TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO CUỐI KÌ

CLASS IMBALANCE PROBLEM AND FOCAL LOSS

SV thực hiện: 17520644 - Phạm Hoàng Đăng Khoa

17520060 – Thái Trung Hiếu

17520618 – Nguyễn Thiệu Khang

Contents

I. Giới thiệu

Trước khi chúng ta tìm hiểu kĩ về Class imbalance và Focal Loss chúng ta sẽ đi sơ qua về Object Detection và ảnh hưởng của 2 thuật ngữ trên vào bài toán Object Detection

Theo thông tin từ nhiều nguồn trên mạng, hiện nay các dự án về Object Detection có 2 hướng tiếp cận đó là :

- One – stage : 1 giai đoạn
- Two – stage : 2 giai đoạn

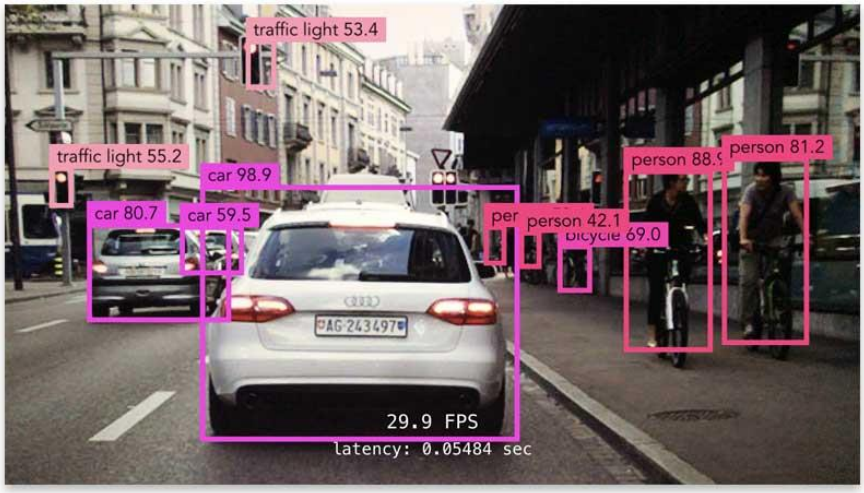
Các công cụ phát hiện đối tượng (Object Detector) cho đến nay có độ chính xác ấn tượng đều dựa trên cách tiếp cận hai giai đoạn R-CNN.

Tới đây chúng ta thường sẽ đặt câu hỏi là : One – stage và Two – stage là gì? và Lí do tại sao các dự án object detection lại có độ chính xác cao lại dựa trên Two-stage mà ko dựa trên One – Stage ?

II. Công việc liên quan

a) Object detection and Two – stage vs One – stage Detectors

Object detection là kỹ thuật thị giác máy tính để tìm kiếm các đối tượng quan tâm trong một hình ảnh:



So với classification, object detection phức tạp hơn. Classification chỉ cho ta biết chủ thể của hình ảnh trong khi object detection có thể tìm thấy nhiều đối tượng, phân loại và xác định vị trí của chúng trong hình ảnh.

Mô hình object detection dự đoán qua những bounding box, mỗi bounding box bao bọc những vật thể nó tìm thấy và cho ra xác xuất phân loại cho những đối tượng đó.

Mô hình object detection thường dự đoán rất nhiều bounding box. Thế nên mỗi hộp sẽ có một điểm cho biết khả năng hộp này có thực sự chứa vật thể hay không. Ở giai đoạn hậu xử lý, ta sẽ lọc ra những hộp có điểm số thấp hơn một mức nhất định (mức này được gọi là Non-maximum Suppression).

Một trong những vấn đề rắc rối sẽ gặp phải là hình ảnh dùng để huấn luyện có thể có từ 0 đến hang chục đối tượng, khi đó mô hình có thể xuất ra nhiều hơn một dự đoán. Vậy làm thế nào để ta biết được nên so sánh dự đoán nào với bounding box nào?

Các mô hình two-stage detector đầu tiên sẽ tạo ra những khu vực đề xuất (proposal area)-là những vùng có khả năng chứa một đối tượng. Sau đó nó đưa ra dự đoán riêng cho từng vùng này. Nhờ đó những mô hình có độ chính xác cao nhưng lại khá chậm vì nó yêu cầu chạy phần phát hiện và phân loại nhiều lần. Các mô hình two-stage có thể kể đến là Faster R-CNN, Mask R-CNN,…

