UNIVERSIDADE DE SANTIAGO DE COMPOSTELA


ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

# Aprendizaxe semi-supervisada para detección de obxectos

*Autor/a:*
**Cristina López Amado**

*Titores:*
**Manuel Mucientes Molina**
**Daniel Cores Costa**

**Grao en Enxeñaría Informática**

**Xullo 2024**

Traballo de Fin de Grao presentado na Escola Técnica Superior de Enxeñaría da Universidade de Santiago de Compostela para a obtención do Grao en Enxeñaría Informática

**D. Manuel Mucientes Molina**, Profesor do Departamento de Electrónica e Computación da Universidade de Santiago de Compostela, e **D. Daniel Cores Costa**, Investigador posdoctoral no Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS),

INFORMAN:

Que a presente memoria, titulada *Aprendizaxe semi-supervisada para detección de obxectos*, presentada por **D. Cristina López Amado** para superar os créditos correspondentes ao Traballo de Fin de Grao da titulación de Grao en Enxeñaría Informática, realizouse baixo nosa titoría no Departamento de Electrónica e Computación da Universidade de Santiago de Compostela.

E para que así conste aos efectos oportunos, expiden o presente informe en Santiago de Compostela, a 3 de xullo de 2024:

Titor/a, Cotitor/a, Alumno/a,

Manuel Mucientes Molina Daniel Cores Costa Cristina López Amado

# Resumo

A detección de obxectos é un dos principais problemas que se abordan no campo da visión por computador. Os detectores tradicionais requiren dunha gran cantidade de imaxes etiquetadas para adestrar. Isto supón unha limitación importante, pois a anotación de imaxes é custosa e a dispoñibilidade dos datos é, a miúdo, limitada. Neste contexto, xorde a aprendizaxe semi-supervisada para detección de obxectos, que aborda o escenario no que existen poucas imaxes etiquetadas pero unha gran cantidade de imaxes sen etiquetar. Este traballo enmárcase nesta área e céntrase no estudo de arquitecturas de tipo *Teacher-Student*. Concretamente, estúdase o detector Unbiased Teacher v2 [1].

O obxectivo deste traballo é estudar diferentes técnicas beneficiosas noutros contextos de aprendizaxe e analizar a súa aplicabilidade a este detector. Específicamente, analizaranse as seguintes propostas: incorporación de módulos propios de arquitecturas *few-shot* (concretamente *Gradient Decoupled Layer* (GDL) e *Prototypical Calibration Block* (PCB) propostos en [2]); substitución da estratexia de limiar fixo para filtrar pseudo-etiquetas por unha estratexia de limiar flexible; e uso dunha estratexia de asignación de etiquetas baseada en *Optimal Transport Asignment* (OTA) [3].

Os experimentos realizados mostran que o uso de GDL resulta beneficioso para o rendemento do detector. Ademais, aínda que o resto de propostas non melloraron o detector, a experimentación revelou varias dificultades que xorden ao adaptar estas estratexias a un detector das características de Unbiased Teacher v2. Esta información pode resultar relevante para futuras análises do problema.

# Índice xeral


**1. Introdución** **1**
- 1.1. Descrición do problema . . . 1
- 1.2. Obxectivos e hipóteses a probar . . . 2
- 1.3. Organización da memoria . . . 3

**2. Estado do coñecemento** **5**
- 2.1. Detección de obxectos . . . 5
- 2.2. Aprendizaxe semi-supervisada para detección de obxectos . . . 7
- 2.3. Arquitectura *Teacher-Student* para detección de obxectos . . . 9

**3. Metodoloxía** **11**
- 3.1. Faster R-CNN . . . 11
- 3.2. A arquitectura Teacher-Student: Unbiased Teacher . . . 13
  - 3.2.1. Plantexamento do problema . . . 14
  - 3.2.2. Etapa de Burn-In . . . 15
  - 3.2.3. Etapa de Aprendizaxe Mutua . . . 17
- 3.3. Técnicas *few-shot* para detección de obxectos: DeFRCN . . . 20
  - 3.3.1. Gradient Decoupled Layers (GDL) . . . 21
  - 3.3.2. *Prototypical Calibration Block* (PCB) . . . 22
  - 3.3.3. Adaptación a Unbiased Teacher v2 . . . 23
- 3.4. Pseudo-etiquetado con limiar flexible . . . 24
  - 3.4.1. *Curriculum Pseudo-Labeling*(CPL) para clasificación . . . 24
  - 3.4.2. Adaptación de CPL para detección . . . 25
- 3.5. Asignación de etiquetas óptima . . . 27
  - 3.5.1. SimOTA . . . 27
  - 3.5.2. Adaptación de SimOTA a detectores de dúas etapas . . . 30

**4. Materiais** **31**
- 4.1. Repositorios de GitHub . . . 31
- 4.2. Servidores de computación . . . 31
- 4.3. Linguaxes de programación . . . 32
- 4.4. Outras ferramentas . . . 32

**5. Probas** **33**
5.1. Deseño experimental . . . . . 33
5.1.1. Métrica e conxunto de datos para detección de obxectos . 33
5.1.2. Configuración das probas e hiperparámetros . . . . . 36
5.2. Experimento 1: Proba base . . . . . 36
5.3. Experimento 2: GDL . . . . . 37
5.4. Experimento 3: PCB . . . . . 38
5.5. Experimento 4: Limiar flexible . . . . . 39
5.6. Experimento 5: SimOTA . . . . . 44

**6. Discusión dos resultados** **47**

**7. Conclusións e posibles ampliacións** **49**

**A. Manual técnico** **51**

**B. Manual de usuario** **55**
B.1. Instalación do *software* . . . . . 55
B.2. Conxunto de datos . . . . . 58
B.3. Execución de adestramentos . . . . . 58

**Bibliografía** **65**

# Índice de figuras

1.1. Exemplo de detección empregando Faster R-CNN. Localízanse os diferentes obxectos da imaxe e o modelo proporciona unha confianza para cada detección. Imaxe extraída de [4]. . . . . 1

2.1. Esquema da arquitectura típica dunha rede neuronal convolucional (CNN). Figura extraída de [5]. . . . . 5
2.2. Arquitectura Fast-RCNN. A imaxe e moitas rexións de interese (RoI) pásanse como entrada a unha CNN. Previamente, cada RoI transfórmase a un tamaño fixo mediante *pooling*. Para cada RoI, a rede ten como saída as probabilidades *softmax* da clasificación e a predición da caixa delimitadora (*bbox*). Figura extraída de [14]. 7

3.1. Arquitectura Faster-RCNN, onde se mostra a propagación cara a adiante (*forward*) e a retropropagación (*backward*). . . . . 12
3.2. Arquitectura Unbiased Teacher v2. . . . . 14
3.3. Esquema da rama de incertidume engadida a Faster R-CNN. . . . 17
3.4. Ilustración de instancias beneficiosas e enganosas. . . . . 19
3.5. Arquitectura Faster-RCNN engadindo os módulos GDL. . . . . 22
3.6. Problema na asignación de etiquetas clásica. A detección da imaxe ten un solapamento maior co *ground truth* de "persoa", polo que un método clásico de asignación de etiquetas asociará a detección a este *ground truth*. Non obstante, dado que o modelo predí "táboa de surf" para a detección, o máis razoable sería que se asignara ao *ground truth* de "táboa de surf". . . . . 27

5.1. Exemplo de Verdadeiros Positivos (VP), Falsos Positivos (FP) e Falsos Negativos (FN) para a clase "can". . . . . 35
5.2. Clases con mellores e peores resultados no detector Unbiased Teacher v2 sen modificacións. . . . . 37
5.3. Comparación dos resultados da validación dos detectores Unbiased Teacher v2 + GDL con e sen limiar flexible ao longo dun adestramento. . . . . 39
5.4. Exemplos de detección empregando limiar flexible. Ilustran algúns dos problemas que xorden con esta estratexia. . . . . 41

5.5. Comparación dos resultados da validación do detector Unbiased Teacher v2 + GDL con limiar flexible, engadindo distintas modificacións. Todos os resultados se corresponden co modelo *Teacher*. . . . . 43
5.6. Evolución do limiar flexible de cada clase ao longo dun adestramento. Cada liña de cores representa unha clase distinta. . . . . 44
5.7. Asignacións diferentes entre SimOTA e asignación baseada en IoU, onde se mostran as vantaxes de usar SimOTA. Os cadros de cores representan o *ground truth* e os cadros grises unha proposta da RPN coa súa predición. Para facilitar a visualización, só se mostra unha detección asignada de forma distinta polos dous métodos e os *ground truth* involucrados. . . . . 45
5.8. Comparación dos resultados da validación dos detectores Unbiased Teacher v2 + GDL con asignación de etiquetas con e sen SimOTA. . . . . 46

A.1. Organización en carpetas do repositorio deste traballo. . . . . 51

# Índice de cadros

5.1. Número de deteccióńs de cada categoría presentes no dataset COCO, ordeadas de máis a menos abundantes. . . . . 34
5.2. Resultados para o detector coas diferentes melloras propostas. Móstrase a media e a desviación típica das execucións realizadas. Os resultados sen desviación típica débense a que só se realizou unha execución. . . . . 37

# Capítulo 1

# Introdución

## 1.1. Descrición do problema

Este Traballo de Fin de Grao enmárcase dentro da área da **visión por computador**, un campo dentro das ciencias da computación ao que pertencen aquelas tarefas relacionadas coa análise e o procesamento de imaxes co obxectivo de extraer información sobre elas. Así, dentro da visión por computador inclúense campos como a clasificación de imaxes, a detección de obxectos, a reconstrución 3D, o seguimento de obxectos en vídeo, etc.

Neste traballo, abordaremos a disciplina da **detección de obxectos**. A súa finalidade é localizar os diferentes obxectos presentes nunha imaxe e clasificalos en función da categoría á que pertenzan, tal e como se mostra na Figura 1.1.

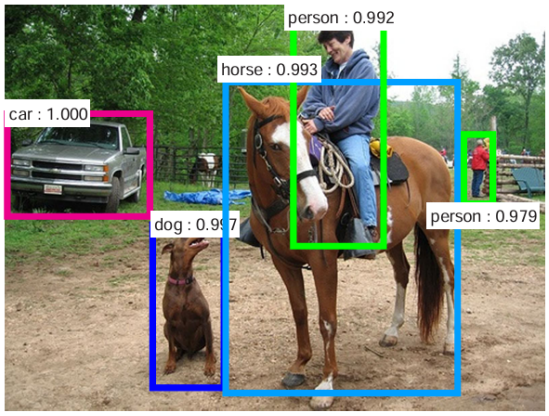



Figura 1.1: Exemplo de detección empregando Faster R-CNN. Localízanse os diferentes obxectos da imaxe e o modelo proporciona unha confianza para cada detección. Imaxe extraída de [4].

Os avances máis relevantes en visión por computador e, en particular, en detección de obxectos, obtivéronse grazas ao uso de técnicas de **Aprendizaxe**

**Automática**. A Aprendizaxe Automática é unha rama dentro da **Intelixencia Artificial**, cuxo obxectivo é dotar ás máquinas da capacidade de aprender dos datos que reciben e xeralizar esta aprendizaxe a novos datos. Máis concretamente, os problemas de visión por computador na actualidade adoitan resolverse empregando técnicas de **Aprendizaxe Profunda**, un subcampo da Aprendizaxe Automática que se caracteriza por empregar redes neuronais, arquitecturas formadas por moitas capas de procesamento onde se aplican transformacións lineais e non lineais aos datos. En visión por computador cobran gran importancia as **redes neuronais convolucionais** (**CNNs**). Así, este traballo enmárcase dentro das aplicacións da Aprendizaxe Automática á detección de obxectos.

Dentro da Aprendizaxe Automática, atópase a **aprendizaxe supervisada**. Nun escenario supervisado dispónse dun conxunto de datos de adestramento, que van acompañados de etiquetas, isto é, da saída que se desexa que o modelo obteña para eses datos. No caso da detección de obxectos, para adestrar o modelo dispónse dunha cantidade suficiente de imaxes etiquetadas, é dicir, imaxes acompañadas dos cadros que encerran os obxectos presentes na imaxe e a categoría á que pertencen ditos obxectos. Deste modo, o adestramento dos detectores tradicionais require dunha gran cantidade de imaxes anotadas. Isto limita enormemente o uso de detectores en numerosos sectores, pois o etiquetado de imaxes é unha tarefa que se fai manualmente e, por tanto, require moitos recursos. Ademais, en moitos campos de aplicación as imaxes deben ser anotadas por expertos e, noutros casos, a dispoñibilidade de imaxes é moi limitada. Un exemplo claro deste problema é a dificultade para obter e anotar imaxes médicas.

Neste contexto xorden novos campos dentro da Aprendizaxe Automática que se centran en desenvolver técnicas específicas para contextos nos que os datos etiquetados son escasos. Este traballo enmárcase dentro da **aprendizaxe semi-supervisada**, que aborda o escenario no que se dispón dunha cantidade moi limitada de datos etiquetados, pero dunha gran cantidade de datos sen etiquetar.

Específicamente, traballarase cunha arquitectura de tipo *Teacher-Student*, que é unha estratexia moi común dentro da aprendizaxe semi-supervisada. Deste modo, partirase do detector de obxectos semi-supervisado **Unbiased Teacher v2** [1] e, tras analizar o seu funcionamento, proporanse diversas modificacións co obxectivo de mellorar o seu rendemento.

## 1.2. Obxectivos e hipóteses a probar

O obxectivo principal deste traballo é o **estudo de diferentes estratexias para abordar a aprendizaxe semi-supervisada para detección de obxectos**. Para isto, partirase do detector semi-supervisado **Unbiased Teacher v2** [1] e analizarase o seu funcionamento, para despois formular unha serie de propostas co obxectivo de mellorar o seu rendemento. Así, propóñense os seguintes **obxectivos específicos**:

- **Obx. 1**: Estudo da arquitectura do detector **Unbiased Teacher v2**, para comprender o seu funcionamento.
- **Obx. 2**: Estudo de diferentes estratexias empregadas por outros modelos e análise da súa aplicabilidade a Unbiased Teacher v2.
- **Obx. 3**: Deseño e implementación das melloras sobre Unbiased Teacher v2.
- **Obx. 4**: Análise dos resultados obtidos e extracción de conclusións.

Específicamente, neste traballo plantéxanse as seguintes **hipóteses** a probar:

- **Hip. 1**: engadir os módulos ***Gradient Decouple Layer*** (**GDL**) e/ou ***Prototypical Calibration Block*** (**PCB**) propostos en [2], que son beneficiosos para arquitecturas *few-shot*, tamén melloran o rendemento de detectores semi-supervisados.
- **Hip. 2**: substituir a estratexia de limiar fixo, empregada por Unbiased Teacher v2 para filtrar pseudo-etiquetas, por unha **estratexia de limiar flexible**, podería axudar a seleccionar as detección máis adecuadas para o adestramento do *Student*.
- **Hip. 3**: substituir a asignación de etiquetas tradicional, empregada por Unbiased Teacher v2, por unha estratexia baseada en ***Optimal Transport Assignment*** **(OTA)** é beneficioso para o adestramento deste detector.

## 1.3. Organización da memoria

Seguindo o *Regulamento do Traballo Fin de Grao* do Grao en Enxeñería Informática da Universidade de Santiago de Compostela (aprobado polo Consello de Goberno o 31 de maio de 2023), este Traballo de Fin de Grao enmárcase dentro do **Tipo A**: desenvolvemento dunha idea, prototipo ou modelado teórico dun Sistema Informático que constitúa unha contribución ás técnicas da Informática. Así, os capítulos que seguen a esta introdución, que constitúe o **Capítulo 1**, organízanse da seguinte maneira:

- O **Capítulo 2** describe o **estado do coñecemento** do problema. Así, revísanse as técnicas actuais máis relevantes no campo da detección de obxectos e as diferentes propostas que existen para abordar un contexto semi-supervisado.
- No **Capítulo 3** detállase a **metodoloxía** empregada: examínase o problema a abordar, descríbense detalladamente as técnicas empregadas e xustifícanse as decisións tomadas para adaptar as distintas melloras propostas ao detector Unbiased Teacher v2.

- No **Capítulo 4** preséntanse os **materiais** *hardware* e *software* empregados na realización do traballo.
- O **Capítulo 5** inclúe o **plan de probas**. Así, descríbese a métrica utilizada para a realización das probas e détallanse os experimentos realizados e os resultados obtidos.
- No **Capítulo 6** realízase a **discusión dos resultados**, co obxectivo de comprobar as hipóteses de partida.
- O **Capítulo 7** serve como conclusión: resúmense as principais aportacións deste traballo e establécense posibles liñas de traballo futuro.

Finalmente, inclúense dous apéndices anexos a esta memoria: o **Apéndice A** correspóndese co manual técnico e o **Apéndice B** co manual de usuario. Ambos inclúen toda a información necesaria para replicar os experimentos realizados neste traballo.

# Capítulo 2

# Estado do coñecemento

## 2.1. Detección de obxectos

A detección de obxectos é un dos problemas máis amplamente estudados no campo da visión por computador. Os avances máis significativos nesta área producíronse grazas ao uso de redes neuronais e, máis concretamente, de **redes neuronais convolucionais** (**CNNs**). As CNNs son redes neuronais que se caracterizan por conter capas convolucionais. Estas capas consisten na aplicación de filtros, matrices bidimensionais cuxos pesos son aprendidos pola rede, que se van desprazando ao longo da imaxe de entrada. A vantaxe principal de empregar este tipo de capas no campo da visión por computador é o aproveitamento da información espacial entre os píxeles da imaxe. Ademais de capas convolucionais, as CNNs tamén conteñen capas de *Pooling*, cuxo obxectivo é reducir o tamaño dos mapas de características, e capas densamente conectadas (*fully connected*). A Figura 2.1 representa unha arquitectura de CNN típica.

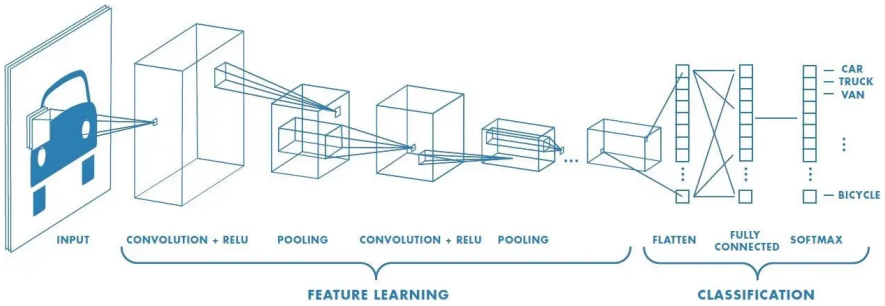



Figura 2.1: Esquema da arquitectura típica dunha rede neuronal convolucional (CNN). Figura extraída de [5].

