 Thân mạng SSD[1]

Khi tiếp hợp vào mạng SSD[1], các lớp liên kết đầy đủ của VGG16[6] được lược bỏ, thay vào đó là những lớp tích chập, mô hình cụ thể của mạng SSD[1] nhóm để xuất được thể hiện ở ảnh dưới.

[38] Thuật ngữ tiếng Anh: Fully connected layer



Hình 6: *Sơ đồ các lớp trong mạng SSD[1]*

Trong đó Khối VGG16 là phần mạng VGG16[6] đã được lược bỏ các lớp liên kết đầy đủ. Các khối còn lại được thể hiện trong bảng sau:

Khối	Số kênh đầu vào	Số kênh đầu ra	Kích thước nhân	Chèn thêm (Padding)	Bước (Stride)
Conv_6	512	1024	3	1	1
Conv_7	1024	1024	3	6	1
Conv_8_1	1024	256	1	0	1
Conv_8_2	256	512	3	1	2
Conv_9_1	512	128	1	0	1
Conv_9_2	128	256	3	1	2
Conv_10_1	256	128	1	0	1
Conv_10_2	128	256	3	1	2
Conv_11_1	256	128	1	0	1
Conv_11_2	128	256	3	0	1

Bảng 1: *Cấu hình các lớp tích chập trong mạng SSD300[1]*

Riêng khối Conv_4 trong ảnh là lớp tích chập thứ 10 trong mạng VGG[6].

Các lớp tích chập đều sử dụng hàm truyền là hàm ReLu, riêng ở sau các lớp Conv_10_1, Conv_10_2, Conv_11_1, Conv_11_2 thì có thêm một lớp dropout[17] với hệ số dropout là 0.2.

Các feature map được trích ra để thực hiện phát hiện, phân loại từ các mủi tên màu đỏ trên hình, cụ thể:

- Ngay sau khối tích chập - pooling 512 thứ nhất trong mạng VGG[6] (hay sau lớp pooling của lớp tích chập thứ 10 của mạng VGG[6] - hay Conv_4).
- Sau các khối tích chập Conv_7, Conv_8, Conv_9, Conv_10, Conv_11.

Đối với mạng SSD300[1] , kích thước của các feature map ứng với số dự đoán được thể hiện ở bảng sau:

Feature map Sinh bởi	Kích thước	Tỉ lệ diện mạo	Số dự đoán
Conv_4	$512 \times 38 \times 38$	$\{1, \frac{1}{2}, 2\}$	5776
Conv_7	$1024 \times 19 \times 19$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	2166
Conv_8	$512 \times 10 \times 10$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	600
Conv_9	$256 \times 5 \times 5$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	150
Conv_10	$256 \times 3 \times 3$	$\{1, \frac{1}{2}, 2\}$	36
Conv_11	$256 \times 1 \times 1$	$\{1, \frac{1}{2}, 2\}$	4

Bảng 2: *Thông tin của các feature map trích được từ mạng SSD300[1]*

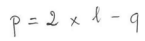
Như vậy, ứng với mỗi ảnh, hệ thống sinh ra 8732 dự đoán. Tương ứng, bộ phận mã hóa của mạng SSD300[1] cũng có những thông số:

- Danh sách kích thước feature map: (38, 19, 10, 5, 3, 1)
- Danh sách kích thước các ổ chuẩn (với $s_{min} = 0.2$ và $s_{max} = 0.9$): (30, 60, 111, 162, 213, 264, 315)
- Ngưỡng Jaccard: 0.5.

2.3 Các thay đổi

Do bản chất mạng SSD300[1] gặp rất nhiều khó khăn trong việc nhận dạng ký tự nhỏ do một số lý do:

- Trong quá trình kết hợp, các ô bọc "đáp án" kết hợp với ô chuẩn phải có chỉ số Jaccard lớn hơn 0.5, nhưng kích thước tối thiểu của ô chuẩn lại là 30×30 pixel, vì vậy nhiều ký tự nhỏ không thể kết hợp được (Trường hợp này xãy ra với hầu hết những ảnh có chữ nhỏ, ví dụ như ảnh dưới)



Hình 7: *Ảnh ví dụ chữ nhỏ (với kích thước thật)*

  Khi không thể kết hợp được thì vùng ký tự đó được hiểu là phần nền, điều này gây bất lợi lớn cho quá trình huấn luyện do vùng ảnh không trống nhưng lúc là nền lúc lại được gắn nhãn.