Mặt khác, các mô hình one-stage detector coi việc phát hiện đối tượng như một vấn đề hồi quy. Nó lấy các hình ảnh đầu vào và tìm hiểu những xác xuất của các lớp và tọa của những bounding box khi đi qua mạng neural một lần duy nhất và dự đoán tất cả bounding box trong lần đó. Nhờ đó những mô hình này nhanh hơn nhiều. Một vài ví dụ phổ biến của mô hình one-stage là YOLO, SSD,…

b) Selective Search

Giải thích một số khái niệm: Region proposal (vùng đề xuất) là vùng có khả năng chứa đối tượng hoặc hình ảnh bên trong nó.Selective search tìm kiếm những vùng được xử lý để đưa những vùng này vào vùng đề xuất.Selective search là một công cụ tìm kiếm có chọn lọc,nó còn là một thuật toán đề xuất vùng được sử dụng để phát hiện đối tượng.Nó được thiết kế nhanh chóng với khả năng thu hồi cao.Thuật toán này dựa trên tính toán theo thứ bậc(phân cấp) của các vùng tương tự dựa trên khả năng tương thích về màu sắc,kết cấu,kích thước và hình dạng.Selective Search sử dụng 4 biện pháp để xử lý sự tương đồng 2 khu vực về:Màu sắc ,kết cấu,kích thước và hình dạng.Nhưng trước khi nói đến những phương pháp mà selective search sử dụng để xử lý ảnh thì ta sẽ nói về các bước mà selective search vận hành để đưa những phân đoạn được xử lý vào vùng đề xuất như thế nào.Selective search sử dụng phương pháp phân đoạn dựa trên biểu đồ của Felzenswalb và Huttenlocher để phân đoạn quá mức hình ảnh dựa trên cường độ của các Pixel[4] nói gọn hơn là Selective search sẽ tìm ra được phân đoạn từ nhỏ đến lớn của các khu vực từ đó sẽ tính toán được các phân cấp và sẽ giải quyết được bài toán.

(Pixel là một điểm ảnh. Note:Các ô vuông nhỏ ở vùng phóng to)

Biểu đồ của Felzenswalb và Huttenlocher (sau khi đọc bài báo viết về biểu đồ của Felzenswalb và Huttenlocher thì đã rút ra một kết luận đơn giản nhất về phương pháp của biểu đồ này) đó là Nhóm các Pixel giống nhau về mặt tri giác thành một vùng.Bước 1:Selective search sẽ xử lý từng khu vực bằng cách lấy phân đoạn giám sát của biểu đồ Felzenswalb và Huttenlocher để làm input.Sau khi đã xong bước 1,Selective search thêm các hộp giới hạn tương ứng với các bộ phận được phân đoạn giám sát vào danh sách các đề xuất khu vực đây sẽ là bước 2.Bước 3 nó sẽ nhóm các phân đoạn giám sát liền kề dựa trên sự giống nhau.Và bước cuối cùng sẽ trở lại bước 1 bởi vì các phân đoạn lớn hơn sẽ hình thành khi lặp lại bước 1 và lúc đó ta sẽ thêm các phân đoạn này vào danh sách đề xuất các khu vực.Từ đó danh sách các đề xuất khu vực sẽ xuất hiện các phân đoạn từ nhỏ tới lớn.Việc này sẽ giúp ta dễ dàng tính toán được các phân cấp và sẽ giải quyết được bài toán.Sau khi nói sơ qua về quá trình Selective search vận dụng Felzenswalb và Huttenlocher để phân cấp. chúng ta sẽ nói đến cách xử lý hình ảnh của Selective search. Đầu tiên là xử lý sự tương đồng về màu sắc giữa 2 khu vực:Biểu đồ màu gồm 25 ngăn được tính toán cho mỗi kênh của hình ảnh và biểu đồ cho tất cả các kênh được ghép nối để thu được bộ mô tả màu dẫn đến bộ mô tả màu 25 × 3 = 75 chiều.Dựa vào giao điểm trên biểu đồ ta có công thức sau:

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k)$$

t_i^k là giá trị biểu đồ cho k^{th} bin trong bộ mô tả màu.

Sau đó,Selective search sẽ xử lý sự tương đồng về kết cấu: Các tính năng kết cấu được tính toán bằng cách trích xuất các dẫn xuất Gaussian ở 8 hướng cho mỗi kênh. Đối với mỗi hướng và cho mỗi kênh màu, biểu đồ 10-bin được tính toán tạo thành bộ mô tả đối tượng địa lý 10x8x3 = 240 chiều. Độ giống nhau về kết cấu của hai vùng cũng được tính toán bằng cách sử dụng các giao điểm biểu đồ.

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k)$$

t_i^k là giá trị biểu đồ cho k^{th} bin trong bộ mô tả kết cấu.