Existe unha gran cantidade de detectores de obxectos e, en liñas xerais, podemos clasificalos en tres grandes grupos: detectores baseados en rexións (*anchor-based*), libres de rexións (*anchor-free*) e detectores baseados en *transformers*. Neste traballo, empregaremos un detector baseado en rexións, polo que explicaremos

este tipo de detectores con máis detalle. Todos estes detectores teñen en común que contan cunha rede neuronal convolucional inicial, denominada **extractor de características (*backbone*)**, cuxo obxectivo é extraer características das imaxes antes de que estas pasen a fases posteriores, como pode ser a localización das caixas. A saída do *backbone* denomínase **mapa de características**. A Figura 2.1, quitando a parte de clasificación, esquematiza a arquitectura dun *backbone* típico. Existe unha ampla variedade de extractores de características, entre os que podemos destacar **AlexNet** [6], a primeira arquitectura baseada en CNNs, e a familia de arquitecturas **ResNet** [7] e as súas variantes, como **ResNeXt** [8]. Outros exemplos de *backbones* son **GoogLeNet** [9], **DenseNet** [10] ou **ConvNeXt** [11]. Cabe destacar tamén que algúns detectores empregan como *backbone* unha **FPN** [12] (*Feature Pyramid Network*), que xera unha "pirámide" de características (mapas de características a diferentes escalas), combinando información de diferentes niveis de resolución dunha rede convolucional co obxectivo de detectar obxectos de diversos tamaños con maior precisión. O detector empregado neste traballo ten unha FPN como extractor de características.

Os **detectores baseados en rexións** (***anchor-based***), tras obter o mapa de características, obteñen para a imaxe un conxunto de propostas de rexións delimitadoras (*anchors*). Os *anchors* son un conxunto de caixas predefinidas na imaxe variando a posición, o tamaño e relación de aspecto. Así, propóñense constantemente os mesmos *anchors* para cada imaxe e estes serven como candidatos para determinar se encerran un obxecto. Os detectores baseados en rexións clasifícanse en detectores de dúas etapas e detectores dunha etapa. Por un lado, os **detectores de dúas etapas** dividen o proceso en dous pasos: primeiro, empregan os *anchors* parar xerar propostas de rexións (rexións de interese) que poden conter obxectos e, posteriormente, clasifican e refinan a regresión destas propostas. Por outro lado, os **detectores dunha etapa** realizan a detección nun só paso, realizando directamente a clasificación e o refinamento da regresión sobre os propios *anchors*. Deste modo, os detectores de dúas etapas adoitan ser máis precisos e robustos ca os dunha etapa, a costa de seren máis lentos e computacionalmente máis costosos. **R-CNN** [13] foi o primeiro detector de dúas etapas. Este foi mellorando ao longo dos anos, dando lugar primeiro a **Fast R-CNN** [14], cuxa arquitectura se ilustra na Figura 2.2, e, posteriormente, a **Faster R-CNN** [4]. Faster R-CNN é un detector amplamente utilizado na actualidade e servirá de base para realizar este traballo, polo que describiremos con máis detalle o seu funcionamento no Capítulo 3. Con respecto aos detectores dunha etapa, destacamos a serie de detectores **YOLO** [15], un dos detectores máis usados, que foi progresivamente mellorando con YOLOv2 [16] ou YOLOv3 [17]. Outros exemplos son **RetinaNet** [18] ou **SSD** [19].

Ao contrario que os detectores descritos ata agora, os **detectores libres de rexións** non utilizan *anchors*, se non que predín directamente sobre o mapa de características a posición central de cada obxecto e as dimensións da caixa delimitadora. Neste grupo destacamos os detectores **FCOS** [20] e **ATSS** [21].

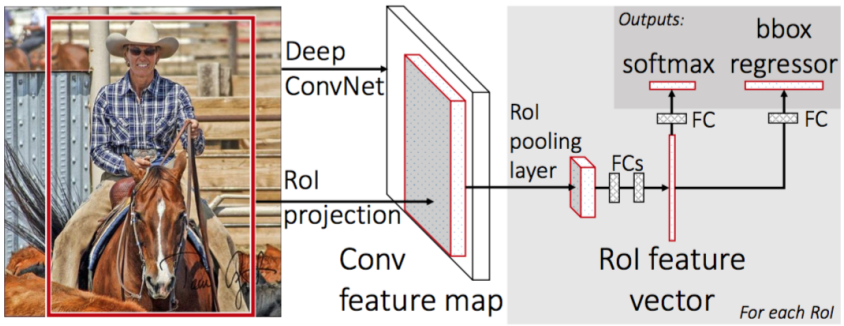



Figura 2.2: Arquitectura Fast-RCNN. A imaxe e moitas rexións de interese (RoI) pásanse como entrada a unha CNN. Previamente, cada RoI transfórmase a un tamaño fixo mediante *pooling*. Para cada RoI, a rede ten como saída as probabilidades *softmax* da clasificación e a predición da caixa delimitadora (*bbox*). Figura extraída de [14].

Finalmente, tras o éxito de empregar *transformers* no campo do procesado da linguaxe natural, nos últimos anos xurdiu unha nova liña de detectores de obxectos: os **detectores baseados en *transformers***. Estes detectores son moi competentes no campo da detección de obxectos e algúns dos máis importantes son **DETR** [22], **SMCA** [23] e **Deformable DETR** [24]. Na actualidade, o detector de obxectos que presenta un maior rendemento é **DINO** [25], que é un detector baseado en **DETR** [22]. Non obstante, debido á alta cantidade de recursos e tempo que precisan para adestrar, non empregaremos *transformers* neste traballo.

## 2.2. Aprendizaxe semi-supervisada para detección de obxectos

Os detectores de obxectos tradicionais requiren para o seu adestramento unha gran cantidade de datos etiquetados. Anotar imaxes supón en xeral un coste moi alto e non sempre é posible, polo que poder resultar unha limitación en moitos sectores. Neste contexto, xorde a **aprendizaxe semi-supervisada para detección de obxectos**, cuxo obxectivo é adestrar os detectores empregando unha pequena cantidade de imaxes etiquetadas e unha gran cantidade de imaxes sen etiquetar.

Na detección de obxectos semi-supervisada en xeral, xoga un papel moi importante o que se coñece como **transformacións dos datos** (*data augmentation*). Esta estratexia consiste en aumentar artificialmente o conxunto de datos de adestramento aplicando distintas transformacións ás imaxes orixinais. En función de se a imaxe transformada cambia pouco ou moito con respecto á imaxe orixi-

nal, adóitase distinguir entre **transformacións débiles** ou *weak augmentation* (reflexión horizontal, redimensionado,...), e **transformacións fortes** ou *strong augmentation* (cambios de cor, facer borrosa a imaxe, transformala a escala de grises, eliminar partes da imaxe,...). Esta técnica é habitual na detección de obxectos en xeral, pero cobra moita máis importancia cando se dispón de poucas imaxes etiquetadas, pois contribúe enormemente a mellorar a capacidade de xeneralización e a robustez do modelo.

Existen diferentes estratexias para integrar no modelo a información dos datos etiquetados e dos datos sen etiquetar. En liñas xerais, seguindo a clasificación realizada en [26], podemos distinguir entre os seguintes tipos de métodos: baseados en pseudo-etiquetado, baseados en regularización da consistencia, baseados en grafos e aprendizaxe por transferencia.

En primeiro lugar, os detectores **baseados en pseudo-etiquetas** parten dun modelo previamente adestrado cos datos etiquetados dispoñibles e empregan este modelo para xerar pseudo-etiquetas dos datos non etiquetados. Deste modo, pódense adestrar modelos que requiren un gran número de imaxes etiquetadas empregando, por un lado, os datos etiquetados e, por outro, os datos con pseudo-etiquetas.

En segundo lugar, os detectores **baseados en regularización da consistencia** tratan de regularizar a consistencia do modelo con respecto á saída obtida para unha mesma imaxe á que se lle aplican varias transformacións (*augmentations*). Unha extratexia para lograr isto é engadir unha función de perda que penalice as diferenzas nas predicións obtidas para unha mesma imaxe con distintas transformacións. Algúns exemplos de detectores baseados en regularización da consistencia son **CSD** [27] ou **PseCo** [28].

O terceiro grupo de detectores son aqueles **baseados en grafos**. Estes métodos representan os datos etiquetados e non etiquetados como nodos dun grafo cuxas aristas miden a semellanza entre nodos. Así, o obxectivo é propagar as etiquetas dos nodos etiquetados aos nodos sen etiquetar empregando a información de semellanza entre cada par de nodos proporcionada polas aristas. De acordo con [26], este tipo de detectores non foron aínda moi explorados na detección de obxectos en xeral, e son máis populares no *tracking* de obxectos.

Por último, os detectores **baseados en aprendizaxe por transferencia** (*transfer learning*) aproveitan que resulta máis difícil conseguir imaxes de detección anotadas (con caixas delimitadoras e clasificación de cada caixa) que imaxes con anotacións só de clasificación. Así, esta estratexia pretende transferir o coñecemento dun clasificador a un detector. Un exemplo deste tipo de detectores é **LSDA** [29].

Un campo moi relacionado coa aprendizaxe semisupervisada é a detección de obxectos ***few-shot***. A aprendizaxe *few-shot* fai referencia ao escenario da aprendizaxe no que existen moitos exemplos etiquetados dunhas certas clases (as clases base), pero moi poucos exemplos doutras clases (as clases novel). Así, falamos de $k$-*shot* para indicar que existen $k$ exemplos etiquetados de cada clase nova. Unha

estratexia moi habitual nun escenario *few-shot* é realizar un adestramento en dúas etapas: unha adestramento inicial empregando só os datos das clases base e un axuste posterior (*fine-tuning*) empregando os datos das clases novel. **DeFRCN** [2] ou **TFA** [30] son exemplos de detectores *few-shot* deste tipo. Neste traballo, adaptaremos algunhas das características de DeFRCN ao noso detector semi-supervisado.

## 2.3. Arquitectura *Teacher-Student* para detección de obxectos

Nunha arquitectura *Teacher-Student* combínase o **pseudo-etiquetado** coa **regularización da consistencia**. Nesta arquitectura pártese dun modelo preadestrado usando o conxunto de datos etiquetados. Este modelo, denominado *Teacher*, xera pseudo-etiquetas para os datos non etiquetados, que son empregadas para adestrar outro modelo, denominado *Student*. Hai que ter en conta que na maioría de casos o *Teacher* e o *Student* teñen a mesma arquitectura. Ademais, estes modelos reciben as imaxes con diferentes transformacións para xerar as pseudo-etiquetas e para adestrar, co obxectivo de regularizar a consistencia con respecto á saída obtida.

Un dos primeiros detectores que empregaron esta estratexia é **STAC** [31]. Este detector adestra en primeiro lugar un *Teacher* con todos os datos etiquetadados. A continuación, xéranse pseudo-etiquetas para todos os datos non etiquetados empregando as predicións que superen un certo limiar de confianza. Finalmente, adéstrase o *Student* empregando todos os datos: as imaxes etiquetadas e as imaxes non etiquetadas con pseudo-etiquetas.

En STAC o *Teacher* xera a pseudo-etiqueta de cada imaxe unha única vez. Non obstante, a medida que avanza o adestramento do *Student*, vaise obtendo un modelo cada vez mellor, polo que podería aproveitarse isto para mellorar a calidade das pseudo-etiquetas durante todo o adestramento. Para solucionar este problema, existen varias propostas. **Instant-Teaching** [32] xera as pseudo-etiquetas dun lote (*batch*) de imaxes e estas son empregadas a continuación como *ground truth* para adestrar o mesmo modelo. Deste modo, a pseudo-etiqueta para cada imaxe non etiquetada non é sempre a mesma, se non que mellora continuamente ao longo do adestramento. Unha versión máis refinada desta idea é a empregada en modelos como **Unbiased Teacher** [33] e **Soft Teacher** [34], nos que existen dous modelos: o *Teacher*, que xera as pseudo-etiquetas, e o *Student*, que usa as pseudo-etiquetas como *ground truth* para adestrar. Ambos modelos teñen a mesma arquitectura, pero con distintos pesos. En cada iteración, por un lado, os pesos do *Student* actualízanse mediante o seu adestramento e, por outro, os pesos do *Teacher* actualízanse mediante Exponential Moving Average (EMA) empregando os pesos do *Student*. Todos estes modelos empregan só aquelas pseudo-etiquetas

do *Teacher* que superan un certo limiar de confianza, que adoita ser a confianza na clasificación. **Unbiased Teacher v2** [1] propón medir tamén a confianza na regresión, engadindo unha rama de incertidume de localización ao modelo. Neste traballo empregarase Unbiased Teacher v2, polo que se describirá con máis detalle esta arquitectura no Capítulo 3. Existen outros detectores, como **Combating Noise** [35], que tamén incorporan a estimación da incertidume na regresión.

Todos os modelos descritos ata agora empregan as predicións duras[1] do *Teacher* como pseudo-etiquetas. Non obstante, existen outras propostas, como **Humble Teacher** [36], que empregan as predicións suaves. Por outra parte, outros detectores como **Dense Teacher** [37] non utilizan pseudo-caixas como *ground truth*, se non que propoñen outra pseudo-etiqueta unificada, denominada *pseudo-etiqueta densa*, que proporciona máis información e require menos hiperparámetros.

Todos as arquitecturas *Teacher-Student* mencionadas ata agora están deseñadas para detectores baseados en rexións, detectores libres rexións ou ambos. Un exemplo de arquitectura *Teacher-Student* baseada en *transformers* é **Omni-DETR** [38]. Non nos pararemos en describir este tipo de detectores, pois non empregaremos *transformers* no noso detector.

Na actualidade, a arquitectura semi-supervisada que presenta un mellor rendemento é **MixPL** [39]. Esta arquitectura pode ser empregada con numerosos tipos de detectores e obtén os mellores resultados cando se combina con **DINO** [25].

[1] No contexto da clasificación, denominamos predición suave (*soft prediction*) á probabilidade dunha instancia de pertencer a cada unha das clases consideradas, mentres que denominamos predición dura (*hard prediction*) á clase con probabilidade máis alta, é dicir, á clase que se predí.

# Capítulo 3

# Metodoloxía

Neste capítulo describiremos as distintas arquitecturas e métodos que se empregarán neste traballo. Así, detallaremos as características da arquitectura Unbiased Teacher v2[1], para o cal primeiro precisaremos describir Faster R-CNN [4], o detector base que emprega esta arquitectura. A continuación, explicaremos as propostas que se engadirán ao detector Unbiased Teacher v2 co obxectivo de mellorar o seu rendemento: os módulos *Gradient Decoupled Layers*(GDL) e *Prototypical Calibration Block* (PCB); unha adaptación da estratexia de limiar flexible proposta en [40]; e o método SimOTA de asignación de etiquetas implementado en [41].

## 3.1. Faster R-CNN

**Faster R-CNN** [4] é un dos modelos de detección de obxectos máis utilizados na actualidade. É a arquitectura que empregaremos como base para os noso detector e consta das seguintes compoñentes, que se ilustran na Figura 3.1:

- ***Backbone***: unha serie de capas convolucionais ao principio da rede que extrae un mapa de características a partir da imaxe de entrada. Este mapa de características será a entrada da RPN e da cabeceira do detector.
- **RPN** (*Region Proposal Network*): módulo que toma o mapa de características dunha imaxe como entrada e obtén como saída un conxunto de propostas de caixas rectangulares onde estarían os obxectos da imaxe, cada un deles acompañado dunha confianza de que esa caixa conteña un obxecto.
- Capa de **RoI Pooling** (*Regions of Interest Pooling*): toma como entrada o mapa de características inicial e as propostas da RPN, que non teñen un tamaño uniforme. Como saída, obtén un mapa de características de tamaño fixo para cada rexión de interese proposta pola RPN. Así, os mapas de características de cada proposta poden ser utilizados por un mesmo clasificador, que precisa unha entrada de tamaño fixo.

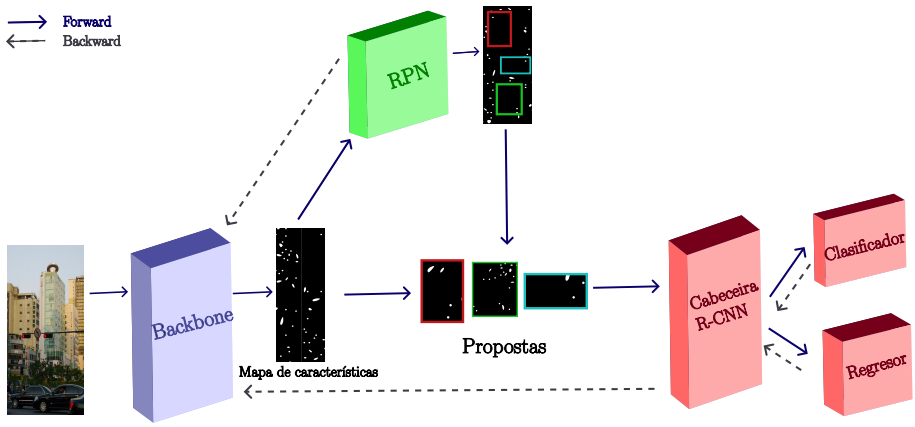



Figura 3.1: Arquitectura Faster-RCNN, onde se mostra a propagación cara a adiante (*forward*) e a retropropagación (*backward*).

- **Cabeceira R-CNN**: módulo final polo que se pasan os mapas de características obtidos pola capa de RoI Pooling. Consta dun **submódulo de clasificación**, que obtén un vector de probabilidades de que a proposta pertenza a cada unha das clases consideradas, e un **submódulo de regresión**, que obtén as coordenadas que o modelo predí para cada proposta.