- Nhiều ký tự có tỉ lệ diện mạo đặc biệt (Ví dụ như ký tự dấu $-$ trong phân số hoặc $\int$) cũng gặp nhiều khó khăn trong việc kết hợp do không đạt được ngưỡng Jaccard.

Vì vậy, nhóm đã sửa một số thông số để tìm cách giải quyết vấn đề trên, trong nhiều bộ thông số thì có 3 mô hình nổi bật.

Giảm kích thước các ô chuẩn

Mô hình này dựa trên mô hình SSD300 nguyên thủy[1], chỉ chỉnh sửa kích thước các ô bọc thông qua chỉnh hai thông số $s_{min} = 0.08$ và $s_{max} = 0.5$, từ đó danh sách kích thước các ô chuẩn được thay đổi thành (9, 24, 54, 84, 114, 144, 174). So với mô hình nguyên thủy, mô hình này không thể kết hợp tốt các ký tự lớn (lớn hơn khoảng 200×200 pixel). Tuy vậy, đối với bài toán về phát hiện các ký tự trong một ảnh, thì kích thước các ký tự không thể quá lớn, nên những ký tự đặc biệt lớn như vậy có thể được xem xét ở độ ưu tiên thấp hơn.

Việc thay đổi tham số ở mô hình này không làm ảnh hưởng lớn đến cấu trúc mạng cũng như quá trình mã hóa, giải mã. Số lượng dự đoán không thay đổi.

Mục đích chính của thay đổi này là để quá trình kết hợp diễn ra tốt hơn khi kích thước của ô chuẩn nhỏ nhất là 9×9, điều này giúp cho những ký tự rất nhỏ vẫn có thể được kết hợp, tránh hiện tượng bỏ sót ký tự như mạng SSD300 nguyên thủy[1].

Giảm kích thước các ô chuẩn và tăng kích thước ảnh đầu vào

Để có được mô hình này, nhóm cần phải có hai thay đổi so với mạng SSD300 nguyên thủy[1]:

- Thay vì sử dụng ảnh 300×300 như nguyên thủy, ảnh sẽ được phóng to lên kích thước 500×500.
- Như ở mô hình đề xuất trước, nhóm thay đổi $s_{min} = 0.08$ và $s_{max} = 0.5$ từ đó thu được danh sách kích thước (15.0, 40, 80, 120, 160., 200, 240, 280).

Những thay đổi trên có làm thay đổi đến cấu trúc mạng SSD[1], khi tăng kích thước ảnh đầu vào nghĩa là kích thước của các feature map thu thập được cũng tăng lên, từ đó kéo số lượng dự đoán trong một ảnh tăng lên như bảng dưới.

Feature map Sinh bởi	Kích thước	Tỉ lệ diện mạo	Số dự đoán
Conv_4	$512 \times 63 \times 63$	$\{1, \frac{1}{2}, 2\}$	15876
Conv_7	$1024 \times 32 \times 32$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	6144
Conv_8	$512 \times 16 \times 16$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	1536
Conv_9	$256 \times 8 \times 8$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	384
Conv_10	$256 \times 4 \times 4$	$\{1, \frac{1}{2}, 2\}$	144
Conv_11	$256 \times 2 \times 2$	$\{1, \frac{1}{2}, 2\}$	64

Bảng 3: *Thông tin của các feature map trích được từ mạng SSD đã giảm kích thước ô chuẩn và tăng kích thước ảnh đầu vào*

Như vậy với mỗi ảnh, hệ thống sinh ra 24148 dự đoán, gấp gần 3 lần so với ảnh SSD300 nguyên thủy[1]. Với những thay đổi này ô bọc thậm chí còn có thể kết hợp với những ô chuẩn nhỏ hơn, bên cạnh đó, mật độ các ô chuẩn cũng dày đặc hơn nhiều so với mạng SSD nguyên thủy[1].

Giảm kích thước các ô chuẩn, tăng kích thước ảnh đầu vào và thêm lớp tích chập

Mô hình này thừa kế các thay đổi ở hai mô hình trước, tuy nhiên ở phần thân mạng SSD[1], nhóm đã gắn thêm 2 lớp tích chập ở cuối với các tham số:

Khối	Số kênh đầu vào	Số kênh đầu ra	Kích thước nhân	Chèn thêm (Padding)	Bước (Stride)
Conv_12_1	256	128	1	0	1
Conv_12_2	128	256	3	0	1

Bảng 4: *Cấu hình các lớp tích chập thêm vào mạng SSD300[1] để được mạng SSD chỉnh sửa*

Hai lớp tích chập này cũng có hàm truyền là hàm relu và đi chung với lớp dropout[17] với hệ số dropout là 0.2.