Xử lý tương đồng về kích thước khuyến khích các vùng nhỏ hơn hợp nhất sớm. Nó đảm bảo rằng các đề xuất khu vực ở mọi quy mô được hình thành ở tất cả các phần của hình ảnh. Nếu biện pháp tương đồng này không được xem xét, một khu vực duy nhất sẽ tiếp tục nuốt chửng tất cả các khu vực lân cận nhỏ hơn từng khu vực một và do đó các đề xuất khu vực ở nhiều quy mô sẽ chỉ được tạo tại vị trí này. Sự giống nhau về kích thước có công thức là:

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}$$

trong đó $size(im)$là kích thước của hình ảnh tính bằng pixel.

Xử lý tương đồng về hình dạng đo lường mức độ phù hợp của hai vùng r_i và r_j với nhau. Nếu r_i phù hợp với r_j, ta muốn hợp nhất chúng để lấp đầy khoảng trống và nếu chúng thậm chí không chạm vào nhau thì chúng không nên được hợp nhất. Khả năng tương thích hình dạng có công thức là:

$$s_{fill}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

trong đó $size(BBij)$ là một hộp giới hạn xung quanh r_i và r_j

Cuối cùng là sự tương đồng cuối cùng giữa hai khu vực được định nghĩa là sự kết hợp tuyến tính của 4 điểm tương đồng nói trên bằng công thức:

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j)$$

trong đó r_i và $r_j$là hai vùng hoặc phân đoạn trong hình ảnh và $a_i \in 0, 1$ biểu thị liệu phép đo độ tương tự có được sử dụng hay không.

c) Foreground-background class imbalance là gì?

Mất cân bằng (imbalance) là một vấn đề trong object detection xảy ra khi sự phân phối của một thuộc tính làm ảnh hưởng đến hiệu suất. Một trong những vấn đề mất cân bằng phổ biến nhất trong object detection là foreground-to-background imbalance biểu hiện ở sự chênh lệch rất lớn giữa những ví dụ tích cực và tiêu cực trong hình ảnh. Trong một hình ảnh nhất định, mặc dù thường có vài ví dụ tích cực, nhưng ta có thể trích xuất được hang triệu ví dụ tiêu cực. Nếu không giải quyết được vấn đề này, nó làm giảm đáng kể độ chính xác của mô hình.

Ở thời kỳ mà deep-learning phát triển, 8 vấn đề mất cân bằng đã được phát hiện và được nhóm lại với 4 dạng chính: Class imbalance, scale imbalance, spatial imbalance, objective imbalance(bảng 1). Trong class imbalance, ngoài ví dụ điển hình là foreground-background imbalance, còn có sự mất cân bằng giữa những foreground (positive). Trong báo cáo này ta sẽ chỉ tìm hiểu về foreground-background imbalance.

Table 1

Type	Imbalance Problem	Related Input Property
Class	Foreground-Background Class Imbalance (§4.1)	The numbers of input bounding boxes pertaining to different classes
	Foreground-Foreground Class Imbalance (§4.2)	
Scale	Object/box-level Scale Imbalance (§5.1)	The scales of input and ground-truth bounding boxes
	Feature-level Imbalance (§5.2)	Contribution of the feature layer from different abstraction levels of the backbone network (i.e. high and low level)
Spatial	Imbalance in Regression Loss (§6.1)	Contribution of the individual examples to the regression loss
	IoU Distribution Imbalance (§6.2)	IoU distribution of positive input bounding boxes
	Object Location Imbalance (§6.3)	Locations of the objects throughout the image
Objective	Objective Imbalance (§7)	Contribution of different tasks (i.e. classification, regression) to the overall loss