A función de perda empregada en Faster R-CNN consta de dúas compoñentes:

- **Función de perda da RPN**: a RPN obtén como saída para cada imaxe un conxunto de propostas $\{\mathbf{t}_i, p_i\}_{i=1}^{N}$, onde $\mathbf{t}_i = (t_i^1, t_i^2, t_i^3, t_i^4)$ son as coordenadas preditas da caixa[1] e $p_i$ é a probabilidade de que a caixa $i$ conteña un obxecto. Así, a función de perda da RPN é:

$$\mathcal{L}_{rpn}(\{p_i, \mathbf{t}_i\}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}_{rpn}^{cls}(p_i, p_i^*) + \frac{\lambda_{rpn}}{N}\sum_{i=1}^{N}\mathcal{L}_{rpn}^{reg}(\mathbf{t}_i, \mathbf{t}_i^*) \tag{3.1}$$

  onde $N$ é o número de caixas propostas pola RPN, $\lambda_{rpn}$ é un parámetro para ponderar as perdas de clasificación $\mathcal{L}_{rpn}^{cls}$ e regresión $\mathcal{L}_{reg}$, $p_i^*$ é o *ground truth* da clasificación binaria para a proposta $i$ ($p_i^* = 0$ se é fondo e $p_i^* = 1$ se contén un obxecto) e $\mathbf{t}_i^*$ é o *ground truth* para a caixa $i$. $\mathcal{L}_{rpn}^{cls}$ correspóndese coa perda de entropía cruzada entre dúas clases (fondo/obxecto) e $\mathcal{L}_{rpn}^{reg}$ é a perda asociada coa posición da caixa cando o contido é un obxecto ($p_i^* = 1$).

$$\mathcal{L}_{rpn}^{cls} = -p_i^* \log p_i - (1 - p_i^*)\log(1 - p_i) \tag{3.2}$$

$$\mathcal{L}_{rpn}^{reg} = p_i^* \cdot \sum_{j\in\{x,y,w,h\}} L_1^{smooth}(t_i^j - t_i^{*j}) \tag{3.3}$$

[1] $(t_i^1, t_i^2)$ coordenadas da esquina inferior esquerda e $(t_i^3, t_i^4)$ da esquina superior dereita

onde $(t_i^x, t_i^y)$ son as coordenadas do centro da caixa e $t_i^w$ e $t_i^h$ as dimensións de ancho e alto respectivamente. A función de perda $L_1^{smooth}$ defínese como:

$$L_1^{smooth}(x) = \begin{cases} 0{,}5x^2 & \text{se } |x| < 1 \\ |x| - 0{,}5 & \text{noutro caso} \end{cases} \tag{3.4}$$

- **Función de perda da R-CNN**: por un lado, o submódulo de clasificación obtén como saída un vector de probabilidades $\mathbf{p}_i = (p_i^1, \dots, p_i^{N_c})$, onde $N_c$ é o número de clases consideradas. Cada elemento $p_i^c$ representa a probabilidade de que o obxecto $i$ pertenza á clase $c$ e o vector $\mathbf{p}_i$ é o resultado de aplicar a función *softmax* despois da última capa. Por outro lado, o submódulo de regresión, obtén un vector $\mathbf{t}_i = (t_i^1, t_i^2, t_i^3, t_i^4)$, que se corresponden coa predición das coordenadas da caixa. Así, a perda da R-CNN obtense como:

$$\mathcal{L}_{rcnn}(\{\mathbf{p}_i, \mathbf{t}_i\}) = \frac{1}{N} \sum_i \mathcal{L}_{rcnn}^{cls}(\mathbf{p}_i, \mathbf{p}_i^*) + \frac{\lambda_{rcnn}}{N} \sum_i \mathcal{L}_{rcnn}^{reg}(\mathbf{t}_i, \mathbf{t}_i^*) \tag{3.5}$$

onde $\lambda_{rcnn}$ é un parámetro para ponderar as perdas de clasificación e regresión, $\mathbf{p}_i^*$ é o *ground truth* da clasificación para a proposta $i$ (vector con $p_i^c = 1$ se o obxecto pertence á clase $c$ e $p_i^c = 0$ noutro caso) e $\mathbf{t}_i^*$ é o *ground truth* para a caixa $i$. $\mathcal{L}_{rcnn}^{cls}$ é a perda de entropía cruzada multiclase

$$\mathcal{L}_{rcnn}^{cls} = -\log(p_i^c) \tag{3.6}$$

onde $c$ é a clase real. $\mathcal{L}_{rcnn}^{reg}$ é a perda asociada coa posición da caixa cando o contido é un obxecto ($p_i^* = 1$) e o máis habitual é calculala como:

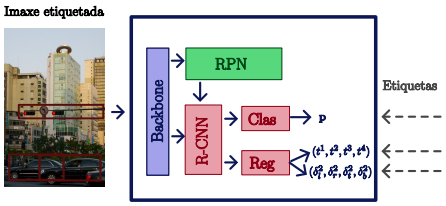
$$\mathcal{L}_{rpn}^{reg} = p_i^* \cdot \sum_{j \in \{x,y,w,h\}} L_1^{smooth}(t_i^j - t_i^{*j}) \tag{3.7}$$

Deste modo, a perda total do modelo obtense como:

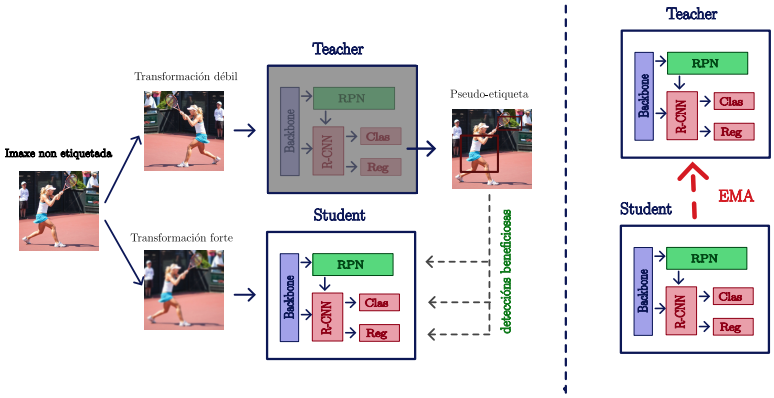
$$\mathcal{L} = \mathcal{L}_{rpn} + \mathcal{L}_{rcnn} \tag{3.8}$$

## 3.2. A arquitectura Teacher-Student: Unbiased Teacher

O obxectivo do que se coñece como arquitectura ***Teacher-Student*** é adestrar un modelo de aprendizaxe profunda nun contexto semi-supervisado. A arquitectura concreta que empregaremos para o noso detector de obxectos denomínase **Unbiased Teacher v2** [1], que é unha versión mellorada de [33]. Nestes detectores, distinguimos fundamentalmente dúas etapas: unha primeira etapa que denominamos **Burn-In** e unha segunda etapa de **Aprendizaxe Mutua**. Na etapa de




(a) Etapa de Burn-In.




(b) Etapa de Aprendizaxe Mutua. Á esquerda, adestramento do *Student* mediante pseudo-etiquetas. Á dereita, actualización do *Teacher* mediante EMA. Nótese que só se está representando a parte non supervisada do adestramento para simplificar.

Figura 3.2: Arquitectura Unbiased Teacher v2.

Burn-In, adéstrase un modelo empregando os datos etiquetados dispoñibles. Tras isto, duplícase este modelo en dúas copias: o *Teacher* e o *Student*. Así, durante a etapa de Aprendizaxe Mutua, por un lado, o *Teacher* xera pseudo-etiquetas para o conxunto de datos non etiquetados, e estas empréganse para adestrar ao *Student*. Por outro, os pesos aprendidos polo *Student* úsanse para actualizar os pesos do *Teacher*, mellorando así a calidade das pseudo-etiquetas segundo avanza o adestramento. A continuación, detallamos as características concretas da arquitectura **Unbiased Teacher v2**, que se ilustra na Figura 3.2

### 3.2.1. Plantexamento do problema

Unbiased Teacher v2 pretende abordar a detección de obxectos nun contexto de aprendizaxe semi-supervisada. Deste modo, para realizar o adestramento están dispoñibles un conxunto de imaxes etiquetadas $\mathcal{D}_s = \{\mathbf{x}_i^s, \mathbf{y}_i^s\}_{i=1}^{N_s}$ e un conxunto de imaxes non etiquetadas $\mathcal{D}_u = \{\mathbf{x}_i^u\}_{i=1}^{N_u}$, onde $N_s$ e $N_u$ son o número de imaxes etiquetadas e non etiquetadas dispoñibles. Denotamos por $\mathbf{x}_i^s$ e $\mathbf{x}_i^u$ a cada imaxe (vectores formados polos valores dos píxeles de cada imaxe) etiquetada e non etiquetada respectivamente, e por $\mathbf{y}_i^s$ ao vector que contén as localizacións de

todas as caixas presentes na imaxe $\mathbf{x}_i^s$ e a categoría á que pertence cada caixa.

### 3.2.2. Etapa de Burn-In

Partiremos dun detector de obxectos con arquitectura **Faster R-CNN**, que adestraremos en primeiro lugar empregando o conxunto de datos etiquetados do que dispoñemos $\mathcal{D}_s = \{\mathbf{x}_i^s, \mathbf{y}_i^s\}_{i=1}^{N_s}$. Denotamos por $\mathcal{L}_{sup}$ á función de perda correspondente á parte supervisada do modelo, que consta de catro compoñentes: a perda de clasificación da RPN $\mathcal{L}_{cls}^{rpn}$, a perda de regresión da RPN $\mathcal{L}_{reg}^{rpn}$, a perda de clasificación da R-CNN $\mathcal{L}_{cls}^{rcnn}$ e a perda de regresión da R-CNN $\mathcal{L}_{reg}^{rcnn}$. Así,

$$\mathcal{L}_{sup} = \sum_i \mathcal{L}_{cls}^{rpn}(\mathbf{x}_i^s, \mathbf{y}_i^s) + \mathcal{L}_{reg}^{rpn}(\mathbf{x}_i^s, \mathbf{y}_i^s) + \mathcal{L}_{cls}^{rcnn}(\mathbf{x}_i^s, \mathbf{y}_i^s) + \mathcal{L}_{reg}^{rcnn}(\mathbf{x}_i^s, \mathbf{y}_i^s) \tag{3.9}$$

As funcións $\mathcal{L}_{cls}^{rpn}$ e $\mathcal{L}_{reg}^{rpn}$ son as xa descritas para Faster R-CNN. As funcións de perda da R-CNN modifícanse para adaptalas a un escenario semi-supervisado.

Por unha parte, para obter $\mathcal{L}_{cls}^{rcnn}$ substitúese a perda de entropía cruzada usada en Faster-RCNN pola **función de perda focal** (***Focal Loss***) **multiclase**. O desbalanceo de clases é un problema inherente á detección de obxectos. Tal e como se afirma en [33], este desbalanceo existe tanto entre as detección fondo-obxecto (arredor dun 70 % das detección do adestramento son fondo) como entre as detección presentes para diferentes clases (MS-COCO [42] contén detección de 80 clases distintas, das cales arredor dun 30 % se corresponden coa clase "persoa"). Por este motivo, se se emprega directamente entropía cruzada na función de perda, o modelo tenderá a predicir en maior medida as clases dominantes. Este problema está presente en calquer detector de obxectos, pero cobra máis importancia nun escenario semi-supervisado, debido a que estas predicións sesgadas cara ás clases dominantes empregaranse como pseudo-etiquetas para adestrar outro detector. Para abordar este problema, substitúese na R-CNN a función de perda de entropía cruzada pola función de perda focal, obtendo

$$\mathcal{L}_{cls}^{rcnn} = -(1 - p_i^c)^\gamma \log(p_i^c) \quad \gamma > 0 \tag{3.10}$$

Vemos que a perda focal multiplica a entropía cruzda polo termo $(1 - p_i^c)^\gamma$, que é maior canto menor sexa a confianza na clasificación. Deste modo, as detección con maior confianza na clasificación teñen menos influencia na función de perda que as detección "difíciles" de clasificar, o cal fai que o modelo se centre máis en clasificar correctamente poucas detección "difíciles" (probablemente pertencentes ás clases menos representadas) fronte a moitas detección "fáciles" (probablemente pertencentes ás clases dominantes). Ademais, canto máis próximo a 0 sexa o hiperparámetro $\gamma > 0$, maior peso cobrarán as clases menos representadas.

Por outra parte, o cálculo de $\mathcal{L}_{reg}^{rcnn}$ tamén cambia con respecto ao empregado en Faster R-CNN. Tal e como explicaremos no seguinte apartado, na etapa de

Aprendizaxe Mutua será necesario obter unha estimación da incertidume dos modelos *Teacher* e *Student* para cada detección. Por este motivo, introdúcese unha rama adicional ao submódulo de regresión: a **rama de incertidume de localización**. Esta pretende medir a incertidume que ten o modelo con respecto á predición de regresión obtida. Esta incertidume debería ser maior canto maior sexa o erro que comete o modelo ao facer unha predición. A rama de incertidume de localización ten como saída un vector $\boldsymbol{\delta}$ con catro coordenadas, que se corresponden coa incertidume do modelo con respecto a cada unha das catro coordenadas da regresión. Para adestrar este novo submódulo de regresión, emprégase unha función de perda que se coñece como ***negative power log-likelihood*** (**NPLL**) e que describimos a continuación. Tras esta primeira etapa, duplicamos o modelo cos seus pesos para obter dous modelos: o *Teacher* e o *Student*.

### Estimación da incertidume: función de perda NPLL

Para modelar a incertidume de localización do modelo, emprégase a solución proposta en [43]. A rama de incertidume, ilustrada na Figura 3.3, adéstrase xunto coa rama de predición das caixas e ambas comparten os mesmos pesos. A saída da rama de incertidume é un vector $\boldsymbol{\delta}$, cuxas compoñentes toman valores entre 0 e 1 (canto máis cercano a 1, maior incertidume). Para obter este valor, aplícase a función *softmax* despois da última capa do submódulo de regresión. Para que este valor estime correctamente a incertidume do modelo sobre unha predición, precisamos introducir unha nova función de perda na etapa de Burn-In. Esta coñécese como ***negative power log-likelihood*** (**NPLL**) e a continuación explicaremos brevemente en que consiste.

Para predicir a incertidume asociada a cada unha das coordenadas de cada caixa, modelaremos estas coordenadas como distribucións normais e adestraremos o modelo para estimar as súas desviacións típicas. A intuición detrás desta idea é que canto maior sexa a desviación típica, maior será a incertidume do modelo. Denotamos por $\boldsymbol{d_g} = (d_g^l, d_g^r, d_g^t, d_g^b)$ e $\boldsymbol{d} = (d^l, d^r, d^t, d^b)$ ao *ground truth* e á predición para unha caixa respectivamente, onde $d^i$ se corresponde coa distancia do centro da caixa á parte esquerda ($i = l$), dereita ($i = r$), superior ($i = t$) ou inferior ($i = b$) da imaxe. Deste modo, modelamos cada predición $\boldsymbol{d}$ como:

$$P_\Theta(\boldsymbol{d_g}|\boldsymbol{d}) = \mathcal{N}(\boldsymbol{d}, \Sigma)$$

onde $\Theta$ son os parámetros do modelo e $\Sigma = \text{diag}(\delta_l^2, \delta_r^2, \delta_t^2, \delta_b^2)$ denota as incertidumes predicidas polo modelo.

Deste xeito, pretendemos maximizar o logaritmo da probabilidade (da función de densidade) de obter un certo *ground truth* dada unha predición. Ademais, como as predicións nunca coincidirán exactamente con ningún *ground truth*, ponderamos a función de perda de cada predición polo IoU entre a predición e o *ground truth*. Así, as deteccióǹs con maior IoU influirán máis na función de perda. Empregando a expresión da función de densidade dunha distribución normal, obtemos

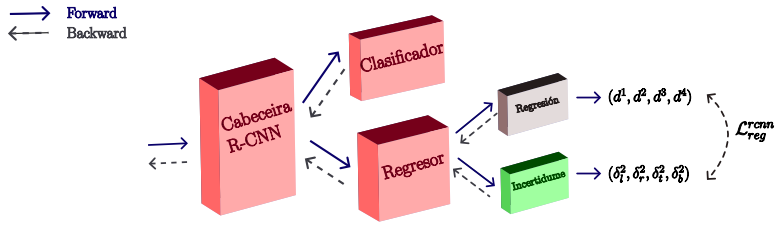



Figura 3.3: Esquema da rama de incertidume engadida a Faster R-CNN.

a seguinte función de perda para cada detección $d$ con *ground truth* asociado $d_g$:

$$\mathcal{L} = -\sum_{k\in\{l,r,t,b\}} \text{IoU} \cdot \log(P_\Theta(d_g^k \mid d^k, \delta_k^2)) \tag{3.11}$$

$$= \text{IoU} \cdot [\sum_{k\in\{l,r,t,b\}} (\frac{(d_g^k - d^k)^2}{2\delta_k^2} + \frac{1}{2}\log\delta_k^2) + 2\log 2\pi] \tag{3.12}$$

Así, se unha imaxe ten detecciónss $\{\boldsymbol{d_j}\}_{j=1}^M$, onde cada $\boldsymbol{d_j}$ ten *ground truth* asociado $\boldsymbol{d_{g_j}}$ con IoU $\eta_j$ e incertidume $\{\boldsymbol{\delta_j}\}_{j=1}^M$,con $\boldsymbol{\delta_j} = (\delta_{j,l}^2, \delta_{j,r}^2, \delta_{j,t}^2, \delta_{j,b}^2)$ a perda da regresión durante a etapa supervisada calcúlase como:

$$\mathcal{L}_{reg}^{rcnn} = \sum_j (\eta_j \sum_{k\in\{l,r,t,b\}} (\frac{(d_{g_j}^k - d_j^k)^2}{2\delta_{j,k}^2} + \frac{1}{2}\log\delta_{j,k}^2) + 2\log 2\pi) \tag{3.13}$$

### 3.2.3. Etapa de Aprendizaxe Mutua

Como xa se comentou, o obxectivo desta etapa é que os modelos ***Teacher*** e ***Student*** evolucionen de maneira conxunta, interactuando para mellorar a súa precisión continuamente ao longo do adestramento. Para isto, por un lado, o *Teacher* xera **pseudo-etiquetas** para as imaxes non etiquetadas e estas úsanse para adestrar ao *Student*. Por outro, os pesos do *Teacher* vanse actualizando de maneira progresiva empregando os pesos aprendidos polo *Student*, mediante o que se coñece como ***Exponential Moving Average*** (**EMA**). Deste xeito, conseguimos que as etiquetas do *Teacher* melloren de maneira continua e estable ao longo do adestramento. Hai que ter en conta que, ademais das pseudo-etiquetas, o *Student* tamén recibe durante esta etapa imaxes do conxunto de datos supervisado. Deste modo, a función de perda do *Student* calcúlase combinando a perda do adestramento supervisado coa perda do adestramento non supervisado como:

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda_u \mathcal{L}_{unsup} \tag{3.14}$$

onde $\mathcal{L}_{sup}$ calcúlase como en 3.9 e $\lambda_u$ é un hiperparámetro. A perda non supervisada $\mathcal{L}_{unsup}$ obtense do mesmo modo, só que empregando pseudo-etiquetas en lugar de *groud-truth*, exceptuando a perda de regresión da RCNN, que detallaremos a continuación.