Thay đổi này cũng làm thay đổi cấu trúc mạng:

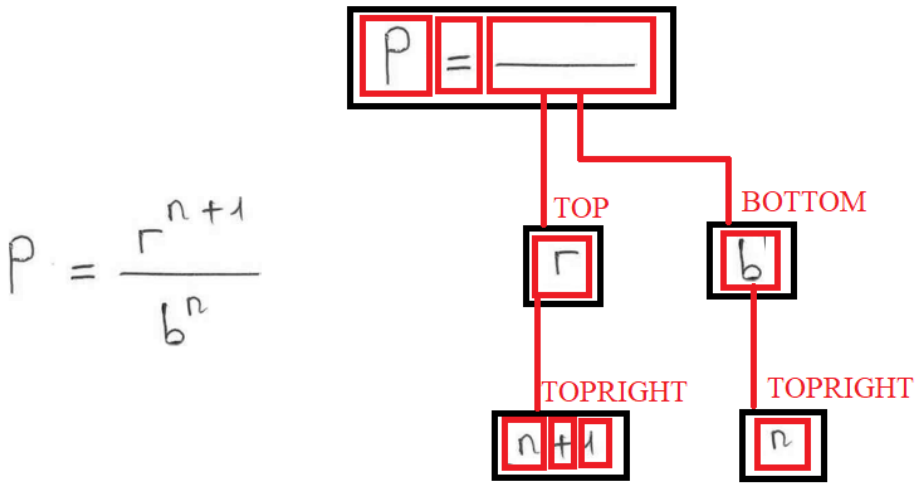
Feature map Sinh bởi	Kích thước	Tỉ lệ diện mạo	Số dự đoán
Conv_4	$512 \times 63 \times 63$	$\{1, \frac{1}{2}, 2\}$	15876
Conv_7	$1024 \times 32 \times 32$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	6144
Conv_8	$512 \times 16 \times 16$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	1536
Conv_9	$256 \times 8 \times 8$	$\{1, \frac{1}{2}, 2, \frac{1}{3}, 3\}$	384
Conv_10	$256 \times 4 \times 4$	$\{1, \frac{1}{2}, 2\}$	160
Conv_11	$256 \times 2 \times 2$	$\{1, \frac{1}{2}, 2\}$	40
Conv_12	$256 \times 2 \times 1$	$\{1, \frac{1}{2}, 2\}$	10

Bảng 5: *Thông tin của các feature map trích được từ mạng SSD[1] đã giảm kích thước ô chuẩn, tăng kích thước ảnh đầu vào và tăng số lớp tích chập*

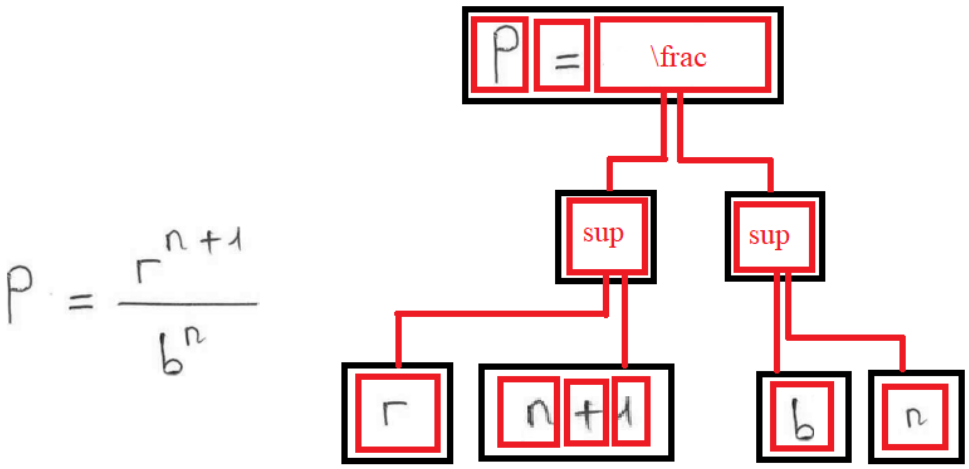
Lớp Conv_12 được thêm vào cuối mạng với mục đích tăng độ dài danh sách feature map đưa vào lớp Multibox, điều này giúp trong quá trình kết hợp, các ký tự lớn hơn có thể được kết hợp.