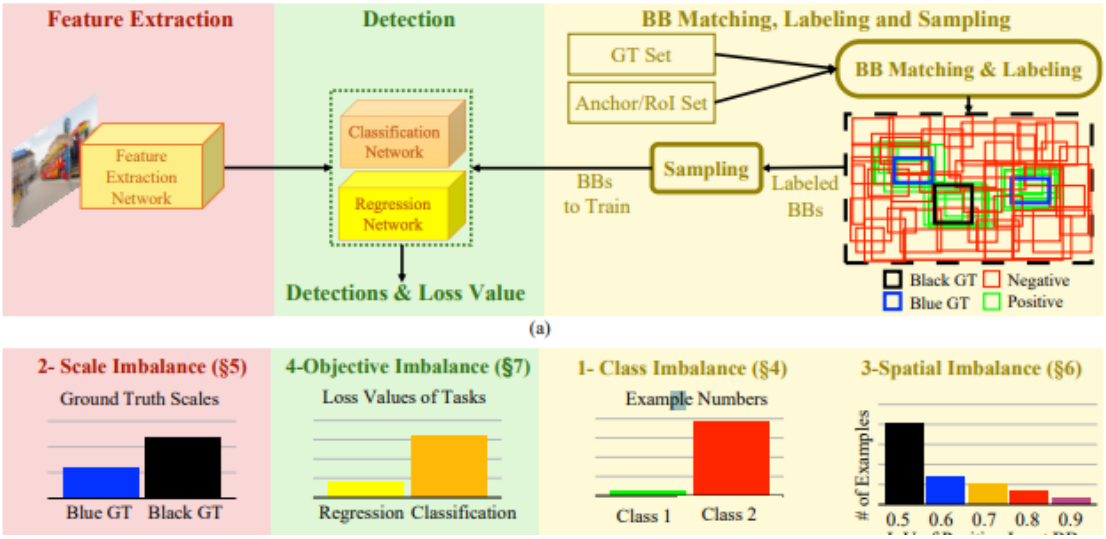


Figure 1

Hình trên mô tả đường đào tạo của một mô hình object detection chung và những ví dụ về mất cân bằng

Như đã đề cập, foreground-background imbalance xảy ra khi số lượng các ví dụ tiêu cực (background) lớn hơn nhiều so với các ví dụ tích cực (foreground). Vấn đề này là không thể tránh khỏi vì hàu hết những bounding box được gán nhãn là background. Vấn đề này xảy ra trong quá trình training và nó không phụ thuộc và số lượng ví dụ cho mỗi lớp trong tập dữ liệu vì chunhs không bào gồm bất kỳ chú thích nào trên background.

Foreground-background imbalance, hay đúng hơn là class imbalance, sẽ được xử lý tại giai đoạn sampling trên đường đào tạo ở Figure 1.

Có 4 nhóm phương pháp để xử lý vấn đề này: Phương pháp lấy mẫu cứng, phương pháp lấy mẫu mềm, phương pháp không lấy mẫu, phương pháp tổng hợp(Table 2).

Ở báo cáo này ta sẽ chỉ tìm hiểu về phương pháp Focal Loss.

Table 2

- **1.Hard Sampling Methods**
  - A.Random Sampling
  - B.Hard Example Mining
    - Bootstraping [55]
    - SSD [19]
    - Online Hard Example Mining [24]
    - IoU-based Sampling [29]
  - C.Limit Search Space
    - Two-stage Object Detectors
    - IoU-lower Bound [17]
    - Objectness Prior [56]
    - Negative Anchor Filtering [57]
    - Objectness Module [58]
- **2.Soft Sampling Methods**
  - Focal Loss [22]
  - Gradient Harmonizing Mechanism [59]
  - Prime Sample Attention [30]
- **3.Sampling-Free Methods**
  - Residual Objectness [60]
  - No Sampling Heuristics [54]
  - AP Loss [61]
  - DR Loss [62]
- **4.Generative Methods**
  - Adversarial Faster-RCNN [63]
  - Task Aware Data Synthesis [64]
  - PSIS [65]
  - pRoI Generator [66]

III. Focal loss

a) Focal loss là gì:

Như chúng ta đã tìm hiểu phía trên về class imbalance và ảnh hưởng của nó trong thuật toán một giai đoạn. Do đó cách giải quyết cho vấn đề này thì tác giả đã suy nghĩ ra một thuật toán mang tên là Focal loss.

Focal loss là một phiên bản cải tiến của Cross entropy cố gắng xử lý vấn đề mất cân bằng lớp bằng cách gán trọng số cao cho các ví dụ khó phân biệt (p<50%) và các ví dụ chiếm ít ưu thế hơn. Trong khi đó các ví dụ chiếm số lượng lớn và dễ phân biệt (p>=50%) sẽ được gán trọng số rất nhỏ.

Trước khi giới thiệu sâu về Focal loss thì ta sẽ giới thiệu sơ về Cross-Entropy.

b) Cross entropy (CE):

Cross entropy thường được sử dụng trong máy học như một hàm mất mát. CE là một phép đo được lấy từ lĩnh vực lý thuyết thông tin, được xây dựng dựa trên entropy và thường tính toán sự khác biệt giữa hai phân phối xác suất. Trong phạm vi bài toán này ta sẽ dùng Cross entropy như là một hàm mất mát để đo lường độ hiệu quả của mô hình phân loại, hàm mất mát CE này tang lên khi xác suất dự đoán khác với thực tế và giảm dần về 0 khi xác suaart khớp hoàn toàn với nhãn thực tế của nó.