**Aprendizaxe do *Student* mediante pseudo-etiquetado**

Unha primeira aproximación para levar a cabo o pseudo-etiquetado, podería ser establecer un limiar mínimo de confianza, de modo que o *Student* reciba como *ground truth* as imaxes coas pseudo-etiquetas xeradas polo *Teacher* que estean por enriba dese limiar. Este enfoque é o máis empregado en modelos cunha arquitectura *Teacher-Student* para clasificación, pero non é tan adecuado aplicalo directamente a detección de obxectos. O motivo principal é que probablemente aparezan caixas duplicadas que en realidade encerran un só obxecto. Para solucionar este problema, [33] aplica **non-maximum supression (NMS)**[2] para eliminar as caixas repetidas antes de filtrar empregando o limiar de confianza. En [33] emprégase este procedemento para adestrar o *Student* e o nivel de confianza do *Teacher* para cada detección vén dado pola probabilidade que obtén a clase asignada cando se fai a clasificación na R-CNN. Non obstante, este enfoque presenta unha limitación importante: as caixas selecciónanse tendo en conta o nivel de confianza do *Teacher* con respecto á clasificación, pois o módulo de regresión de Faster R-CNN ten como saída as coordenadas da caixa predita, pero non se estima a confianza desta predición. Por este motivo, esta aproximación non mide a confianza que ten o *Teacher* sobre a regresión. Para solucionar este problema, [1] propón mellorar a rama de regresión do modelo, engadindo unha estimación da confianza sobre a regresión.

O obxectivo de mellorar a rama de regresión do modelo é conseguir seleccionar aquelas propostas que son *beneficiosas* para adestrar o *Student* e eliminar aquelas que son *enganosas*. O enfoque da aprendizaxe mutua, no que o *Teacher* se actualiza continuamente mediante EMA, debería dar lugar a que o *Teacher* sexa máis preciso e estable que o *Student*. Deste modo, se denotamos por $\tilde{\boldsymbol{d}}_{\boldsymbol{t}}$, $\tilde{\boldsymbol{d}}_{\boldsymbol{s}}$ e $\boldsymbol{d}_{\boldsymbol{g}}$ as predicións para a regresión do *Teacher*, *Student* e a etiqueta do *ground truth* respectivamente, podemos definir dous tipos de instancias:

- Unha **instancia *beneficiosa*** é aquela que satisfai $||\tilde{\boldsymbol{d}}_{\boldsymbol{t}} - \boldsymbol{d}_{\boldsymbol{g}}|| \leq ||\tilde{\boldsymbol{d}}_{\boldsymbol{s}} - \boldsymbol{d}_{\boldsymbol{g}}||$.
- Unha **instancia *enganosa*** é aquela que satisfai $||\tilde{\boldsymbol{d}}_{\boldsymbol{t}} - \boldsymbol{d}_{\boldsymbol{g}}|| > ||\tilde{\boldsymbol{d}}_{\boldsymbol{s}} - \boldsymbol{d}_{\boldsymbol{g}}||$.

Deste xeito, como o *Teacher* debería ter resultados mellores ca o *Student*, tal e como se ilustra na Figura 3.4, consideramos que as instancias nas que o *Student* predí mellor son *enganosas* e non se deberían usar para o adestramento.

Nun contexto semi-supervisado, as etiquetas $\boldsymbol{d}_{\boldsymbol{g}}$ non están dispoñibles, polo que é necesario buscar outra aproximación para decidir se o *Teacher* é mellor ca o *Student* ou non. Para isto, introdúcese unha rama adicional ao submódulo de regresión: a **rama de incertidume de localización**. Tal e como se describiu no apartado anterior, esta rama pretende medir a incertidume que teñen cada

[2]NMS é unha técnica habitual en detección de obxectos que consiste en agrupar nunha mesma detección aquelas caixas que teñan unha superposición (IoU) maior ca certo valor. De entre todas as caixas agrupadas, elíxese aquela con unha confianza de clasificación maior


Figura 3.4: Ilustración de instancias beneficiosas e enganosas.

un dos modelos con respecto á predición de regresión obtida. O obxectivo de engadir esta rama é poder dar unha nova definición de instancia *enganosa* para poder filtrar as pseudo-etiquetas que se empregan para adestrar o *Student* sen necesidade de dispoñer do *ground truth*. Así, na etapa de Aprendizaxe Mutua, empregaranse as incertidumes do *Teacher* e do *Student* para eliminar aquelas predicións $\hat{d}_s^k$ con $k = l, r, t, b$ que teñen asociadas pseudo-etiquetas que cumpren algunhas das seguintes condicións:

- O *Student* ten unha confianza demasiado alta: $\delta_s^k \leq \sigma_s$
- O *Student* ten maior confianza ca o *Teacher*: $\delta_s^k < \delta_t^k + \sigma$

onde $\delta_s^k$ é a incertidume obtida polo *Student* para a predición $\hat{d}_s^k$, $\delta_t^k$ é a incertidume obtida polo *Teacher* para a pseudo-etiqueta asociada $\hat{d}_t^k$ e $\sigma_s, \sigma \geq 0$ son hiperparámetros que regulan a confianza máxima permitida para o *Student* e o marxe permitido no que o *Student* pode superar ao *Teacher*. Así, a función de perda non supervisada para a regresión obtense como:

$$\mathcal{L}_{reg}^{unsup} = \begin{cases} \sum_k |\hat{d}_t^k - d_s^k| & \text{se } \delta_s^k > \sigma_s \quad \text{e} \quad \delta_t^k + \sigma \leq \delta_s^k \\ 0 & \text{noutro caso} \end{cases} \tag{3.15}$$

Cabe destacar que na función de perda non supervisada se teñen en conta de maneira independente cada un dos bordos da caixa (esquerdo, dereito, superior e inferior), é dicir, para unha mesma caixa pode ser que algúns dos seus bordos se usen na función de perda e outros non.

A idea fundamental que xustifica este procedemento para filtrar as pseudo-etiquetas é que o *Teacher* só debería guiar ao *Student* naquelas instancias nas que teña menos incertidume, pois isto indica que o *Teacher* está cometendo menos erro ca o *Student*. Pola contra, se o *Student* comete menos erro ca o *Teacher* nunha detección, non se debería empregar a instancia "errónea" do *Teacher*, pois esta guiaría ao *Student* cara a unha detección peor.

### Actualización do *Teacher* mediante *Exponential Moving Average* (EMA)

As pseudo-etiquetas xeradas polo *Teacher* fan que sexa posible adestrar o *Student* con novas imaxes, facendo que o modelo mellore continuamente. Parece

razoable empregar esta mellora á súa vez para xerar mellores pseudo-etiquetas. Para isto, é necesario actualizar o *Teacher* cos novos pesos aprendidos polo *Student*. Unha primeira aproximación podería ser actualizar directamente o *Teacher* cos novos pesos do *Student* en cada iteración. Non obstante, isto daría lugar a un *Teacher* moi cambiante que xeraría etiquetas inestables. Por este motivo, a solución elexida para facer esta actualización é aplicar o que se coñece como ***Exponential Moving Average*** **(EMA)**. Se denotamos por $\theta_t$ e $\theta_s$ os pesos do *Teacher* e do *Student* respectivamente, os pesos do *Teacher* actualízanse en cada iteración mediante EMA do seguinte xeito:

$$\theta_t \leftarrow \alpha\theta_t + (1-\alpha)\theta_s \tag{3.16}$$

onde $\alpha$ é un hiperparámetro. Así, os pesos do *Teacher* cambian máis lentamente e as pseudo-etiquetas xeradas son máis estables. O *Teacher* obtido desta maneira pode entenderse como un "resumo" do conxunto dos modelos *Student* en diferentes iteracións do adestramento. Ademais, en [33] obsérvase que o feito de utilizar EMA para actualizar o *Teacher* tamén reduce o problema asociado ao desbalanceo de clases comentado na sección anterior.

#### Transformacións fortes e débiles das imaxes

Por último, xa se comentou que unha técnica moi empregada en contextos semi-supervisados é aumentar artificialmente o conxunto de datos aplicando diferentes **transformacións** ás imaxes orixinais. En Unbiased Teacher v2, durante a etapa de Aprendizaxe Mutua, o *Teacher* recibe imaxes con transformacións débiles para xerar as pseudo-etiquetas que sexan confiables, mentres que o *Student* recibe estas mesmas imaxes con transformacións fortes, para que este reciba unha maior diversidade de imaxes. Ademais de no adestramento non supervisado, tamén se emprega un conxunto de datos aumentado con transformacións fortes e débiles durante o Burn-In e na parte supervisada da etapa de Aprendizaxe Mutuo.

## 3.3. Técnicas *few-shot* para detección de obxectos: DeFRCN

Unha das hipóteses deste traballo é comprobar se é posible aplicar a un detector semi-supervisado algunha das técnicas propias dun escenario *few-shot*. Para iso, introduciremos no noso detector os módulos **GDL** e **PCB** propostos para o detector **DeFRCN** [2], un detector de obxectos *few-shot* con detector base Faster R-CNN, que propón unha serie de melloras para adaptalo a un contexto *few-shot*. Un escenario *few-shot* e un semi-supervisado comparten a característica común de dispoñer de poucos datos etiquetados. Por este motivo, é interesante analizar

se os problemas detectados en [2] para os detectores de obxectos *few-shot* e as solucións propostas son aplicables ao noso detector nun contexto semi-supervisado.

### 3.3.1. Gradient Decoupled Layers (GDL)

Unha das melloras que se proporán para o detector Unbiased Teacher v2 é engadir o módulo ***Gradient Decoupled Layer*** **(GDL)** proposto para o detector **DeFRCN** [2].

Como xa se comentou, en Faster R-CNN as imaxes son procesadas en primeiro lugar a través do *backbone*, unha rede convolucional que extrae un mapa de características que será a entrada tanto da RPN como da R-CNN. A RPN emprega este mapa de características para xerar propostas que son clase-agnósticas (só clasifican en obxecto ou fondo) e a R-CNN utilízao para levar a cabo a regresión e a clasificación das propostas da RPN, diferenciando entre as distintas clases. Polo tanto, todos estes módulos se optimizan de maneira conxunta minimizando unha única función de perda. A Figura 3.1 ilustra a retropropagación do gradiente en Faster R-CNN, mostrando a situación que se acaba de explicar. Isto xera un problema, pois non todos os módulos do detector perseguen o mesmo obxectivo: empregando o mesmo mapa de características extraído polo *backbone*, a RPN pretende xerar propostas de obxectos, sen ter en conta a que clase pertencen, mentres que que a R-CNN pretende identificar a que clase concreta pertence cada obxecto. Deste modo, os pesos do *backbone* optimízanse combinando a función de perda da RPN, que é clase-agnóstica, e a función de perda da R-CNN, que distingue entre as distintas clases.

Nun escenario *few-shot* é habitual adestrar o detector que contén todas as clases, partindo dun detector adestrado só coas clases base. Así, é fácil que a RPN confunda obxectos con fondo, pois obxectos das clases novas eran considerados como fondo no adestramento base. Por este motivo, o problema descrito é especialmente importante nun escenario *few-shot*. Nun escenario semi-supervisado non se da exactamente este problema. Non obstante, o número de imaxes etiquetadas dispoñibles nun escenario semi-supervisado para algunhas clases pode ser moi semellante ás dispoñibles nun contexto *few-shot*. Por exemplo, o conxunto de datos COCO contén aproximadamente 200 instancias pertencentes á clase "secador de pelo". Se se considera o caso no que se dispón do 2 % das imaxes etiquetadas, unha proporción moi habitual na literatura, o número de secadores etiquetados dispoñibles será cercano a 4-shot. Así, parece razoable pensar que o problema descrito tamén podería aparecer no noso detector, polo que trataremos de aplicar a solución proposta para DeFRCN.

Para solucionar este problema, [2] propón axustar o grao de desacoplamento entre os módulos *backbone*, RPN e R-CNN engadindo un novo módulo á arquitectura: o módulo GDL. Así, tal e como se mostra na Figura 3.5, engádense dous módulos GDL: un entre o *backbone* e a RPN e outro entre o *backbone* e a RCNN. Por un lado, durante a propagación cara a adiante (*forward propagation*), GDL

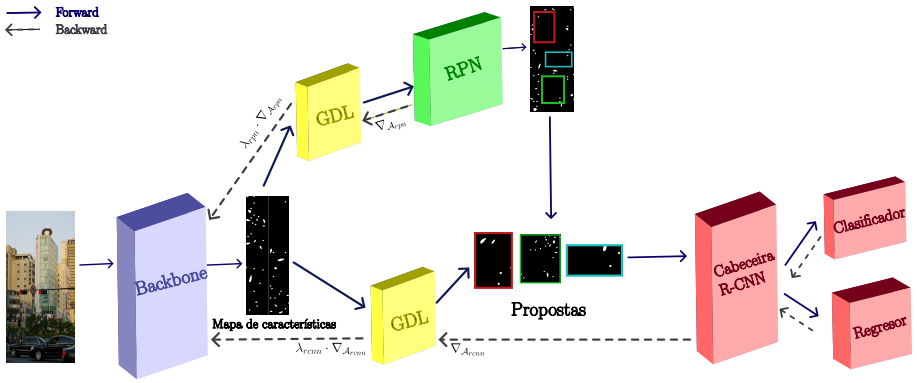



Figura 3.5: Arquitectura Faster-RCNN engadindo os módulos GDL.

consiste nunha única capa densamente conectada cunha matriz de pesos e un *bias* adestrables. Deste modo, partindo dunha mesma entrada, o vector de características que entra na RPN será distinto do que entra na RCNN. Se denotamos por $\mathcal{A}_{rpn}$ e $\mathcal{A}_{rcnn}$ ás transformacións afíns dos módulos GDL colocados antes da RPN e da R-CNN respectivamente, dado un vector de entrada $x$, o vector de características que entra na RPN será $\mathcal{A}_{rpn}(x)$ e o que entra na R-CNN é $\mathcal{A}_{rcnn}(x)$. Por outro lado, durante a retropropagación (*back-propagation*), o módulo GDL multiplica o gradiente procedente da capa seguinte por unha constante e propágao cara á capa anterior. Deste xeito, o Jacobiano da transformación asociada ao GDL multiplícase por unha constante $\lambda \in [0, 1]$ antes de pasarllo á capa anterior. Isto permite regular a influencia dos pesos da RPN e da R-CNN na actualización dos pesos do *backbone*. Así, se denotamos por $\lambda_{rpn}$ e $\lambda_{rcnn}$ as constantes polas que se multiplica o gradiente no GDL da RPN e da R-CNN respectivamente, os pesos do *backbone*, $\theta_b$, actualízanse nun paso de descenso de gradiente como:

$$\theta_b \longleftarrow \theta_b - \gamma \left( \lambda_{rpn} \frac{\partial \mathcal{L}_{rpn}}{\partial \theta_b} + \lambda_{rcnn} \frac{\partial \mathcal{L}_{rcnn}}{\partial \theta_b} \right) \tag{3.17}$$

onde $\gamma$ é a taxa de aprendizaxe (*learning rate*) e $\mathcal{L}_{rpn}$ e $\mathcal{L}_{rcnn}$ as funcións de perda da RPN e da R-CNN respectivamente. Así, engadir os módulos GDL non afecta aos pesos da RPN e da RCNN, se non unicamente aos pesos do *backbone*. Ademais, canto máis cercana sexa $\lambda_{rpn}$ ($\lambda_{rcnn}$) a 1 e $\lambda_{rcnn}$ ($\lambda_{rpn}$) a 0, máis influencia terán os pesos da RPN (RCNN) fronte aos da R-CNN (RPN).

### 3.3.2. *Prototypical Calibration Block* (PCB)

A segunda mellora que se proporá para o detector Unbiased Teacher v2 é engadir o ***Prototypical Calibration Block* (PCB)** proposto para **DeFRCN** [2].

A cabeceira R-CNN de Faster R-CNN debe realizar dúas subtarefas: a clasificación, que precisa características invariantes por translacións, e a regresión,

cuxas características deben variar se hai translacións. Deste modo, de maneira análoga ao que se explicou no apartado anterior para a RPN e a R-CNN, os submódulos de clasificación e regresión da cabeceira perseguen obxectivos distintos, pero comparten características que se optimizan en conxunto. Para reducir este problema, [2] propón engadir un módulo PCB durante a etapa de inferencia, coa función de desacoplar as tarefas de clasificación e regresión.

Este PCB consiste nun **clasificador** preadestrado con Imagenet [44], unha capa de **RoIAlign** e un **banco de prototipos**. Cada imaxe con clases novel pásase polo extractor de características e emprégase RoIAlign xunto coas caixas *ground truth* para obter unha representación de cada instancia das clases novel. O banco de prototipos está formado por un prototipo $p_c$ para cada clase novel, que se obtén como:

$$p_c = \frac{1}{|S_c|} \sum_{(x_i, y_i) \in S_c} x_i \tag{3.18}$$

onde $S_c$ é o conxunto de imaxes que contén algunha instancia da clase $c$, os $x_i$ son as características extraidas da instancia e $y_i$ é a etiqueta. Na etapa de inferencia, dada unha proposta $\hat{y}_i = (c_i, s_i, b_i)$, onde $c_i$ e $b_i$ son a clase e a caixa preditas e $s_i$ a confianza na clasificación, o PCB obtén as características $x_i$ para a caixa $b_i$ e calcula a similaridade coseno entre $x_i$ e $p_{c_i}$ como:

$$s_i^{cos} = \frac{x_i \cdot p_{c_i}}{||x_i||\,||p_{c_i}||} \tag{3.19}$$

Así, calcúlase unha nova confianza na clasificación $s_i^*$ como:

$$s_i^* = \alpha s_i + (1 - \alpha) s_i^{cos} \tag{3.20}$$

onde $\alpha$ é un hiperparámetro.