3 Phân tích cú pháp (Draculae Parser [5])

Bộ phân tích cú pháp Draculae[5] này có nhiệm vụ chuyển dữ liệu dự đoán là các ô bọc sang dữ liệu biểu thức toán học dạng latex. Trong quá trình phân tích cú pháp, biểu thức được lưu trữ dưới dạng cây BST[5][39] là một cấu trúc cây dựa trên các đường cơ sở, các nhánh của cây thể hiện một phân vùng bên trên, bên dưới, ... của nút và là một phân vùng ảnh có đường cơ sở riêng và Lexed - BST[5] là một cấu trúc cây tương tự như BST nhưng thay vì chỉ biểu thị vị trí bên trên, bên dưới thì cây biểu thị quan hệ phân số, chỉ số trên, chỉ số dưới, ...



Hình 8: *Một cây BST[5]*



Hình 9: *Một cây Lexed - BST[5]*

[39] Viết tắt: baseline Syntax Tree: Cây cú pháp dựa trên đường cơ sở

3.1 Sinh cây BST[5]

Đây là giai đoạn phức tạp nhất và cũng là quan trọng nhất, ở giai đoạn này, dữ liệu đầu ra từ mạng SSD[1] là danh sách các ô bọc sẽ được phân tích và xây dựng một cây BST[5]. Công việc này có thể được phân ra thành bốn công đoạn: Tìm ký tự chủ đạo, xác định những ký tự trên đường cơ sở, tái phân vùng và xử lý nút con

Trước khi tiến hành sinh cây, ta cần phải phân loại các ký tự vào những thể loại khác nhau, điều này có mục đích giúp xác định vị trí trọng tâm tốt hơn, xác định được vùng ảnh gắn với chỉ số trên, chỉ số dưới và giúp phân biệt một số loại ký tự có thể có cấu trúc đặc biệt trong biểu thức (ví dụ như ký tự $\sum$ thì có thể có ký tự nằm trên và bên dưới, nhưng ký tự *sin* thì gần như không bao giờ có). Như Zanibbi đã đề xuất cho Draculae[5], các ký tự được phân vào các lớp như bảng dưới:

Lớp ký tự	Tung độ trọng tâm	Bên dưới	Bên trên	Chỉ số dưới	Chỉ số trên
Non-Script Các ký tự phép tính $(+, -, \rightarrow, ...)$	$\frac{1}{2}H$	$\frac{1}{2}H$	$\frac{1}{2}H$	-	-
Open Bracket Ký tự mở ngoặc (	cH	$min(Y)$	$max(Y)$	-	-
Root Ký tự căn $\surd$	cH	$min(Y)$	$max(Y)$	tH	$H - tH$
Variable Range Ký tự có thể viết ở trên hoặc dưới $\sum$, $\int$, lim ...	$\frac{1}{2}H$	tH	$H - tH$	tH	$H - tH$
Ascender Ký tự nổi lên A..Z, 0..9, b,d,f ...	cH	tH	$H - tH$	tH	$H - tH$
Descender Ký tự chìm xuống g, p, y, j, ..., b,d,f ...	$H - cH$	$\frac{1}{2}H + t\frac{1}{2}H$	$H - t\frac{1}{2}H$	$\frac{1}{2}H + t\frac{1}{2}H$	$H - t\frac{1}{2}H$
Centered Ký tự trung tâm o, n, u, },) ...	$\frac{1}{2}H$	tH	$H - tH$	tH	$H - tH$

Bảng 6: *Bảng phân lớp các ký tự cần nhận diện và các ngưỡng xác định phân vùng con*

Trong đó:

- Các cột thứ hai trở đi là tọa độ của một số điểm, ngưỡng đặc biệt tính từ góc trái bên dưới của ký tự. Hệ trục tọa độ có trục tung hướng lên và trục hoành hướng sang

phải. Cột "Tung độ trọng tâm" thể hiện tung độ y của trọng tâm của ký tự, cột bên dưới và bên trên thể hiện ngưỡng trên và ngưỡng dưới của ký tự, các ký tự vượt ngoài ngưỡng đó sẽ được xem là nằm bên trên hoặc nằm bên dưới ký tự đang xét (Ví dụ như một ký tự nào đó có tung độ trọng tâm lớn hơn $max(Y)$ của một ký tự thuộc lớp *Root* thì sẽ được xem là nằm bên trên). Cột chỉ số trên, chỉ số dưới thể hiện ngưỡng để được phân vào vùng chỉ số trên, chỉ số dưới. Riêng hai lớp Non-scipt và Open Bracket có giá trị hai cột này không xác định là do những ký tự của lớp này không có chỉ số trên, dưới.