Ở trong ví dụ lần này chúng ta sẽ xét một bài toán phân lớp nhị phân, bao gồm 2 lớp : 0 và 1 . Công thức của Cross entropy của chúng ta sẽ là:

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

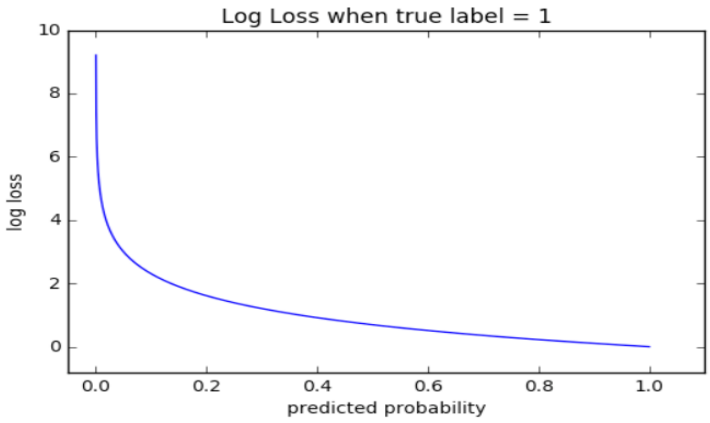
Để thu gọn công thức chúng ta sẽ đặt p_t như sau:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$$

Suy ra công thức CE cuối cùng sau khi rút gọn sẽ là:

$$\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t).$$

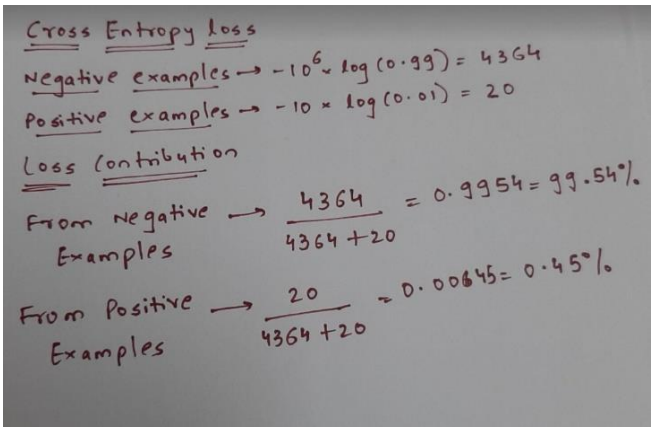
Hình dưới đây sẽ cho ta thấy khái quát được hình dạng của Cross Entropy



Qua trên ta có thể thấy được cách tính toán của Cross entropy và nó có 2 hạn chế tương đối lớn mà tác giả đã tìm ra đó là : không giải quyết tốt được khi tập dữ liệu bị lệch và những mẫu dễ dàng phân loại (p>=50%) vẫn còn bị phạt quá lớn. Do đó, tác giả đã suy nghĩ ra một phương pháp mang tên Focal loss dựa trên ý tưởng của Cross Entropy. Sự cải tiến hơn của Focal loss so với Cross Entropy đó là đã thêm 2 tham số : α và $(1-p_t)^\gamma$.

Chúng ta sẽ lấy một ví dụ về cross entropy bị chi phối thế nào khi gặp Imbalance class.

Vd : 10^6 ví dụ tiêu cực với P=0.99 , 10 ví dụ tích cực với P=0.01



Như chúng ta thấy đáng lẽ p=0.01 (có nghĩa phân lớp rất sai) với các ví dụ tích cực thì nó sẽ chiếm phần lớn trong loss, nhưng không nó chỉ chiếm 0.45% . Trong khi nhóm ví dụ tiêu cực p=0.99 (có nghĩa là phân lớp cho nó rất đúng) thì nó sẽ phải chiếm phần nhỏ trong loss, nhưng không nó chiếm tới 99.54%. Những ví dụ phân loại tương đối đúng chiếm tới 99.54% trong loss dẫn đến việc loss của chúng ta chú ý quá nhiều vào các ví dụ phân lớp đúng mà lại không chú ý đến các ví dụ phân lớp sai, trong khi hàm loss của chúng ta chỉ nên tập trung nhiều vào các ví dụ phân lớp sai. Do đó, để cân bằng việc giữa các lớp quá lệch với nhau các nhà khoa học đã nghĩ ra tham số có tên : alpha balancing.

c) Alpha Balancing

Công thức :

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t).$$

Pt: đã được định nghĩa ở phía trên

α : siêu tham số chúng ta sẽ điều chỉnh cho phù hợp với mô hình

Chức năng của alpha là cân bằng giữa các lớp với nhau, chi tiết hơn có nghĩa là điều chỉnh alpha-t của từng lớp sao cho giảm đi độ chênh lệch về số lượng dữ liệu của các lớp là ít nhất có thể. Theo nhiều bài báo, blog trên mạng thì tôi có tham khảo thấy một công thức để tính alpha-t để việc chênh lệch giữa các class là ít nhất có thể.

Công thức tính alpha-t đó là:

$$\alpha_t = \frac{1}{fi+\varepsilon}$$

fi : tần suất xuất hiện của class i

ε : là một số rất nhỏ để tránh việc chia cho 0.