### 3.3.3. Adaptación a Unbiased Teacher v2

O feito de que DeFRCN e Unbiased Teacher v2 teñan ambos como detector base Faster R-CNN, facilita a integración dos módulos GDL descritos. Así, é necesario modificar o detector Faster R-CNN base para que conteña estes dous módulos. Con respecto aos parámetros $\lambda_{rpn}$ e $\lambda_{rcnn}$, [2] determina que os valores máis adecuados para DeFRCN son $\lambda_{rpn} = 0$ e $\lambda_{rcnn} = 1$, o cal implica que a función de perda da RPN non se emprega para optimizar o *backbone*. Na sección 5.3 detallaremos os valores destes parámetros para o noso detector.

Con respecto ao PCB, é unha compoñente que só se utiliza na etapa de inferencia. No noso detector semi-supervisado empregarémolo na inferencia que fai o *Teacher* para xerar as pseudo-etiquetas para adestrar o *Student*. Ademais, crearemos o banco de prototipos co conxunto de imaxes etiquetadas das que dispoñemos.

## 3.4. Pseudo-etiquetado con limiar flexible

Como se explicou na sección 3.2, o detector Unbiased Teacher v2 que empregamos utiliza como pseudo-etiquetas para adestrar o *Student* aquelas detecciónsdo *Teacher* que superen un certo limiar de confianza. Este limiar é fixo durante todo o adestramento e común a todas as clases, o cal é unha aproximación habitual dentro da aprendizaxe semi-supervisada. Non obstante, aínda que esta estratexia asegura que soamente se empreguen detecciónsde alta calidade para adestrar o *Student*, tamén ignora unha cantidade considerable de imaxes. Ademais, as etiquetas correspondentes a cada clase trátanse do mesmo xeito, sen ter en conta que unhas clases son máis difíciles de aprender ca outras.

### 3.4.1. *Curriculum Pseudo-Labeling*(CPL) para clasificación

Para resolver estes problemas, [40] propón unha estratexia de aprendizaxe que denomina *Curriculum Pseudo-Labeling* (CPL), ou pseudo-etiquetado curricular. Esta consiste en substituír o limiar fixo e común a todas as clases por un limiar flexible distinto para cada clase, e que se vai axustando dinamicamente de acordo co estado da aprendizaxe de cada clase. [40] deseña este método para mellorar un clasificador de imaxes. A continuación, describiremos brevemente en que consiste e explicaremos detalladamente que elementos é necesario modificar para poder adaptalo ao noso modelo de detección.

A idea de usar un limiar dinámico é que as predicións das clases que alcanzan unha precisión menor durante a avaliación teñan que superar un limiar máis baixo que aquelas cunha maior precisión. Isto fomenta que se usen máis exemplos das clases con menos precisión e que se sexa máis exixente coa calidade das pseudo-etiquetas naquelas clases que xa alcanzan unha precisión alta. Non obstante, non é posible obter a precisión alcanzada por cada clase durante o adestramento nun contexto semi-supervisado, pois non resulta viable reservar unha porcentaxe dos datos etiquetados, que xa son escasos, para realizar validacións. Por isto, é necesario buscar outra maneira de estimar o estado da aprendizaxe de cada clase. [40] considera que, fixado un limiar, aquelas clases nas que un número pequeno de instancias superan ese limiar teñen unha maior dificultade de aprendizaxe. Deste xeito, estima o estado da aprendizaxe dunha clase $c$ no instante $t$ como:

$$\sigma_t(c) = \sum_{n=1}^{N} \mathbb{1}(\text{máx}(p_{m,t}(y|u_n)) > \tau) \cdot \mathbb{1}(\arg\text{máx}(p_{m,t}(y|u_n)) = c) \tag{3.21}$$

onde $p_{m,t}(y|u_n)$ é a predición do modelo para a imaxe etiquetada $u_n$ no instante $t$ e $N$ é o número total de imaxes non etiquetadas. Así, se o conxunto de datos está balanceado, canto maior sexa $\sigma_t(c)$ maior será o estado da aprendizaxe para a clase $c$ no instante $t$. Tras normalizar $\sigma_t(c)$ para obter valores entre 0 e 1,

multiplícase este valor por un limiar fixo $\tau$ para obter o limiar para a clase $c$ no instante $t$, $\tau_t(c)$:

$$\beta_t(c) = \frac{\sigma_t(c)}{\max\limits_{c}(\sigma_t(c))} \qquad \tau_t(c) = \beta_t(c)\tau \tag{3.22}$$

Así, a clase con mellor estado da aprendizaxe no instante $t$, terá limiar $\tau_t(c) = \tau$ e o resto de clases terán un limiar inferior, que será máis pequeno canto peor sexa o estado da aprendizaxe de dita clase. Por último, como as imaxes se procesan por lotes, as imaxes que se teñen en conta para obter o limiar dinámico vanse engadindo progresivamente en cada lote a medida que avanza o adestramento, en lugar de utilizar as $N$ imaxes dende o principio.

### 3.4.2. Adaptación de CPL para detección

Nun primeiro momento, poderiamos pensar en empregar directamente a estratexia anterior para cada unha das caixas de cada imaxe procesada por un detector de obxectos, pois cada caixa encerra un obxecto, que se pasa a un clasificador. Non obstante, existen varios motivos polos cales non se pode empregar este método directamente e é necesario adaptar CPL para o noso detector de obxectos.

1. Nun clasificador, o número de imaxes non etiquetadas $N$ é fixo e coñécese de antemán. Non obstante, nun detector, o número de obxectos pertencentes a cada clase presentes nunha imaxe como resultado dunha predición do *Teacher* pode variar ao longo do adestramento. Por exemplo, nunha época o *Teacher* pode predicir que unha imaxe contén 7 coches e na seguinte época predicir que contén 4 coches, 1 camión e 3 trens. Ademais, nun clasificador as imaxes correspondentes a cada obxecto non varían, mentres que nun detector o contido das caixas varía ao longo do adestramento.

2. Como se comentou, os valores $\sigma_t(c)$ estiman o estado da aprendizaxe dunha clase cando o número de imaxes de cada clase está balanceado. Non obstante, en detección de obxectos é moi común que exista un gran desbalanceo entre detecciónss pertencentes a distintas clases. Por exemplo, no conxunto de datos COCO[42] o número de detecciónss pertencentes á clase "persoa" é tres ordes de magnitude maior que o número de detecciónss pertencentes á clase "secador de pelo". Así, se empregamos directamente a Ecuación (3.22), as clases maioritarias terán limiares próximos a $\tau$ e as clases minoritarias terán limiares próximos a 0, o cal non reflicte o estado da aprendizaxe de cada clase.

Para solucionar estas limitacións e obter unha estratexia de limiar flexible para o noso detector de obxectos propóñense as seguintes modificacións:

1. Para cada imaxe non etiquetada, almacenaranse as deteccióǹs que se empregaron para adestrar o *Student* a última vez que se utilizou esa imaxe. Así, cando esta imaxe se volva a utilizar, elimínase a información asociada á última vez que se usou, é dicir, as deteccióǹs antigas xa non se teñen en conta no cálculo do limiar dinámico e son substituidas polas novas deteccióǹs.

2. Os valores $\sigma_t(c)$ calcularanse "balanceando" artificialmente o conxunto de deteccióǹs de cada clase. Para conseguir isto, o valor $\sigma_t(c)$ da clase maioritaria permanecerá igual ca estaba e o resto de $\sigma_t(c)$ escálanse por un número que será maior cantas menos deteccióǹs haxa da clase $c$. Por exemplo, se "persoa" é a clase maioritaria e existen a metade de deteccióǹs de "coche" ca de "persoa", o $\sigma_t(c)$ asociado a "coche" multiplicarase por 2. O número de deteccióǹs de cada clase presente nas imaxes non etiquetadas é un valor descoñecido, polo que non podemos empregalo para obter os novos $\sigma_t(c)$. Deste modo, aproximaremos este valor empregando as deteccióǹs presentes nos datos etiquetados. Así, en primeiro lugar, obtense para cada clase $c$ un valor $\gamma(c)$, que chamaremos **factor de balanceo**

$$\gamma(c) = \frac{N_c}{\max\limits_c(N_c)} \tag{3.23}$$

onde $N_c$ é o número de deteccióǹs pertencentes á clase c no conxunto de imaxes etiquetadas. En segundo lugar, en cada instante de tempo obtense para cada clase o novo $\sigma_t(c) \leftarrow \sigma_t(c) \cdot \gamma(c)$ e emprégase (3.22) para calcular o limiar dinámico $\tau_t(c)$ empregando os novos $\sigma_t(c)$.

O Algoritmo 1 detalla o adestramento dun detector *Teacher-Student* empregando este novo limiar dinámico.

Finalmente, hai que ter en conta que ao principio do adestramento empréganse moi poucas deteccióǹs para calcular $\sigma_t(c)$ e $\beta_t(c)$ e nalgún caso haberá clases para as que non exista aínda ningunha detección. Por este motivo, a estimación do estado da aprendizaxe pode non ser fiable nas primeiras iteracións do adestramento. Para solucionar este problema, [40] introduce unha **etapa de quentamento** (*warm-up*), na que $\beta_t(c)$ non se calcula empregando a Ecuación (3.22) cando dominan os datos sen utilizar. Así, implementaremos unha etapa de quentamento análoga para o noso detector.

Para iso, en primeiro lugar, **estimamos o número de deteccióǹs totais nos datos sen etiquetar empregando os datos etiquetados** (contamos o número de obxectos medio por imaxe nas imaxes etiquetadas e multiplicamos polo número de imaxes sen etiquetar). Deste modo, en calquera punto do adestramento, podemos estimar cantos obxectos aínda non se utilizaron. Seguindo a proposta de [40], cando $\max_c \sigma(c)$ sexa menor que o número de obxectos sen usar,

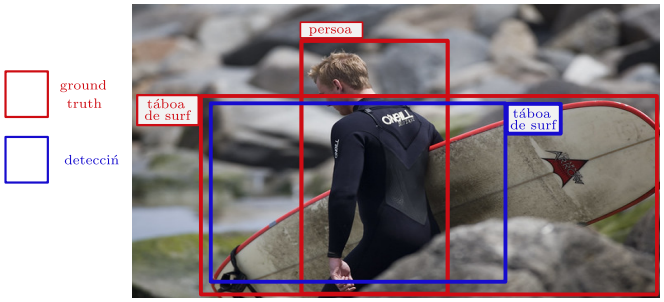



Figura 3.6: Problema na asignación de etiquetas clásica. A detección da imaxe ten un solapamento maior co *ground truth* de "persoa", polo que un método clásico de asignación de etiquetas asociará a detección a este *ground truth*. Non obstante, dado que o modelo predí "táboa de surf" para a detección, o máis razoable sería que se asignara ao *ground truth* de "táboa de surf".

empregarase a seguinte ecuación para calcular $\beta_t(c)$:

$$\beta_t(c) = \frac{\sigma_t(c)}{\max\{\max_c(\sigma_t(c)), N\}} \tag{3.24}$$

onde $N$ é o número estimado de obxectos sen utilizar.

## 3.5. Asignación de etiquetas óptima

No adestramento dun detector de obxectos é necesario asociar a cada detección un *ground truth* para calcular a función de perda. Isto denomínase **asignación de etiquetas**. Específicamente en Faster R-CNN [4], a asignación de etiquetas consiste en asociar cada proposta da RPN a un dos *ground truth* da imaxe, ou considerala fondo. A estratexia clásica para realizar esta asignación consiste en asociar cada proposta ao *ground truth* co que teña un maior IoU, sempre e cando se supere un IoU mínimo e, se non se supera este solapamento mínimo, considerala fondo. Deste modo, a asignación de etiquetas realízase unicamente empregando a regresión das deteccións, e non se ten en conta a súa clasificación. Isto pode non ser o máis adecuado cando unha imaxe contén obxectos de categorías diferentes moi cercanos entre si, tal e como se mostra na Figura 3.6. Por este motivo, xorden novas propostas para realizar a asignación de etiquetas tendo en conta o contexto global da imaxe e non só o solapamento das deteccións con cada *ground truth*.

### 3.5.1. SimOTA

Unha das melloras que implementaremos para o noso detector é substituir a estratexia de asignación de etiquetas clásica empregada por Unbiased Teacher v2 polo método **SimOTA** proposto en [41]. SimOTA é unha aproximación do

**Algoritmo 1** Algoritmo para adestrar un detector *Teacher-Student* con limiar flexible.

**Entrada:** Un conxunto de imaxes etiquetadas $\{(x_i^s, y_i^s)\}_{i=1}^{N_s}$, un conxunto de imaxes non etiquetadas $\{x_i^u\}_{i=1}^{N_u}$, un limiar fixo $\tau$, tamaño de Batch $B$

Para cada clase $c$, calcular o seu factor de balanceo $\gamma(c)$ empregando (3.23)
Inicializar $\tau(c) = 0 \quad \forall c$
Inicializar $pasanLimiar(c) = 0 \quad \forall c$
**Mentres** non se alcance o máximo de iteracións **facer**
  Inicializar a lista de pseudo-etiquetas para o *Student* $l_{pseudo} = \{\}$
  Inicializar as predicións de todas as imaxes non etiquetadas $\{y_i^u\}_{i=1}^{N_u}$ a unha lista vacía $y_i^u = \{\}$, indicando que aínda non foron utilizadas.
  **Para** $b = 1$ **ata** $B$ **facer**
    Obter as $N_{i_b}$ predicións do *Teacher* para a imaxe non etiquetada $x_{i_b}^u$, $\{d_j\}_{j=1}^{N_{i_b}}$, con $d_j = (b_j, c_j, s_j)$, onde $b_j$ son as coordenadas da caixa, $c_j$ a clase predita e $s_j$ a confianza da clasificación.
    **Se** $y_{i_b}^u$ non é unha lista vacía **entón**
      **Para** $d_k = (b_k, c_k, s_k)$ **en** $y_{i_b}^u$ **facer**
        $pasanLimiar(c_k) \leftarrow pasanLimiar(c_k) - 1$
      **Fin para**
      $y_{i_b}^u = \{\}$
    **Fin se**
    **Para** $j = 1$ **ata** $N_{i_b}$ **facer**
      **Se** $s_j > \tau(c_j)$ **entón**
        Engadir a predición $d_j$ a $y_{i_b}^u$
        $pasanLimiar(c_j) \leftarrow pasanLimiar(c_j) + 1$
      **Fin se**
    **Fin para**
    Engadir $(x_{i_b}^u, y_{i_b}^u)$ a $l_{pseudo}$
  **Fin para**
  Facer unha iteración de adestramento supervisado do *Student* co *batch* de imaxes etiquetadas.
  Facer unha iteración de adestramento semi-supervisado do *Student* coas imaxes e etiquetas de $l_{pseudo}$.
  Actualizar o *Teacher* mediante EMA.
  **Para todos** c **facer**
    Calcular o número de instancias balanceadas $\sigma(c) = pasanLimiar(c) \cdot \gamma(c)$
  **Fin para**
  **Para todos** c **facer**
    Actualizar $\tau(c)$ empregando (3.22)
  **Fin para**
**Fin mentres**

método **OTA** (do inglés, ***Optimal Transport Assignment***, asignación de transporte óptimo) proposto en [3]. Empregaremos esta aproximación en lugar do método OTA orixinal porque é computacionalmente moito menos custosa.

**OTA** plantexa a asignación de predicións a *ground truths* como un **problema do transporte**, que é un problema clásico de programación lineal. Cada *ground truth* vese como un punto de oferta que contén $k$ etiquetas positivas, e cada predición é un punto de demanda que necesita unha unidade de etiqueta positiva. Así, o custo $c_{ij}$ de transportar unha unidade de etiqueta positiva do *ground truth* $i$ á predición $j$ obtense como:

$$c_{ij} = L_{ij}^{clas} + \gamma L_{ij}^{reg}, \tag{3.25}$$

onde $\gamma$ é un hiperparámetro, e $L_{ij}^{clas}$ e $L_{ij}^{reg}$ son as perdas de clasificación e regresión entre o *ground truth* $i$ e a predición $j$. Ademais, establécese un punto de demanda adicional para asignar "fondo" a todas as propostas que non se asignaron a ningún *ground truth*. Así, os custos de cada predición con ese punto de demanda obtéñense coa función de perda de clasificar dita predición como "fondo". Deste modo, resólvese o problema do transporte plantexado e asígnase cada predición a un *ground truth*.

**Algoritmo 2** Algoritmo SimOTA de asignación de etiquetas.

**Entrada:** O conxunto de predicións $\{p_j\}_{j=1}^{N_p}$ e *ground truths* $\{g_i\}_{i=1}^{N_g}$ presentes nunha imaxe, hiperparámetros $r$ e $q$.

1: **Para todos** $g_i$ **facer**
2: Obter unha lista $\{p_j^{g_i}\}_j$ coas predicións cuxo centro está a unha distancia menor ca $r$ do centro de $g_i$.
3: **Fin para**
4: **Para todos** $g_i$, $p_j^{g_i}$ **facer**
5: Calcular o custos $c_{ij}$ de asignar a predición $p_j^{g_i}$ ao *ground truth* $g_i$ empregando a Ecuación (3.25).
6: **Fin para**
7: **Para todos** $g_i$ **facer**
8: Elixir de $\{p_j^{g_i}\}_j$ as $q$ predicións con maior IoU con $g_i$: $\{p_{j_k}\}_{k=1}^{q}$.
9: $k_{g_i} = \sum_{k=1}^{q} \text{IoU}(p_{j_k}, g_i)$, onde $\text{IoU}(p_{j_k}, g_i)$ é o IoU entre a predición $p_{j_k}$ e o *ground truth* $g_i$.
10: **Fin para**
11: **Para todos** $g_i$ **facer**
12: Asignar a $g_i$ as $k_{g_i}$ predicións de $\{p_j^{g_i}\}_j$ con menor custo $c_{ij}$.
13: **Fin para**
14: As predicións de $\{p_j\}_{j=1}^{N_p}$ non asignadas a ningún $g_i$, asígnanse como *fondo*: $g_{fondo}$.
15: **Retornar** Lista coas asignacións predición-*ground truth* $\{(p_j, g_i)\}_{j=1}^{N_p}$, onde $i = 1, \ldots, N_g$ ou $i = fondo$.