- H là chiều cao của ký tự.
- c (Viết tăt của centroid ratio): là tham số để xác định trọng tâm cho một số ký tự đặc biệt (ví dụ như các ký tự ngoặc hoặc chìm xuống dưới), tùy người viết chữ có thể có những giá trị phù hợp khác nhau.
- t Là ngưỡng cho chỉ số trên, dưới và chỉ phần bên trên, bên dưới. Tham số này biểu thị sự nhạy cảm đối với những ký tự trên dưới, tham số này càng lớn thì các ký tự khi được phân tích vị trí càng dễ được xếp vào chỉ số trên, chỉ số dưới.

Bảng trên thể hiện quy tắc để kiểm tra hai ký tự có vị trí tương đối như thế nào với nhau, cụ thể:

- B chứa trong A (dành cho trường hợp ký tự căn: $\surd$) khi B có tung độ nằm trong khoảng từ ngưỡng chỉ số dưới đến ngưỡng chỉ số trên của A và hoành độ trọng tâm của B nằm trong khoảng $(Min(X_B), Max(X_B)$.
- A và B liền kề nếu như trọng tâm của B có tung độ nằm trong khoảng từ ngưỡng chỉ số dưới đến ngưỡng chỉ số trên của A và hoành độ trọng tâm của B lớn hơn $Max(X_A)$.
- B là chỉ số trên của A nếu như trọng tâm của B lớn hơn ngưỡng chỉ số trên của A.
- B là ký tự phân số thống trị A khi trọng tâm của A có hoành độ nằm trong khoảng $(Min(X_B), Max(X_B)$

Trước khi tiến hành sinh cây BST, danh sách ô bọc cần phải được sắp xếp theo chiều tăng dần hoành độ của mép phải ô bọc x (hay cụ thể hơn là $min(x)$).

Tìm ký tự chủ đạo

Trong hầu hết trường hợp, ký tự chủ đạo là ký tự nằm bên trái nhất, tuy vậy, ta vẫn cần phải kháng được những trường hợp người dùng viết chữ không chuẩn gây thụt ra ngoài, hoặc những ký tự có dạng như:

$$\sum_{n=100000} a$$

Thì ký tự chủ đạo là $\sum$ nhưng ký tự bên trái nhất lại là n, Draculae[5] được thiết kế để có thể chống lại những trường hợp như vậy.

Để tìm ký tự chủ đạo, ta cần thực hiện những công đoạn sau:

- Trước hết, ta tìm ký tự nằm bên trái nhất.
- Kiểm tra ký tự đó có phải là Variable Range. Nếu đúng thì ta kiểm tra xem ký tự đó có bị thống trị bởi một phân số hay không, nếu có thì ký tự chủ đạo là ký tự phân số, nếu không thì ký tự chủ đạo là ký tự Variable Range vừa tìm được. Nếu ký tự vừa tìm được không phải là Variable Range, thì ta tìm kiếm ký tự Variable Range nằm bên trái nhất và sang bước tiếp theo.
- Nếu không còn ký tự Variable Range nào khác, thì ký tự bên trái nhất vừa tìm được ở bước trên được kiểm tra bị thống trị bởi phân số hay không và trả về ký tự chủ đạo (ký tự đó hoặc ký tự phân số). Nếu còn một ký tự Variable Range khác trong biểu thức, ta tiếp tục với bước kế tiếp.
- Ta kiểm tra xem ký tự Variable Range khác đó có ký tự nào nằm liền kề bên trái không. Nếu không thì tự Variable Range được kiểm tra xem có bị thống trị bởi ký tự phân số hay không và chọn ký tự chủ đạo ký tự phù hợp. Nếu tồn tại ký tự liền kề bên trái, thì ký tự bên trái nhất được kiểm tra có bị thống trị bởi phân số không và chọn ký tự chủ đạo phù hợp.