Nhưng chúng ta đã nói về Cross Entropy ở phía trên, nó gặp 2 vấn đề là : không giải quyết tốt được khi tập dữ liệu bị lệch và những mẫu dễ dàng phân loại (p>=50%) vẫn còn bị phạt quá lớn. Alpha Balancing chỉ mới giải quyết được hạn chế đầu tiên mà chưa giải quyết được hạn chế thứ 2. Do đó, tác giả mới đưa ra phương pháp có tên là Focal loss phát triển để giải quyết được vấn đề số 2

d) Focal loss Definition

Alpha balances chỉ đóng vai trò là cân bằng tầm quan trọng của các lớp với nhau (giảm sự lệch của các lớp lại), nhưng nó không thể giải quyết được các ví dụ khó/ dễ.

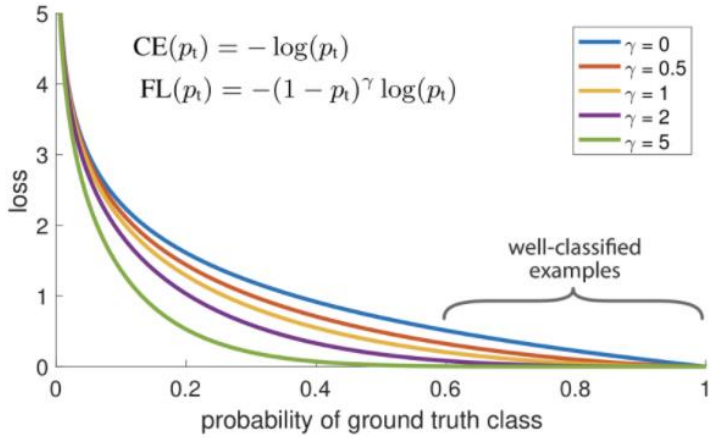
Các ví dụ khó / dễ có nghĩa là : Với pt>=50% thì chúng ta đã phân lớp đúng cho nó. Nhưng với những trường hợp đâu đó pt khoảng 50->80% (được gọi là khó vì nó không khớp hoàn toàn) thì mất mát của Entropy vẫn là quá lớn. Nên tác giả của bài báo đã suy nghĩ ra một công thức có tên là Focal loss nó giải quyết được vấn đề phạt nặng (loss cao) những thằng phân loại sai và những thằng phân loại đúng thì nó (loss rất thấp ≈≈ 0). Do

đó lúc này hàm mất mát sẽ chuyển toàn bộ sự chú ý sang những thằng phân loại sai và hoàn toàn không quan tâm đến các phân loại đúng. Công thức của Focal loss là:

$$\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t).$$

γ : đóng vai trò là điều khiển hình dạng của đường cong của loss function. Giá trị γ càng cao, tổn thất tới các ví dụ phân loại đúng càng thấp và ngược lại với phân loại sai.

Các tác giả , nhà khoa học đã chạy thực nghiệm tương đối nhiều và đã lựa chọn ra tham số γ = 2 là phù hợp nhất đối với mô hình của họ. Lí do vì sao lại như vậy chúng ta hãy nhìn hình bên dưới và cùng thảo luận với nhau



Với γ=0 thì nó chính là Cross Entropy luôn, chúng ta thấy ở đây các ví dụ dễ nhận dạng (khi pt>=50%) các ví dụ này vẫn bị phạt ở mức tương đối cao

Với γ=0.5 và γ=1 : Lần này các ví dụ dễ nhận dạng đã bị phạt ở mức thấp hơn so với γ=0 nhưng nó vẫn còn tương đối cao hơn so với γ=2 và γ=5

Với γ=5 : Lần này các ví dụ dễ nhận dạng đã xuống rất thấp và rất phù hợp để lựa chọn. Nhưng nó lại xảy ra một điều đó là các ví dụ (pt=40->50%) thì loss nó đã về 0, điều này là hoàn toàn không đúng vì chúng ta chỉ không nên phạt những thằng (pt>=50%) đằng này với γ=5 thì từ (pt>=40%) nó đã không phạt nên loại γ=5.

Với γ=2 : Lần này mọi thứ đều hoàn hảo, đối với các ví dụ dễ nó đã tiến về 0 và còn những ví dụ khó nó vẫn phạt tương đối nặng. Cho nên γ=2 là tham số mà tác giả chọn.

Khi ta thử nghiệm và gộp cả alpha-balance với focal loss thì nó cho hiệu suất tốt hơn. Do đó công thức Focal loss cuối cùng sẽ là:

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^{\gamma}\log(p_t).$$

Dễ dự báo: Chúng ta thấy rằng mô hình huấn luyện trên mẫu mất cân bằng thường dự báo chính xác các mẫu đa số. Những trường hợp này được gọi là dễ dự báo. Xác suất pt của của các trường hợp dễ dự báo có xu hướng cao hơn. Do đó $(1\text{-}p_t)^{\gamma}$ có xu hướng rất nhỏ và dường như không tác động lên loss function

Khó dự báo: Trường hợp khó dự báo thì pt là một giá trị nhỏ hơn. Do đó độ lớn tác động của nó lên loss function là $(1\text{-}p_t)^{\gamma}$ sẽ gần bằng 1. Mức độ tác động này lớn hơn rất nhiều lần so với trường hợp dễ dự báo.