Tal e como se afirma en [3], resolver este problema do transporte aumenta o tempo de adestramento nun 20 %. Así, [41] propón unha aproximación de OTA computacionalmente menos custosa: SimOTA. Esta é a estratexia que incorporaremos ao noso detector, polo que a describimos en detalle a continuación.

O Algoritmo 2 mostra a asignación de etiquetas realizada por **SimOTA** para unha imaxe. En primeiro lugar, SimOTA só considera para cada *ground truth* aquelas predicións con centro a unha distancia máxima do centro do *ground truth* (paso 2). Unha vez obtidas estas predicións, calcúlanse os custos para todos os pares predición - *ground truth* empregando a Ecuación (3.25) (paso 5). Deste modo, evítase asignar predicións a *ground truths* que están demasiado lonxe e afórrase o tempo que levaría calcular todos os custos. A continuación, para cada *ground truth* selecciónanse as $k$ predicións con menor custo. Así, estas serán as predicións asignadas a dito *ground truth* (paso 12) e o resto de predicións asignaranse a "fondo" (paso 14). O valor de $k$ para cada *ground truth* obtense como se indica no paso 7 empregando o hiperparámetro $q$.

### 3.5.2. Adaptación de SimOTA a detectores de dúas etapas

SimOTA é un método proposto para a familia de detectores YOLO[15], que son detectores dunha etapa. Neste traballo estase a empregar Faster R-CNN, que é un detector de dúas etapas. Nos detectores dunha etapa, non se obteñen propostas de detección, polo que SimOTA aplícase directamente sobre *anchors*. Pola contra, en Faster R-CNN a RPN xera unha serie de propostas que se asignan a fondo ou obxecto. Deste modo, SimOTA debe aplicarse sobre as propostas da RPN e non sobre os *anchors*. Isto fai que sexa necesario adaptar lixeiramente o algoritmo para empregalo en Faster R-CNN. Ata onde sabemos, non existen polo momento adaptacións de OTA a detectores de dúas etapas. Así, propoñemos a seguinte adaptación:

- A RPN obtén un número fixo de propostas que se asignan a algún *ground truth* (obxecto) ou a fondo. Soamente se aplicará SimOTA sobre aquelas propostas que na RPN se asignaron a obxecto.
- As propostas que na RPN se asignaron a fondo déixanse como están e teñen *ground truth* asociado "fondo".
- Aplícase SimOTA a aquelas propostas que na RPN se asignaron a algún *ground truth*, e obtense unha nova asignación proposta-*ground truth*. Así, no Algoritmo 2, o conxunto de predicións de entrada corresponderase co conxunto de propostas asignadas a "obxecto" pola RPN.
- Aquelas propostas que quedaron sen asignar con SimOTA, non se pasan á cabeceira R-CNN e, por tanto, non se utilizan para adestrar o detector. Así, as propostas que o Algoritmo 2 devolva con asignación $g_{fondo}$ non se empregarán para adestrar o detector.

# Capítulo 4

# Materiais

## 4.1. Repositorios de GitHub

Este traballo parte como base do detector **Unbiased Teacher v2** [1], cuxa implementación se atopa publicada no seguinte repositorio de GitHub: `https://github.com/facebookresearch/unbiased-teacher-v2`. Así, partiuse do código implementado neste repositorio e fixéronse cambios sobre o mesmo.

Por outra parte, para engadir os módulos GDL e PCB empregouse a implementación publicada en GitHub do detector **DeFRCN** [2] : `https://github.com/er-muyue/DeFRCN` e para engadir SimOTA empregouse parte da implementación de **YOLOX** [41]: `https://github.com/Megvii-BaseDetection/YOLOX.git`. En ambos casos foi necesario modificar a implementación para adaptala á implementación de Unbiased Teacher v2 da que partiamos.

## 4.2. Servidores de computación

Para executar os modelos empregados neste traballo é imprescindible contar con GPUs. Así, todas as probas realizadas foron levadas a cabo ben nos servidores **GPGPU** `ctgpgpu6` e `ctgpgpu11` do Centro Singular de Tecnoloxías Intelixentes da Universidade de Santiago de Compostela (CiTIUS), ou ben no servidor **FINISTERRAE-III** do Centro de Supercomputación de Galicia (CESGA). A continuación, especificamos as características máis importantes destes servidores:

- `ctgpgpu6`: servidor **SIE LADON 4214** [45] con dous procesadores **Intel Xeon Silver 4212** [46] con 192 GB de memoria RAM. Conta coas seguintes GPUs: unha **Nvidia Quadro P6000** [47] de 24GB, unha **Nvidia Quadro RTX 8000** [48] de 48 GB e dúas **Nvidia A100** [49] de 80 GB . Emprega o sistema operativo **Centos 7.7** [50], que é unha distribución de Linux.

- `ctgpgpu11`: servidor **Gybabyte G482-Z54** [51] con dous procesadores **AMD EPYC 7413** [52] con 256 GB de memoria RAM. Conta coas se-

guintes GPUs: cinco **NVIDIA A100** [49] de 80 GB . Emprega o sistema operativo **AlmaLinux 9.1** [53], que é unha distribución de Linux.

- FINISTERRAE-III: empregouse o grupo de nodos denominado `a100`. Cada nodo conta con dous procesadores **Intel Xeon Ice Lake 8352Y**[54] con 32 cores cada un, 256 GB de memoria RAM e dúas GPUs **Nvidia A100**[49].

## 4.3. Linguaxes de programación

O código de Unbiased Teacher v2 e o do resto de detectores empregados para engadir modificacións, están implementados en Python. Así, elixiuse **Python** [55] como linguaxe de programación. Ademais, empregáronse numerosas **bibilotecas** de Python, entre as que destacamos: **NumPy** [56], que proporciona ferramentas para o manexo de vectores e matrices e múltiples de funcións matemáticas; **PyTorch** [57], que proporciona operacións de aprendizaxe profunda e permite manexar de eficientemente tensores multidimensionais mediante o uso de GPUs; **Matplotlib** [58], que permite crear gráficas e outras visualizacións; **OpenCV** [59], que proporciona algoritmos e ferramentas para tarefas de visión por computador. Ademais, Unbiased Teacher v2 parte da implementación de **Detectron2** [60], unha biblioteca que implementa os algoritmos máis comúns en detección de obxectos empregando PyTorch, como, por exemplo, Faster-RCNN [4].

Por outra parte, para automatizar os experimentos usáronse *scripts* de **Bash**.

Finalmente, é importante destacar o uso de **CUDA** (*Compute Unified Device Architecture*)[61], unha plataforma de computación paralela e un modelo de programación, que permite utilizar **GPUs** para realizar cálculos de propósito xeral. PyTorch ten soporte para CUDA, o cal permite que as operacións tensoriais e os algoritmos de PyTorch se executan moito máis eficientemente Para o desenvolvemento deste traballo non se empregou CUDA directamente, pois aproveitouse a xestión das GPUs xa implementada no detector Unbiased Teacher v2.

## 4.4. Outras ferramentas

Ademais do xa comentado, tamén se empregaron as seguintes ferramentas:

- **Docker**[62]: as execucións e probas realizadas leváronse a cabo empregando contedores de Docker. Os contedores son contornos lixeiros que inclúen as bibliotecas e dependencias necesarias para que un programa se execute. Isto permitiu eliminar problemas de compatibilidade entre programas.
- **Visual Studio Code**[63]: editor de código cuxas extensións facilitan enormenente tarefas como execución en servidores remotos, execución en contedores Docker, edición e depuración de código en Python, etc. Neste traballo, empregouse para toda a implementación do código.

# Capítulo 5

# Probas

## 5.1. Deseño experimental

### 5.1.1. Métrica e conxunto de datos para detección de obxectos

Para a realización das probas, empregouse o conxunto de datos para detección de obxectos **MS-COCO** [42]. COCO é un conxunto de datos moi empregado para avaliar modelos de detección e segmentación de obxectos. Conta con máis de 200.000 imaxes anotadas, con obxectos pertencentes a 80 categorías distintas. A distribución das detecciónss por categorías presentes en COCO móstrase no Cadro 5.1. Este conxunto de datos divídese en dous subconxuntos: adestramento (*training*) e validación (*validation*). Concretamente, todas as probas están realizadas cos conxuntos de adestramento e validación de COCO 2017. Ao traballar nun contexto semi-supervisado, o conxunto de datos de adestramento divídense en datos etiquetados e datos sen etiquetar. Así, soamente se empregará unha pequena porcentaxe das etiquetas presentes no conxunto de datos. O habitual nesta área é considerar diferentes porcentaxes de datos etiquetados para avaliar o modelo e comparalo con outros. Por exemplo, Unbiased Teacher v2 proba o detector cun 0.5 %, 1 %, 2 %, 5 % e 10 % dos datos etiquetados. Así, sería desexable avaliar as melloras propostas neste traballo coas mesmas porcentaxes de datos etiquetados. Non obstante, estamos a traballar cun modelo que tarda moito tempo en realizar un adestramento completo (entre 3 e 5 días, en función das GPUs empregadas). Isto supón unha gran limitación á hora de realizar execucións, facendo inviable obter esa cantidade de resultados. Por este motivo, decidiuse avaliar as melloras propostas empregando un **2 % de supervisión**, é dicir, utilízanse como datos de adestramento un 2 % das imaxes do conxunto de datos etiquetadas e o resto das imaxes sen etiquetar. A separación entre imaxes etiquetadas e sen etiquetar realízase de maneira aleatoria.

Para avaliar a calidade das detecciónss obtidas polo modelo, emprégase o que

| | | | | | |
|---|---|---|---|---|---|
| **persoa** | 257.253 | **coche** | 43.533 | **cadeira** | 38.073 |
| **libro** | 24.077 | **botella** | 24.070 | **cunca** | 20.574 |
| **mesa de comedor** | 15.695 | **bol** | 14.323 | **semáforo** | 12.842 |
| **bolso** | 12.342 | **paraugas** | 11.265 | **barco** | 10.576 |
| **paxaro** | 10.542 | **camión** | 9.970 | **banco** | 9.820 |
| **ovella** | 9.223 | **plátano** | 9.195 | **papaventos** | 8.802 |
| **mochila** | 8.714 | **moto** | 8.654 | **planta en maceta** | 8.631 |
| **vaca** | 8.014 | **copa de viño** | 7.839 | **coitelo** | 7.760 |
| **cenoria** | 7.758 | **brócoli** | 7.261 | **bicicleta** | 7.056 |
| **rosquilla** | 7.005 | **esquís** | 6.623 | **floreiro** | 6.577 |
| **cabalo** | 6.567 | **gravata** | 6.448 | **teléfono móbil** | 6.422 |
| **reloxo** | 6.320 | **laranxa** | 6.302 | **balón** | 6.299 |
| **torta** | 6.296 | **culler** | 6.159 | **maleta** | 6.112 |
| **táboa de surf** | 6.095 | **autobús** | 6.061 | **pizza** | 5.807 |
| **tv** | 5.803 | **sofá** | 5.779 | **mazá** | 5.776 |
| **mando a distancia** | 5.700 | **lavabo** | 5.609 | **patinete** | 5.536 |
| **can** | 5.500 | **elefante** | 5.484 | **garfo** | 5.474 |
| **cebra** | 5.269 | **avión** | 5.129 | **xirafa** | 5.128 |
| **portátil** | 4.960 | **raqueta de tenis** | 4.807 | **gato** | 4.766 |
| **oso de peluche** | 4.729 | **tren** | 4.570 | **bocadillo** | 4.356 |
| **cama** | 4.192 | **váter** | 4.149 | **luva de béisbol** | 3.747 |
| **forno** | 3.334 | **bate de béisbol** | 3.273 | **hot dog** | 2.884 |
| **teclado** | 2.854 | **frisbee** | 2.681 | **táboa de snowboard** | 2.681 |
| **neveira** | 2.634 | **rato** | 2.261 | **sinal de stop** | 1.983 |
| **cepillo de dentes** | 1.945 | **boca de incendios** | 1.865 | **microondas** | 1.672 |
| **tesoiras** | 1.464 | **oso** | 1.294 | **parquímetro** | 1.283 |
| **tostadora** | 225 | **secador de pelo** | 198 | | |

Cadro 5.1: Número de detecciónes de cada categoría presentes no dataset COCO, ordeadas de máis a menos abundantes.

se coñece como **IoU** (*do inglés,* ***Intersection over Union***, intersección sobre a unión). O IoU mide o nivel de solapamento entre as caixas obtidas polo detector e as caixas reais anotadas na imaxe. Cálculase como o cociente entre a área da intersección de ambas caixas e a área da súa unión. Deste modo o IoU é un valor entre 0 e 1, onde 1 representa una coincidencia exacta e 0 que as caixas non se intersecan. Como é habitual en Machine Learning en xeral, precisamos coñecer o número de Falsos Negativos (FN), Falsos Positivos (FP) e Verdadeiros Positivos (VP) para definir a **precisión** (***precision***) e **sensibilidade** (***recall***) dun modelo. Estes conceptos aplicados á detección de obxectos ilústranse na Figura 5.1.

$$\text{precision} = \frac{VP}{VP + FP} \qquad \text{recall} = \frac{VP}{VP + FN} \tag{5.1}$$

Así, para cada clase considerada obtense un valor de precisión e sensibilidade. Ademais, hai que ter en conta que para obter estes valores é necesario definir un nivel de solapamento mínimo para considerar que unha detección se corresponde

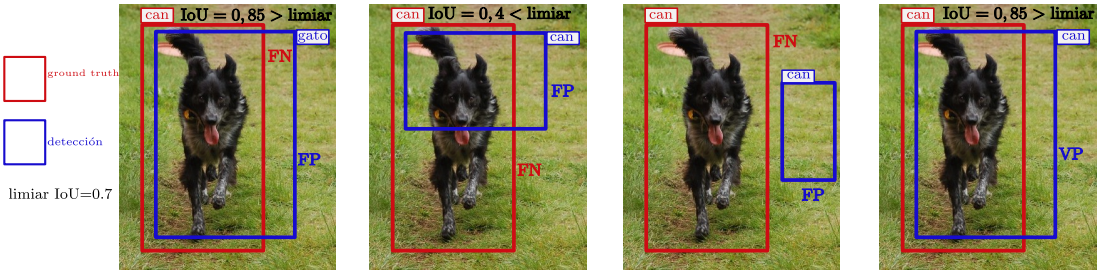



Figura 5.1: Exemplo de Verdadeiros Positivos (VP), Falsos Positivos (FP) e Falsos Negativos (FN) para a clase "can".

con un *ground truth*. Así, a cada *ground truth* asóciaselle a detección con maior IoU, sempre e cando este valor supere un certo limiar.

Por outra parte, soamente se considerarán as deteccións que superen un certo limiar de confianza na clasificación. Así, a medida que aumente este limiar, en xeral, aumentará a precisión e diminuirá a sensibilidade. Para eliminar a dependencia da elección deste limiar na medida do rendemento dun modelo, emprégase o que se coñece como **curva de precisión-*recall***, que representa os valores de precisión e *recall* dunha clase concreta para diferentes limiares de confianza. Así, definimos o que se coñece como **AP** (do inglés, ***Average Precision***, precisión media) como a área encerrada baixo a curva de precisión-*recall*.

$$AP = \int_0^1 p(r)dr, \text{ onde } p \text{ é a curva de precisión-}recall \tag{5.2}$$

O máis habitual é aproximar a curva de precisión-*recall* mediante interpolación lineal. Así, discretízase a curva nun número determinado de valores de *recall* ($\{r_i\}_i$) e obtense a precisión $\{p_i = p(r_i)\}_i$ para cada $r_i$. Deste modo, o AP obtense facendo a media dos $p_i$. Neste traballo seguirase o estándar definido por COCO, que consiste en realizar a interpolación en 101 puntos de *recall* (valores entre 0 e 1 con paso de 0,01).

Como xa se explicou, unha detección asóciase a un *ground truth* se ten un IoU maior a un certo limiar. Deste modo, en función do limiar establecido, teranse distintos valores de AP. En COCO, o estándar é calcular o **AP50** (IoU con limiar de 0,5), o **AP75** (IoU con limiar de 0,75) e o **AP** (media do AP para limiares de IoU entre 0,5 e 0,95 con paso de 0,05). Este último AP emprégase como métrica principal para determinar o rendemento dun detector nunha determinada clase. Outras medidas comúns son o **APs** (AP para obxectos pequenos), o **APm** (AP para obxectos medianos) e o **APl** (AP para obxectos grandes)[1]. Por último, dado que estes valores de AP se calculan para cada clase, tamén se obtén o **mAP**, a media dos AP entre todas as clases consideradas. Como é habitual en detección

[1]Un obxecto considérase pequeno se a área da caixa é inferior a $32^2$, grande se é superior a $96^2$ e mediano se está entre estes valores.

de obxectos, denotaremos o mAP simplemente por AP para simplificar, sempre que isto non xere confusión.

### 5.1.2. Configuración das probas e hiperparámetros

A continuación, detallaremos as características específicas do detector empregado e os hiperparámetros seleccionados. Co obxectivo de que os nosos resultados sexan comparables cos resultados presentados para Unbiased Teacher v2 en [1], empregaremos as mesmas condicións e hiperparámetros sempre que sexa posible.

- Pártese dun *backbone* preadestrado con ImageNet [44], cuxa arquitectura é unha FPN [12].
- Empregaremos un tamaño de mini-lote (*mini-batch*) de 32 imaxes etiquetadas e 32 sen etiquetar durante a Aprendizaxe Mutua. Durante a etapa de Burn-In, o tamaño de mini-lote será de 64 imaxes.
- Utilizaremos unha taxa de aprendizaxe (*learning rate*) de 0,01.
- Establecemos o número máximo de iteracións en 180.000, aínda que se parará o adestramento antes se este se estanca.
- A etapa de Burn-In durará 2.000 iteracións e o resto do adestramento será de Aprendizaxe Mutua. Como explicaremos máis adiante, decidiuse aumentar a 3.000 iteracións de Burn-In no experimento con SimOTA 5.6.
- O limiar de confianza para as prediccióǹs do *Teacher* será 0,7, é dicir, para que unha detección se empregue para adestrar o *Student* esta debe ter unha confianza na clasificación maior a 0,7. Este valor non aplica no caso de usar limiar flexible.
- O parámetro $\alpha$ usado na Ecuación (3.16) para aplicar EMA será $\alpha = 0{,}9996$.
- Os hiperparámetros da Ecuación (3.15) para limitar as incertidumes do *Teacher* e do *Student* serán $\sigma = 0{,}1$ e $\sigma_s = 0{,}5$.