Xác định ký tự trên đường cơ sở

Sau khi đã có được ký tự chủ đạo, ta cần phải tìm những ký tự liền kề theo chiều ngang với ký tự vừa tìm được, những ký tự này chính là ký tự trên đường cơ sở. Để xác định các ký tự trên đường cơ sở, ta lấy ký tự chủ đạo vừa xác định được để xét:

1. Ta phần vùng các ký tự nằm ở vùng bên trên, bên dưới, góc trên bên trái và góc dưới bên trái (không phân vùng các ký tự nằm bên phải) vào nút con của nút đang xét.
2. Nếu như ký tự đang xét thuộc lớp Non-script, ký tự đó được đánh dấu nằm trên đường chủ đạo, ta quay lại tìm ký tự chủ đạo trong các ký tự còn lại và xác định các ký tự trên đường cơ sở với ký tự chủ đạo đó.
3. Ta kiểm tra danh sách các ký tự còn lại xem có ký tự nào liền kề bên phải với ký tự đang xét hay không, nếu có, ta kiểm tra ký tự đó có bị thống trị bởi ký tự khác hay không và quay lại bước 1 với ký tự được xét là ký tự thống trị nhất (hoặc chính ký tự liền kề nhất vừa tìm được đối với trường hợp nó không bị thống trị bởi ký tự nào).

4. Nếu hệ thống chạy đến bước này thì có nghĩa là trong các ký tự còn lại, không còn ký tự nào liền kề với ký tự đang xét, vì vậy ta tiến hành phân vùng đưa các ký tự còn lại vào nút con góc trên bên phải (chỉ số trên) và góc dưới bên phải (chỉ số dưới) của nút đang xét.

Sau khi thực hiện bước này lần đầu tiên, ta thu được một cây có ba tầng, tầng đầu chứa nút gốc, tầng thứ hai chứa các ký tự nằm trên đường cơ sở, tầng thứ ba chứa các nút con của các nút chứa ký tự nằm trên đường cơ sở, các nút con này có thể chứa một hoặc nhiều ký tự, chưa là một cây.

Tái phân vùng

Ở bước trước, nút con được phân vào vùng góc trên bên trái và góc dưới bên trái, điều này không có ý nghĩa đối với ngữ nghĩa của biểu thức toán học và khó khăn trong việc sinh cây Lexed BST[5] sau này. Vì vậy ở bước này, các nút con sẽ được thay đổi vị trí cho phù hợp với ngữ nghĩa của biểu thức. Cụ thể:

- Các nút con thuộc vùng góc trên bên trái và góc dưới bên trái sẽ được phân vùng vào vùng góc trên bên phải và góc dưới bên phải của ký tự trước đó.
- Riêng đối với các ký tự Variable Range, hệ thống thực hiện thêm bước kiểm tra sự liền kề đối với những ký tự trong vùng con để phân vùng phù hợp.

Xử lý nút con

Tới bước này, ta đã có một cây có ba tầng với tầng thứ hai có phân vùng nút con khá chuẩn xác. Nhiệm vụ bây giờ là phải tiếp tục phân vùng cho các nút con của các ký tự trên đường cơ sở. Việc này được thực hiện bằng cách xem từng nút con là từng vùng ảnh mới riêng biệt và thực hiện lại ba bước trên để thu được các cây con.

Sau quá trình này, ta thu được một cây BST biểu thị vị trí tương đối giữa các nút. Cây này tiếp tục được đi qua [Lexical Pass] để thu được Lexed - BST[5].

3.2 Sinh cây Lexed - BST[5]

Sau khi đã thu được cây BST từ danh sách các ô bọc, nhiệm vụ còn lại là không quá phức tạp. Trước khi tạo được chuỗi Latex, ta cần phải sinh ra được một cây Lexed - BST[5], cây này chứa các thông tin như "nhóm ký tự" và cấu trúc biểu thức:

- Nhóm ký tự: Ban đầu, các ký tự được gom thành từng nhóm có nghĩa (Ví dụ các ký tự "s", "i", "n" thì sẽ được gộp lại thành một ký tự "sin" duy nhất.
- Ta sử dụng các luật khác nhau để sinh ra cây Lexed - BST[5].