γ : có thể hiểu là điều khiển hình dạng của đường cong như hình trên. Giá trị γ càng cao, tổn thất tới các ví dụ phân loại đúng càng thấp và ngược lại với phân loại sai.

Lấy một ví dụ : pt1 = 70 % và pt2 = 30% và γ =2 thì ta sẽ so sánh tỉ lệ chênh lệch mà 2 thằng này đóng góp vào loss như thế nào:

$$\frac{(1-0.3)^2}{(1-0.7)^2} = 5.44$$

Và ta thấy pt2 đóng góp vào loss gấp 5.5 lần thằng pt1

e) Tác động tới đạo hàm

Ta có đạo hàm của *focal loss* như sau:

$$\begin{aligned}
\frac{\delta FP(\mathbf{p},\mathbf{q})}{\delta q_i} &= \frac{-\alpha_i\delta[(1-q_i)^{\gamma}\log(q_i)]}{\delta q_i} \\
&= \frac{-\alpha_i[-\gamma(1-q_i)^{\gamma-1}\log(q_i) + \frac{(1-q_i)^{\gamma}}{q_i}]}{\delta q_i} \\
&= \frac{\alpha_i(1-q_i)^{\gamma}[\frac{\gamma\log(q_i)}{1-q_i} - \frac{1}{q_i}]}{\delta q_i} \\
&\triangleq \frac{\alpha_i(1-q_i)^{\gamma}g(x)}{\delta q_i}
\end{aligned}$$

Dòng thứ 4 ở biến đổi trên ta đã đặt: $g(x) = [\frac{\gamma\log(qt)}{1-qt} - \frac{1}{qt}]$

Mặt khác ta có một bất đẳng thức hết sức quan trọng đối với log(x) khi x ϵ(0,1]. Ta có : log(x) >= β($\frac{1}{x} - 1$) Với ∀β <= 1.

Chứng minh bất đẳng thức này bằng khảo sát sự biến thiên của đạo hàm khá đơn giản như sau:

Đặt f(x) = log(x) – β($\frac{1}{x} - 1$) suy ra:

$$f'(x) = \frac{1}{x} + \frac{\beta}{x^2} = \frac{x+\beta}{x^2} \leq \frac{1+\beta}{x^2} \leq 0$$

Như vậy f(x) nghịch biến trên (0,1]. Từ đó suy ra cực tiểu $\min_{x\epsilon(0,1]} f(x) = f(1) = 0$

Tương tự ta cũng chứng minh được bất đẳng thức sau:

$$\log(x) \leq \beta(\frac{1}{x} - 1) \text{ với } \forall \beta \geq 0.$$

Như vậy nếu chọn β_1 <= -1 và β_2 >= 0 là hai giá trị bất kì ta có:

$$g(x) = \frac{\gamma \log(q_i)}{1-q_i} - \frac{1}{q_i} \geq \frac{\gamma\beta_1(\frac{1}{q_i} - 1)}{1-q_i} - \frac{1}{q_i} = \gamma\beta_1 - 1 = C_1$$

Và đồng thời:

$$g(x) = \frac{\gamma \log(q_i)}{1-q_i} - \frac{1}{q_i} \leq \frac{\gamma\beta_2(\frac{1}{q_i} - 1)}{1-q_i} - \frac{1}{q_i} = \gamma\beta_2 - 1 = C_2$$

Điều này chứng tỏ giá trị của g(x) bị chặn trong đoạn [C_1,C_2]. Đây là một nhận định hết sức quan trọng vì điều đó chứng tỏ rằng độ lớn gradient của Focal loss sẽ phần lớn phụ thuộc vào $(1-q_t)^\gamma$

Bên dưới sẽ có 2 trường hợp:

> Dễ dự báo: Giá trị $(1-q_t)^\gamma$ có xu hướng rất nhỏ và do đó ảnh hưởng của gradient descent của các trường hợp dễ dự báo không đáng kể
>
> Khó dự báo: $(1-q_t)^\gamma$ sẽ gần bằng 1. Mức độ tác động lên gradient lớn hơn rất nhiều so với trường hợp dễ dự đoán.