## 5.2. Experimento 1: Proba base

En primeiro lugar, realizouse unha proba inicial sen realizar ningunha modificación no detector orixinal Unbiased Teacher v2. Deste modo, replicáronse os resultados presentados en [1] e puidéronse analizar posibles problemas. No Cadro 5.2 móstranse os resultados para o detector sen modificacións presentados en [1].

Ademais do rendemento global, resulta interesante mostrar as **grandes diferenzas no AP obtido para diferentes clases**. O Cadro 5.2 mostra que o

| | AP | APs | APm | APl |
|---|---|---|---|---|
| **Unbiased Teacher v2** | 28,37 ± 0,03 | 13,965 | 30,64 | 39,25 |
| **Unbiased Teacher v2 + GDL** | 28,70 ± 0,125 | 13,22 ± 0,056 | 31,31 ± 0,169 | 39,95 ± 0,216 |
| **Unbiased Teacher v2 +GDL+ Limiar flexible** | 28,67 ± 0,04 | 12,83 ± 0,092 | 31,11 ± 0,17 | 39,48 ± 0,396 |
| **Unbiased Teacher v2 +GDL+SimOTA** | 26.77 | 12.54 | 29.32 | 36.84 |

Cadro 5.2: Resultados para o detector coas diferentes melloras propostas. Móstrase a media e a desviación típica das execucións realizadas. Os resultados sen desviación típica débense a que só se realizou unha execución.

rendemento do detector é moito **peor nos obxectos pequenos ca nos obxectos grandes**, pois o APs é un terzo do APl. A Figura 5.2 mostra o AP para as clases con mellores e peores resultados, onde se reflicte claramente esta diferenza entre obxectos grandes e pequenos.

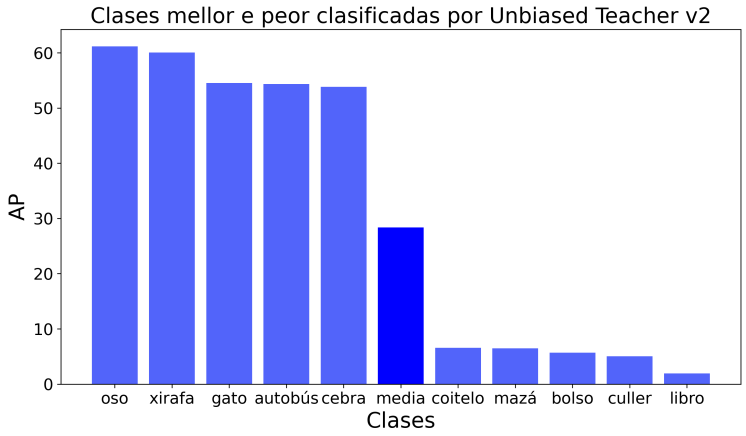



Figura 5.2: Clases con mellores e peores resultados no detector Unbiased Teacher v2 sen modificacións.

## 5.3. Experimento 2: GDL

O segundo experimento levado a cabo consiste en engadir os módulos *Gradient Decouple Layer* (GDL) propostos para o detector *few-shot* DeFRCN en [2], tal e como se explicou na sección 3.3. Así, en primeiro lugar, foi necesario determinar os valores dos hiperparámetros $\lambda_{rpn}$ e $\lambda_{rcnn}$ da Ecuación (3.3.1). Para isto, fixéronse unhas pequenas probas previas cun 20 % dos datos dispoñibles para determinar os valores máis adecuados para estes hiperparámetros nun tempo razoable. O resultado destas probas foi establecer $\lambda_{rpn} = 0$ e $\lambda_{rcnn} = 1$. Isto implica que os

pesos do *backbone* só se actualizan empregando os pesos da cabeceira R-CNN, tal e como sucede co detector DeFRCN.

O Cadro 5.2 mostra os resultados obtidos para este novo detector. Os resultados da táboa correspóndense coa media e a desviación típica das 5 execucións que se realizaron. Así, pódese observar que o rendemento total do modelo **mellora lixeiramente cos módulos GDL**, cunha mellora media dun 1,2 % . Esta mellora dase nos obxectos medianos e grandes, empeorando lixeiramente o rendemento nos obxectos pequenos.

Finalmente, dado que engadir módulos GDL mellorou o rendemento do detector, o resto de experimentos realizaranse a partir do detector Unbiased Teacher v2 engadindo os módulos GDL.

## 5.4. Experimento 3: PCB

O terceiro experimento levado a cabo consiste en engadir o ***Prototypical Calibration Block* (PCB)** proposto para DeFRCN, tal e como se explica na sección 3.3. Do mesmo xeito que en DeFRCN, empregouse ResNet 101 [7] como clasificador. Tras realizar unha serie de probas, variando o valor do hiperparámetro $\alpha$ da Ecuación (3.20), comprobouse que en todos os casos os resultados obtidos son moi inferiores aos do detector orixinal sen PCB. Así, **engadir o módulo PCB empeora o rendemento do detector**.

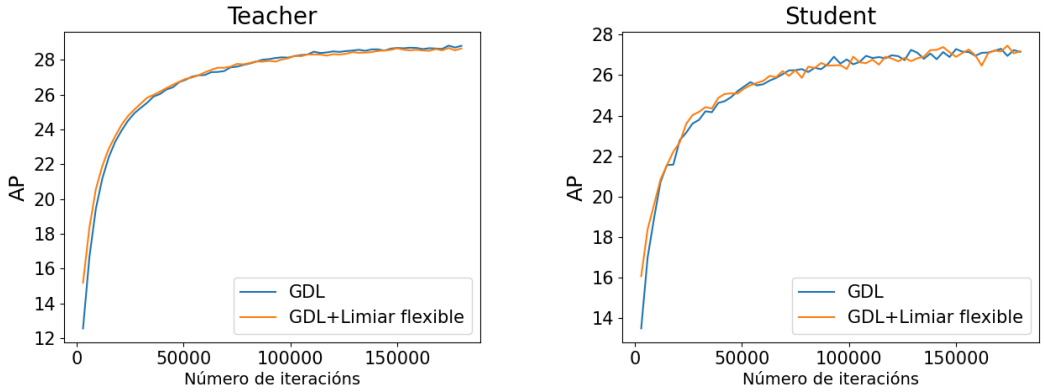
O módulo PCB está pensado para un escenario *few-shot*, onde as únicas instancias das clases novel dispoñibles son as que se utilizan para crear o banco de prototipos. Non obstante, nun contexto semi-supervisado disponse de moitas máis instancias, só que estas están sen etiquetar. Unha posible razón de por que engadir este módulo deteriora o rendemento do detector é que o PCB permanece invariante ao longo de todo o adestramento, polo que non aproveita en absoluto a mellora do rendemento que se produce durante a Aprendizaxe Mutua. Deste modo, nun punto do adestramento onde o modelo teña un rendemento aceptable grazas á estratexia de pseudo-etiquetado, é probable que as confianzas proporcionadas polo PCB sexan menos fiables que as proporcionadas polo *Teacher*, o cal daría lugar a unha perda de rendemento. De feito, en [2] faise un estudo da influenza do PCB no rendemento, no cal se determina que as melloras de engadir PCB son maiores para 10-*shot* que para 30-*shot*. Isto reforza a nosa hipótese de que PCB é especialmente útil cando se dispón de poucos datos. De todos modos, sería necesaria unha análise máis profunda para sacar conclusións sobre isto.

Dado que o detector non consegue mellorar engadindo o PCB, todos os demais experimentos se realizarán sen incluir este módulo.

## 5.5. Experimento 4: Limiar flexible

Partindo do detector Unbiased Teacher v2 con módulos GDL, o cuarto experimento realizado consiste en engadir o limiar flexible descrito na sección 3.4.

En primeiro lugar, implementouse esta modificación seguindo o Algoritmo 1. Para iso, primeiro foi necesario establecer o hiperparámetro $\tau$ da Ecuación (3.22), é dicir, o limiar fixo empregado para calcular os limiares dinámicos de cada clase. Tras realizar unhas pequenas probas cun 20 % dos datos dispoñibles, estableceuse $\tau = 0{,}8$. O Cadro 5.2 mostra os resultados obtidos para as dúas execucións que se realizaron. Observamos que esta implementación **non supón ningunha mellora** con respecto ao detector con GDL. A Figura 5.3 mostra a evolución ao longo dun adestramento do rendemento do detector con e sen limiar flexible. Observamos claramente que a evolución do AP do detector non amosa practicamente diferenzas cando se engade un limiar flexible.




(a) Evolución do AP do *Teacher* (b) Evolución do AP do *Student*

Figura 5.3: Comparación dos resultados da validación dos detectores Unbiased Teacher v2 + GDL con e sen limiar flexible ao longo dun adestramento.

Se analizamos o rendemento por clases, observouse que **as clases que máis empeoran co limiar flexible son as que teñen un AP máis alto**. Concretamente, na execución realizada, as clases cun AP inferior a 20 melloraron o seu AP unha media de 0,07 puntos ao engadir o limiar flexible, mentres que as clases con AP superior a 40 empeoran unha media de 0,85 puntos.

O Algoritmo 1 é unha adaptación á detección de obxectos do algoritmo de limiar flexible proposto en [40] para problemas de clasificación. A detección de obxectos é un problema máis complexo que a clasificación, e existen moitos motivos polos cales esta adaptación de limiar flexible podería non estar mellorando o noso detector, a pesar de que a proposta de [40] si que mellora os clasificadores semi-supervisados. Algúns dos posibles problemas son os seguintes:

1. Nun problema de clasificación está garantizado que todas imaxes conteñen un obxecto, independentemente do valor da confianza na clasificación da

pseudo-etiqueta do *Teacher*. Pola contra, nun problema de detección é posible que unha detección do *Teacher* non conteña ningún obxecto, especialmente se esta detección ten un nivel de confianza baixo. Así, resulta interesante plantexarse se, no caso de que o algoritmo de limiar dinámico estableza un limiar moi baixo (incluso 0) para algunha clase, se deberían empregar detecciónss con confianzas tan baixas para adestrar o *Student*.

2. Empregar un limiar dinámico dará lugar a que para algunhas clases pasen detecciónss cunha confianza máis baixa da que pasarían cun limiar fixo. Isto fará que pasen máis detecciónss e, por tanto, aumentará a probabilidade de que se empreguen para adestrar o *Student* detecciónss múltiples, isto é, detecciónss con diferentes niveis de confianza que realmente se refiren ao mesmo obxecto. Este problema non aparece nun problema de clasificación, pois nese contexto cada imaxe contén un único obxecto a clasificar.

3. Tal e como se mostra nas Ecuacións (3.21) e (3.22), a conta do número de instancias que pasaron o limiar dinámico emprégase como estimación do estado da aprendizaxe de cada clase. Non obstante, ao principio do adestramento esta estimación non é moi fiable: téñense en conta moi poucas imaxes e é probable que aínda non apareceran detecciónss para algunhas clases. Ademais podería dar lugar a que o limiar sexa moi baixo ao principio do adestramento para algunhas clases, o cal empeora as problemáticas 1 e 2. Este problema tamén sucede na clasificación de imaxes e é abordado por [40] engadindo unha etapa de quentamento (*warm-up*), na que o limiar dinámico se obtén doutro xeito cando o número de instancias empregadas dunha determinada clase é moi pequeno.

4. En liñas xerais, observouse que as clases que máis empeoran cando se emprega este limiar flexible son aquelas que teñen APs máis altos, que tenden a ter tamén limiares dinámicos máis altos. Unha posible causa disto é que o *Teacher* obteña moi poucas detecciónss con confianzas moi altas e que isto faga que se usen poucos exemplos para adestrar o *Student*. Ademais, é probable que os exemplos usados sexan sempre os mesmos.

5. Para as clases con limiares dinámicos altos, existirán detecciónss que non se empreguen para adestrar o *Student*, pero que superarían o limiar fixo establecido para o detector sen limiar flexible. Así, as rexións da imaxe onde se atopan estas detecciónss moi problemente conteñan un obxecto, pero o *Student* está aprendendo que son fondo. De novo, este problema non se dá nun contexto de clasificación.

A Figura 5.4 contén exemplos de detecciónss realizadas empregando limiar flexible que ilustran algúns dos problemas identificados.

Co obxectivo de intentar mitigar o efecto destes problemas, propuxéronse as seguintes modificacións para o detector con limiar flexible inicial:

(a) Varias etiquetas "mesa de comedor" para un só obxecto.

(b) Dúas etiquetas "tren" para un único obxecto.

(c) Etiqueta "paxaro" onde hai fondo.

Figura 5.4: Exemplos de detecciónss empregando limiar flexible. Ilustran algúns dos problemas que xorden con esta estratexia.

- **Establecemento dun limiar mínimo**: independentemente do limiar dinámico obtido para unha determinada clase, as detecciónss que non superen un limiar mínimo de confianza non serán empregadas para adestrar o *Student*. Isto reducirá a aparición dos problemas 1 e 2.
- **Aplicación de NMS antes de adestrar o *Student***: co obxectivo de reducir o número de detección múltiples descrito no problema 2, aplícase *Non-Maximum Suppression* (NMS) ás detecciónss do *Teacher* que pasaron o limiar dinámico.
- **Establecemento dun limiar máximo**: independentemente do limiar flexible obtido para unha determinada clase, as detecciónss que superen un determinado limiar de confianza empregaranse para adestrar o *Student*. Ademais, implementouse unha segunda versión desta idea: en lugar de empregar todas as detecciónss que pasen o limiar máximo, selecciónanse aleatoriamente $n$ detecciónss, onde $n$ é o número de detecciónss que pasaron o limiar dinámico. Isto reducirá a aparición do problema 4.
- **Iteracións iniciais sen limiar flexible**: en lugar de empregar o limiar dinámico dende o principio do adestramento, as primeiras iteracións da etapa de Aprendizaxe Mutua realízanse cun limiar fixo. Isto permite que a calidade das *pseudo-etiquetas* non sexa tan mala cando se introduce o limiar dinámico e que existan menos detecciónss que dean lugar ao problema 1. Ademais, durante estas primeiras iteracións cóntase o número de detecciónss de cada clase que pasan o limiar fixo, de modo que as primeiras iteracións con limiar flexible poden empregar esta información para obter unha mellor

estimación do estado da aprendizaxe de cada clase. Isto reduce o problema 3.

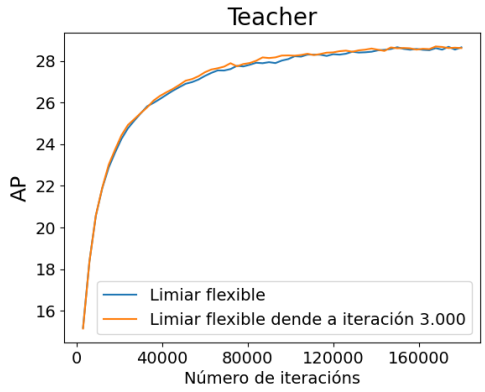
- **Etapa inicial de quentamento**: para mitigar o problema 3, adáptase a detección de obxectos a etapa de quentamento (*warm-up*) proposta en [40]. Deste modo, cando dominen os datos sen utilizar, non se empregará a Ecuación (3.22) para calcular o limiar de cada clase, se non que se empregará a Ecuación (3.24), tal e como se describiu na sección 3.4.

- **Empregar detecciónsns por enriba dun limiar fixo para adestrar a RPN e ignorar as súas rexións na R-CNN**: as detecciónsns que superan o limiar fixo de confianza para o detector sen limiar flexible seguramente conteñan un obxecto, aínda que a súa confianza non supere o limiar dinámico. Así, estas detecciónsns empregaranse para adestrar a RPN (que só distingue entre obxecto e fondo), pero non se empregarán para adestrar a cabeceira R-CNN. Ademais, as propostas da RPN que teñan como *ground truth* algunha destas detecciónsns, non se usarán na función de perda da cabeceira. Deste modo, as rexións próximas a estas detecciónsns ignoraranse na R-CNN e non aprenderá que son fondo. Isto permite resolver o problema 5. Fixéronse probas con limiar fixo 0,5 e 0,7.

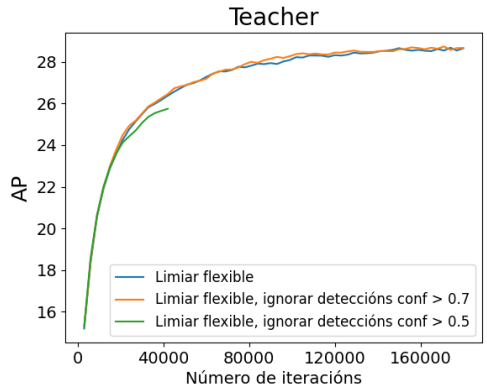
Moitas destas propostas de mellora requiren fixar algún hiperparámetro: limiar mínimo/máximo, limiar de IoU para aplicar NMS, etc. Hai que ter en conta que este detector tarda varios días en realizar un adestramento completo, polo que executar estas propostas probando diferentes hiperparámetros non é viable nun tempo razoable. Así, optouse por executar estas propostas e ir monitorizando continuamente se estas melloraban o modelo orixinal. Así, se nun punto do adestramento se observaba que o rendemento claramente non estaba mellorando, cortábase a execución. A Figura 5.5 mostra os resultados para algunhas das execucións realizadas. Só mostraremos os resultados para o *Teacher*, pois este é o modelo máis estable e con mellor rendemento.

Tal e como se ve na figura, **ningunha das propostas logrou mellorar os resultados orixinais**. Así é todo, exceptuando a etapa inicial de quentamento, ningún destes cambios parece empeorar tampouco o rendemento do detector. A continuación, daremos algunhas posibles razóns de por que estas propostas poden non estar mellorando o noso detector.

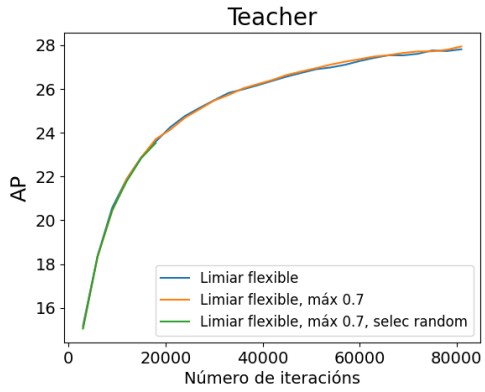
En primeiro lugar, Figura 5.6 mostra a evolución dos limiares flexibles de cada clase ao longo dun adestramento. Observamos que só dúas clases manteñen limiares baixos durante unha boa parte do adestramento, e que todas as demais teñen **limiares dinámicos superiores a 0,5 durante a maior parte do adestramento**. Isto fai que o establecemento dun limiar mínimo teña un impacto moi reducido no adestramento global. Ademais, redúcese a importancia dos problemas 1 e 2, o cal pode explicar que aplicar NMS non teña moito efecto nos resultados.