Luật sinh

Để chuyển từ cây BST[5] sang cây lexed - BST[5], nhóm có đề xuất một số luật sinh:
Gọi ký tự đang xét là A

- Nếu cây BST B nằm trong nút con góc phải bên dưới của A thì sinh ra một nút sub(A, B)
- Nếu cây BST B nằm trong nút con góc phải bên trên của A thì sinh ra một nút sup(A, B)
- Nếu cây BST B nằm trong nút con bên trong của A và A là ký tự căn ($\sqrt{}$) thì sinh ra một nút sqrt(B)
- Nếu A có nút con bên trên hoặc bên dưới (hoặc cả hai), thì tùy thuộc vào ký tự A mà ta sinh ra các nút khác nhau (Ví dụ như các ký tự $\sum$, $\lim$, $\prod$, $\int$, $\rightarrow$ và quan trọng nhất là dấu gạch ngang trong phân số)

Sau khi sinh được cây Lexed - BST[5], hệ thống sẽ duyệt qua cây theo chiều tiền thứ tự để sinh mã Latex

Chương 5 Hiện thực, đánh giá

Để xây dựng hệ thống nhận diện biểu thức toán học viết tay, nhóm cần hoàn thành ba công việc chính bao gồm thu thập dữ liệu, huấn luyện mạng SSD và xây dựng chương trình sinh mã latex.

1 Chuẩn bị dữ liệu

2 Hiện thực hệ thống

2.1 Quá trình tiền huấn luyện

Như đã trình bày ở trên, mạng SSD[1] mà nhóm lựa chọn có sử dụng mạng cơ sở là VGG16[6], đây là một mạng phân lớp[40]. Để đạt được kết quả huấn luyện cho mạng SSD[1] tốt hơn, tác giả đã khuyến cáo sử dụng phương pháp học chuyển tiếp[41][1]

Học chuyển tiếp

Để thực hiện phương pháp này, ta cần phải chuẩn bị một tập dữ liệu riêng gồm các ảnh kích thước 32×32. Các ảnh này được lấy từ tập huấn luyện của đề tài QAK.

Khi đã thu được một mô hình phù hợp (Huấn luyện tập tiền huấn luyện đã hội tụ), nhiệm vụ kế tiếp là ta phải chuyển các tham số đã học được sang một mô hình của mạng SSD[1].

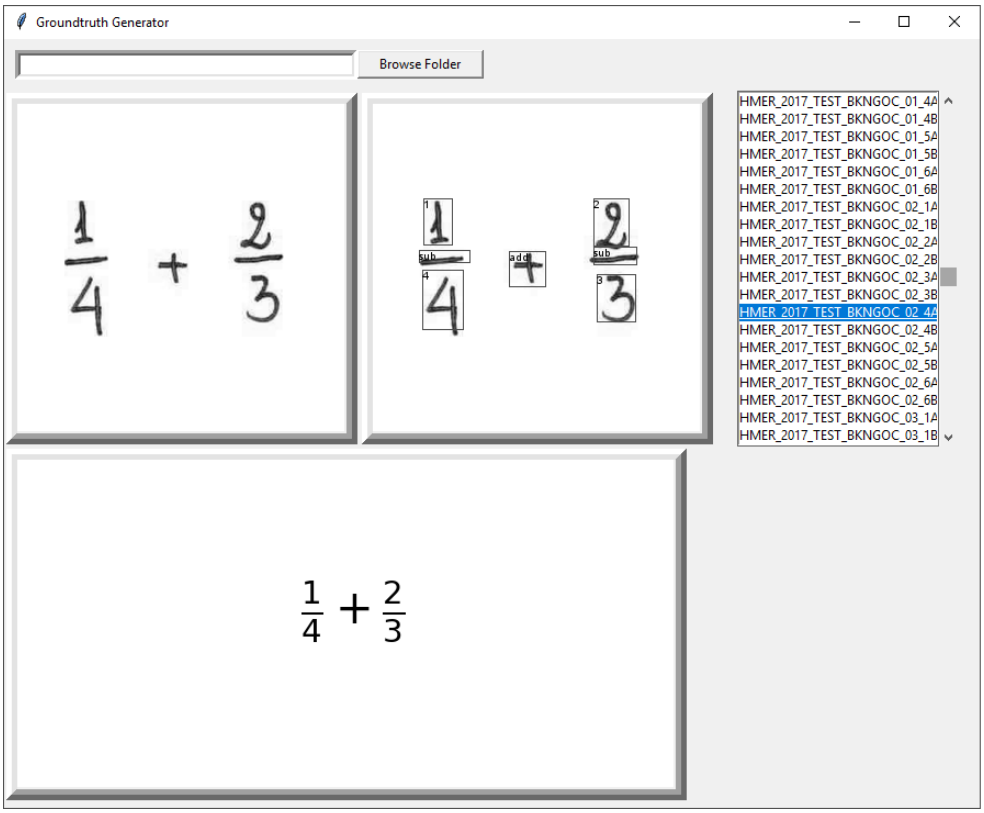
Quá trình này không quá phức tạp, nhóm đã viết một đoạn mã giúp tạo một mô hình mạng SSD rỗng, sau đó gán tham số từ mô hình mạng VGG16[6] vừa thu được sang mô hình mạng SSD mới. Ta bỏ qua các lớp liên kết đầy đủ do phần này đã được lược bỏ, lớp pooling cũng được bỏ qua do lớp này không có tham số.