Đó chính là điều kì diệu giúp cho các mô hình object detection sử dụng focal loss có độ chính xác cao hơn

IV. Viết và chạy thử.

```python
def binary_focal_loss(y_pred , y_true,gamma=2.0 , alpha=0.25 ,reduction="mean",function=torch.sigmoid,**kwargs):
    if isinstance(alpha,(list,tuple)):
        pos_alpha = alpha[0] # postive sample ratio in the entire dataset
        neg_alpha = alpha[1] #(1-alpha) # negative ratio in the entire dataset
    elif isinstance(alpha ,(int,float)):
        pos_alpha = alpha
        neg_alpha = (1-alpha)

    # if else in function can be simplified be removing setting to default to sigmoid  for educational purpose
    if function is not None:
        y_pred = function(y_pred , **kwargs) #apply activation function
    else :
        assert ((y_pred <= 1) & (y_pred >= 0)).all().item() , "negative value in y_pred value should be in the range of 0 to 1 inclusive"

    pos_pt = torch.where(y_true==1 , y_pred , torch.ones_like(y_pred)) # positive pt (fill all the 0 place in y_true with 1 so (1-pt)=0 and log(pt)=0.0) where pt is 1
    neg_pt = torch.where(y_true==0 , y_pred , torch.zeros_like(y_pred)) # negative pt

    pos_modulating = (1-pos_pt)**gamma # compute postive modulating factor for correct classification the value approaches to zero
    neg_modulating = (neg_pt)**gamma # compute negative modulating factor


    pos = -pos_alpha* pos_modulating*torch.log(pos_pt) #pos part
    neg = -neg_alpha* neg_modulating*torch.log(1-neg_pt) # neg part

    loss = pos+neg  # this is final loss to be returned with some reduction

    # apply reduction
    if reduction =="mean":
        return loss.mean()
    elif reduction =="sum":
        return loss.sum()
    elif reduction =="none":
        return loss # reduction mean
    else:
        raise f"Wrong reduction {reduction} is choosen \n choose one among [mean,sum,none]  "
```

```
tensor([[-0.3204,  1.9149, -0.7586],
        [ 0.7278, -1.1753, -1.3638],
        [-1.2560,  0.6942,  0.0554],
        [ 0.4899,  0.0629, -0.0683],
        [ 0.7691,  0.3859, -0.7012],
        [-1.1072,  0.4527,  0.3343],
        [ 0.0756,  1.6809,  0.7069],
        [-0.8612, -0.8888, -0.3786],
        [-0.7174, -0.4026,  0.6459],
        [ 0.5147,  2.3735,  1.5506],
        [ 0.6520, -0.5849,  0.8950],
        [-0.8604,  0.5190, -0.1252],
        [ 0.0056, -0.7319,  0.6226],
        [-1.0344,  1.1826, -0.4460],
        [-0.0643,  0.4920,  1.0905],
        [-0.0534,  1.1417,  1.9151],
        [-0.4730,  0.8037,  2.1016],
        [-0.4188,  1.2146, -0.0068],
        [-0.9137, -0.0655,  1.6845],
        [-1.4788, -0.9740, -2.3563],
        [ 0.1139,  1.1416,  0.4770],
        [-0.0334,  1.2011,  1.3362],
        [ 0.2451,  0.5860,  0.5101],
        [ 0.2738, -0.5534,  1.0397],
        [ 1.7760, -0.5403, -1.5208],
        [-0.9020, -1.5657, -0.2480],
        [ 0.7809,  0.1468,  1.0403],
        [-0.4248,  0.7806,  0.0707],
        [-0.8197,  0.4961,  0.4546],
        [-2.0410, -1.1032,  0.0363],
        [ 1.3283,  1.2612, -0.2900],
        [ 0.7583, -1.7698,  0.1724]])
```

```
tensor([[1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.],
        [0., 1., 0.],
        [1., 0., 0.],
        [0., 0., 1.]])
```

Bên trái và bên phải lần lượt là : y_predict và y_true.

Kết quả khi chạy cross entropy và focal loss lần lượt là :

```
(tensor(0.8233), tensor(0.1807))
```

V. Tài liệu tham khảo

https://machinethink.net/blog/object-detection/
https://arxiv.org/pdf/1909.00169.pdf
https://machinelearningmastery.com/what-is-imbalanced-classification/
https://medium.com/analytics-vidhya/how-focal-loss-fixes-the-class-imbalance-problem-in-object-detection-3d2e1c4da8d7
https://amaarora.github.io/2020/06/29/FocalLoss.html
https://phamdinhkhanh.github.io/2020/08/23/FocalLoss.html#1-focal-loss-function
https://medium.com/visionwizard/understanding-focal-loss-a-quick-read-b914422913e7
https://www.analyticsvidhya.com/blog/2020/08/a-beginners-guide-to-focal-loss-in-object-detection/
https://learnopencv.com/selective-search-for-object-detection-cpp-python/