Sau khi đã thu được mô hình mạng SSD, ta có thể bước tiếp đến công đoạn huấn luyện mạng SSD.

3 Xây dựng bản thử nghiệm

Khi đã thu được mô hình hoàn thiện của SSD, nhóm xây dựng một bản thử nghiệm bằng python. Giao diện khá đơn giản, người dùng chỉ việc chọn thư mục chứa ảnh, sau đó chọn ảnh cần nhận diện. Chương trình sẽ xử lý và đưa ra 3 ảnh gồm ảnh gốc ban đầu, ảnh thể hiện các ô bọc mà mạng dự đoán và ảnh biểu thức toán học được tạo ra từ đoạn mã Latex.

[40] Thuật ngữ tiếng anh: Classification
[41] Thuật ngữ tiếng anh: Transfer Learning



Hình 10: *Giao diện bản thử nghiệm*

4 Đánh giá ưu, nhược điểm

4.1 Ưu điểm

- Hệ thống có thể nhận diện được ký tự ở nhiều kích thước khác nhau.
- Nhận diện được nhiều biểu thức mà nếu chỉ dùng kỹ thuật phân vùng thì khó có thể làm được (ví dụ như biểu thức có ký tự dính nhau).
- Số lượng ký tự nhận diện được khá đa dạng.

4.2 Nhược điểm

- Khi có nhiều ký tự trong ảnh với mật độ lớn thì hệ thống nhận diện gặp nhiều khó khăn.
- Tốc độ xử lý một ảnh còn chậm.
- Hậu xử lý vẫn còn chưa thật sự hiệu quả.
- Các ký tự hoàn toàn trùng lên nhau dễ bị bỏ sót (dấu căn).

Chương 6 Tổng kết

1 Kết luận

2 Hướng phát triển trong tương lai

Hệ thống vẫn còn nhiều điểm có thể cải thiện thêm, trong tương lai, có thể:

- Thêm phần hậu xử lý ở các giai đoạn nhận diện ký tự và sinh cây Lexed - BST.
- Mở rộng tập dữ liệu về cả số lượng ảnh và số lượng nhãn.
- Cải thiện khả năng nhận diện các ký tự có tỉ lệ diện mạo đặc biệt.
- Đưa biểu thức đã nhận dạng được vào ứng dụng (Như có thể vẽ đồ thị, giải phương trình).

Tài liệu

[1] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[2] J. Zhang, J. Du, S. Zhang, D. Liu, Y. Hu, J. Hu, S. Wei, and L. Dai, "Watch, attend and parse: An end-to-end neural network based approach to handwritten mathematical expression recognition," *Pattern Recognition*, 2017.

[3] W. He, Y. Luo, and F. Yin, "Context-aware mathematical expression recognition: An end-to-end framework and a benchmark," pp. 3235–3240, Dec. 2016.

[4] A. N. Quoc and K. N. Anh, *Nhận dạng biểu thức toán học*. 2015.

[5] R. Zanibbi, D. Blostein, and J. Cordy, "Recognizing mathematical expressions using tree transformation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 11, pp. 1455 – 1467, Nov. 2002.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] Kuangliu, "pytorch-ssd." https://github.com/kuangliu/pytorch-ssd/, May. 2017.

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[9] Y. Lecun, L.Boutou, and Y.Bengio, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 88, no. 11, pp. 2278 – 2324, Nov. 1998.

[10] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2147–2154, 2014.

[11] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[12] R. H. J. Ha and I. Phillips, "Understanding mathematical expressions from document images," 1995.

[13] N. Okamoto and M. Bin, "Recognition of mathematical expressions by using the layout structures of symbols," 1991.

[14] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[15] K.-F. Chan and D.-Y. Yeung, "Mathematical expression recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 3, no. 1, pp. 3–15, Aug. 2000.

[16] R. H. Anderson, "Syntax-directed recognition of hand-printed two-dimensional mathematics," *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, pp. 436–459, Aug. 1967.

[17] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *Journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.