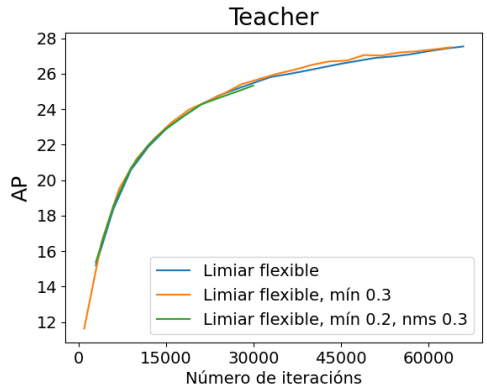

(a) 3.000 iteracións iniciais sen limiar flexible



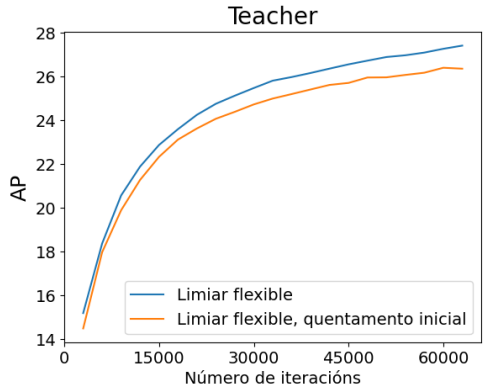

(b) Ignorar rexións con confianza $> 0{,}5$ e 0,7




(c) Fixar un limiar máximo




(d) Fixar un limiar mínimo e/ou NMS




(e) Etapa inicial de quentamento

Figura 5.5: Comparación dos resultados da validación do detector Unbiased Teacher v2 + GDL con limiar flexible, engadindo distintas modificacións. Todos os resultados se corresponden co modelo *Teacher*.

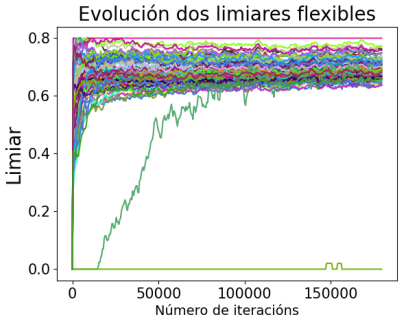



Figura 5.6: Evolución do limiar flexible de cada clase ao longo dun adestramento. Cada liña de cores representa unha clase distinta.

En segundo lugar, é posible que a razón de que as clases con mellor rendemento sexan as que empeoran máis usando limiar flexible non sexa a descrita no problema 4. Neste caso, establecer un limiar máximo non melloraría os resultados. Sería necesaria unha análise máis profunda para determinar por que as clases con APs máis altos non se ven beneficiadas en xeral polo uso dun limiar dinámico.

Por outra parte, para a etapa inicial de quentamento emprégase unha estimación do número de imaxes non usadas. É posible que esta estimación non sexa tan boa como debería e empeore o rendemento do detector.

Por último, analizouse o número de propostas que se ven afectadas polo último cambio suxerido, isto é, cantas rexións se empregan para adestrar a RPN pero son ignoradas na función de perda da R-CNN. Analizaremos isto no caso de empregar limiar fixo de 0,7, pois se se emprega 0,5 os resultados empeoran. Observouse que ao principio do adestramento a porcentaxe de propostas ignoradas si que é máis ou menos significativa, supoñendo entre un 5 e un 9 % do total de propostas nas primeiras 10.000 iteracións. Non obstante, esta porcentaxe vaise reducindo a medida que avanza o adestramento e a partir da iteración 30.000 as propostas ignoradas supoñen menos dun 0,5 % do total. Deste modo, o impacto deste cambio no adestramento global é moi limitado.

En conclusión, en vista das melloras descritas en [40] sobre o uso dun limiar flexible en clasificación, existen razóns para pensar que un detector de obxectos tamén se debería beneficiar deste tipo de estratexias. Non obstante, o uso dun limiar flexible en problemas de detección supón unha serie de dificultades adicionais, que requiren dunha análise máis profunda fóra do alcance deste traballo.

## 5.6. Experimento 5: SimOTA

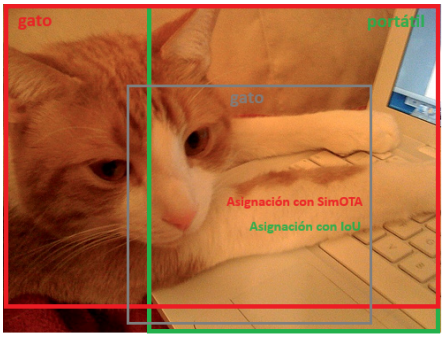
Finalmente, partindo do detector Unbiased Teacher v2 con módulos GDL, o último experimento levado a cabo consiste en modificar a estratexia de asignación de etiquetas tradicional pola estratexia SimOTA detallada na sección 3.5.1.

O primeiro que se observou é que o adestramento con SimOTA evoluciona

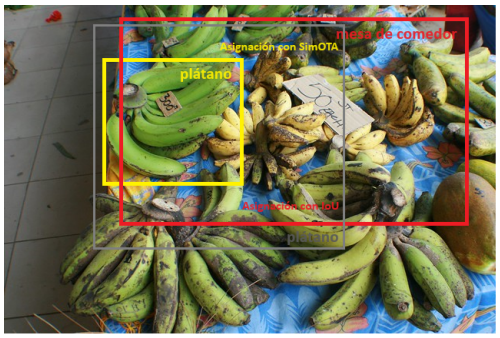
máis lentamente ca o adestramento orixinal. Isto ten sentido tendo en conta a adaptación a detectores de dúas etapas descrita no apartado 3.5.2, pois o número de detecciónss que pasan á R-CNN é inferior se engadimos SimOTA (as propostas non asignadas polo Algoritmo 2 non se empregan para adestrar). De feito, comprobouse que o número de detecciónss empregadas polo detector con SimOTA supoñen arredor de dous terzos das que se empregan co detector orixinal. Por este motivo, é razoable considerar a posibilidade de aumentar o número de iteracións da etapa de Burn-In. Así, tras realizar unhas pequenas probas, determinouse que o máis beneficioso para o adestramento é establecer **3.000 iteracións de Burn-In**.

Por outra parte, foi necesario determinar os valores dos hiperparámetros $r$ e $q$ do Algoritmo 2. Para isto, realizáronse unhas pequenas probas cun 20 % dos datos e decidiuse empregar $q = 20$ e $r = 10$.

Así, realizouse un experimento con 3.000 iteracións de Burn-In seguindo a implementación do Algoritmo 2 adaptada a un detector de dúas etapas, tal e como se explica no apartado 3.5.2. A Figura 5.7 mostra algunhas das detecciónss obtidas con esta implementación. As detecciónss mostradas correspóndense con propostas da RPN que se asocian a distintos *ground truth* en función de se se emprega SimOTA ou asignación de etiquetas tradicional. Estas detecciónss mostran claramente as vantaxes de utilizar SimOTA.




(a) A proposta predí "gato", pero ten máis IoU co *gt* "portátil". SimOTA asigna esta proposta ao *gt* "gato".

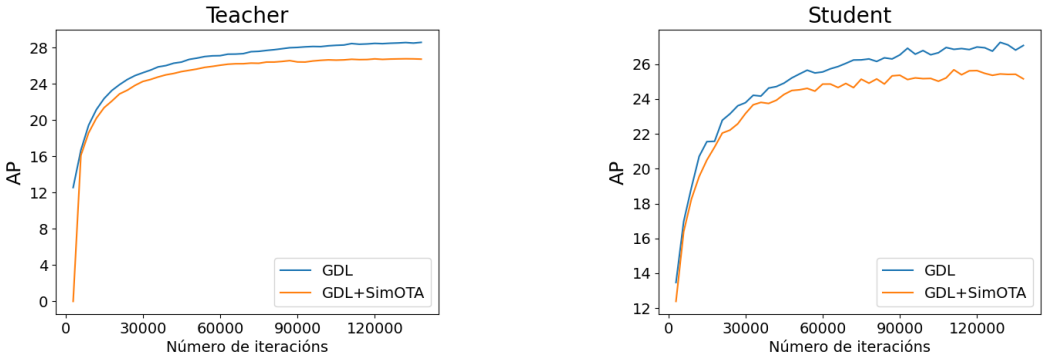



(b) A proposta predí "plátano", pero ten máis IoU co *gt* 'mesa". SimOTA asigna esta proposta ao *gt* "plátano".

Figura 5.7: Asignacións diferentes entre SimOTA e asignación baseada en IoU, onde se mostran as vantaxes de usar SimOTA. Os cadros de cores representan o *ground truth* e os cadros grises unha proposta da RPN coa súa predición. Para facilitar a visualización, só se mostra unha detección asignada de forma distinta polos dous métodos e os *ground truth* involucrados.

O resultado final obtido con este experimento móstrase no Cadro 5.2. Observamos que **a adaptación de SimOTA proposta non consegue mellorar o detector**. A Figura 5.8 mostra unha comparación da evolución do adestramento entre empregar SimOTA e non facelo. Así, observamos que, ao principio do adestramento, ambos detectores melloran a un ritmo semellante, aínda que a

versión sen SimOTA é lixeiramente peor. Non obstante, a medida que avanza o adestramento, as curvas de AP vanse separando e o rendemento do detector con SimOTA estáncase.




(a) Evolución do AP do *Teacher*

(b) Evolución do AP do *Student*

Figura 5.8: Comparación dos resultados da validación dos detectores Unbiased Teacher v2 + GDL con asignación de etiquetas con e sen SimOTA.

A continuación, listamos algúns problemas detectados nesta implementación:

- Como xa se comentou, observouse que a estratexia de asignación proposta descarta sobre un terzo das propostas da RPN que se usarían con asignación tradicional. Así, é posible que se descarten deteccióǹs que serían beneficiosas para o adestramento. Sería convinte reformular esta proposta para que se empreguen no adestramento unha maior cantidade de deteccióǹs.

- Analizáronse as diferenzas entre as asignacióǹs de ambos métodos e observouse que, a pesar de que SimOTA realiza asignacióǹs mellores nalgúns casos como os da Figura 5.7, tamén se dá o caso contrario. Estudando os casos nos que SimOTA realiza unha asignación peor, chegouse á conclusión de que a maioría das veces se debe a unha mala estimación de $k$, o valor que indica cantas propostas asignar a cada *ground truth*. Para axustar mellor este valor $k$ sería necesario modificar o hiperparámetro $q$. A duración dos adestramentos limitou moito probar adecuadamente diferentes hiperparámetros, polo que sería necesaria unha análise máis profunda de cómo inflúe o valor de $q$ na calidade das asignacióǹs.

A asignación de etiquetas implementada é unha adaptación de SimOTA a un detector de dúas etapas. Ata onde sabemos, non existen intentos na literatura de empregar unha estratexia baseada en OTA en detectores de dúas etapas. Así, a proposta de adaptación presentada realizouse sen contar con exemplos previos. Ademais, a duración do adestramento limitou moito o número de probas que se levaron a cabo. Por este motivo, aínda existe moita marxe de mellora para a estratexia proposta.

# Capítulo 6

# Discusión dos resultados

A partir dos resultados da experimentación, podemos extraer as seguintes conclusións:

1. O rendemento do detector Unbiased Teacher v2 presenta moitas diferenzas en función do tamaño dos obxectos: as deteccióńs de obxectos pequenos son significativamente peores que as de obxectos medianos e grandes.

2. O detector Unbiased Teacher v2 mellora o seu rendemento cando se desacoplan as tarefas da RPN e da R-CNN empregando *Gradient Decouple Layers* (GDL).

3. Os detector Unbiased Teacher v2 empeora claramente o seu rendemento cando se engade un *Prototypical Calibration Block* (PCB).

4. A adaptación proposta para detección de obxectos da estratexia de limiar flexible descrita en [40] para filtrar as pseudo-etiquetas do *Teacher* non logrou mellorar o rendemento do detector Unbiased Teacher v2. Non obstante, os resultados tampouco empeoraron significativamente.

5. As clases máis perxudicadas cando se emprega a estratexia de limiar flexible son aquelas que teñen un maior AP.

6. Empregar unha estratexia de limiar flexible nun problema de detección de obxectos presenta unha serie de dificultades adicionais comparado cun problema de clasificación: na sección 5.5 detállanse algúns dos problemas detectados.

7. As modificacións realizadas sobre a estratexia de limiar flexible orixinal para solucionar os problemas detectados non lograron mellorar o detector. Na sección 5.5 detállanse as modificacións probadas e analízanse posibles razóns da ausencia de melloras.

8. Empregar unha estratexia de asignación de etiquetas adaptando SimOTA [41] a un detector de dúas etapas non logrou mellorar o rendemento do detector Unbiased Teacher v2.

9. Detectáronse varios problemas na proposta de SimOTA presentada, que non se puideron abordar debido a limitacións de tempo e recursos.

En vista dos resultados obtidos, podemos afirmar que a **Hipótese 1** establecida na sección 1.2 queda **parcialmente confirmada**. As conclusións 2 e 3 mostran que algunhas técnicas propias de arquitecturas *few-shot* tamén son beneficiosas para contextos semi-supervisados, mentres que outras poden deteriorar o seu rendemento.

Con respecto á **Hipótese 2**, esta **non queda confirmada, pero tampouco desmentida**. As conclusións 4 e 7 mostran que non se logrou deseñar unha estratexia de limiar flexible que mellorara o detector semi-supervisado. Non obstante, a conclusión 6, unida ao éxito deste tipo de estratexias en problemas de clasificación, suxiren que é necesaria máis investigación nesta liña. Ademais, só se probou esta estratexia co detector Unbiased Teacher v2, o cal non significa que non poida mellorar o rendemento doutros detectores semi-supervisados.

Por último, a **Hipótese 3 tampouco queda confirmada nin desmentida**. Por un lado, a conclusión 8 mostra que non se logrou mellorar o detector empregando unha estratexia de asignación de etiquetas baseada en OTA [3]. Non obstante, a conclusión 9, unida ao feito de que, ata onde sabemos, este traballo é o primeiro intento de adaptar este tipo de estratexia a un detector de dúas etapas, suxiren que é necesaria máis investigación ao respecto.

# Capítulo 7

# Conclusións e posibles ampliacións

Ao longo deste traballo estudouse a **aprendizaxe semi-supervisada para detección de obxectos**, unha rama de gran importancia debido ás dificultades e custos que supón a anotación manual de imaxes. Concretamente, este traballo centrouse na arquitectura de tipo *Teacher-Student*.

En primeiro lugar, analizouse en profundidade o detector de obxectos **Unbiased Teacher v2**, co obxectivo de entender o seu funcionamento para despois introducir melloras no seu rendemento.

Tras esta análise, propúxose a idea de introducir neste detector semi-supervisado melloras propias de arquitecturas ***few-shot***, dadas as similitudes entre ambos contextos de aprendizaxe. Así, comprobouse que un detector semi-supervisado benefíciase do desacoplamento entre a RPN e a cabeceira R-CNN proporcionado polos módulos **GDL**, pero que o desacoplamento entre a clasificación e a regresión da cabeceira aportado polo **PCB** resulta perxudicial.

A continuación, decidiuse modificar a arquitectura para que, en lugar de empregar un limiar fixo para filtrar as pseudo-etiquetas, se empregara un **limiar flexible** baseado no estado da aprendizaxe de cada clase. Esta modificación revelou que o uso de estratexias de limiar flexible no campo da detección de obxectos supón unha serie de dificultades adicionais que non aparecen en problemas de clasificación. Deste modo, a adaptación proposta non conseguiu mellorar o detectector e ningunha das modificacións suxeridas para solucionar os problemas observados logrou mellorar os resultados.

Finalmente, tratouse de modificar a **asignación de etiquetas** realizada por Unbiased Teacher v2 por unha estratexia baseada en OTA [3]. Concretamente, adaptouse o algoritmo **SimOTA** proposto en [41] a un detector de dúas etapas. Esta adaptación non conseguiu mellorar o detector, pero observáronse certos problemas que non puideron ser abordados por falta de tempo e recursos.

Neste traballo, tratouse de adaptar estratexias beneficiosas noutros contextos a un detector semi-supervisado de dúas etapas cunha arquitectura *Teacher-*

*Student*. Os resultados obtidos, en xeral, non lograron mellorar o rendemento do detector, pero tampouco o empeoraron de maneira moi significativa. Ademais, detectáronse unha serie de problemas nas adaptacións propostas, que requerirían dunha análise máis profunda fóra do alcance deste traballo. Todo isto, sumado ás limitacións na experimentación que supuxo o tempo de adestramento do detector, suxire que é necesaria máis investigación sobre a incorporación de estratexias de limiar flexible e asignación de etiquetas baseada en OTA para o detector Unbiased Teacher v2.

Tendo isto en conta, dous posibles traballos futuros que axudarían a determinar se as estratexias estudadas poden beneficiar a detectores semi-supervisados como Unbiased Teacher v2 son:

- Realizar unha análise detallada da relevancia no adestramento de cada un dos problemas detectados no Experimento 5.5. A partir disto, propoñer novas adaptacións á detección de obxectos da estratexia de limiar flexible descrita en [40] para clasificación.

- Realizar probas exhaustivas para determinar os hiperparámetros máis adecuados para que a adaptación de SimOTA proposta realice o menor número posible de asignacións erróneas. Ademais, explorar estratexias nas que o número de detección descartadas no adestramento sexa inferior.

# Apéndice A

# Manual técnico

Neste manual técnico preséntase o código fonte empregado, e inclúese unha descrición do contido dos diferentes arquivos presentados. Deste modo, espérase que resulte doado realizar modificacións sobre distintas partes do código.

A totalidade do código e outra información de interese pode atoparse no seguinte repositorio público de GitHub:

`https://github.com/cristinalopezamado/TFG.git`.

A estrutura en carpetas do repositorio, esquematizada na Figura A.1, é a seguinte:

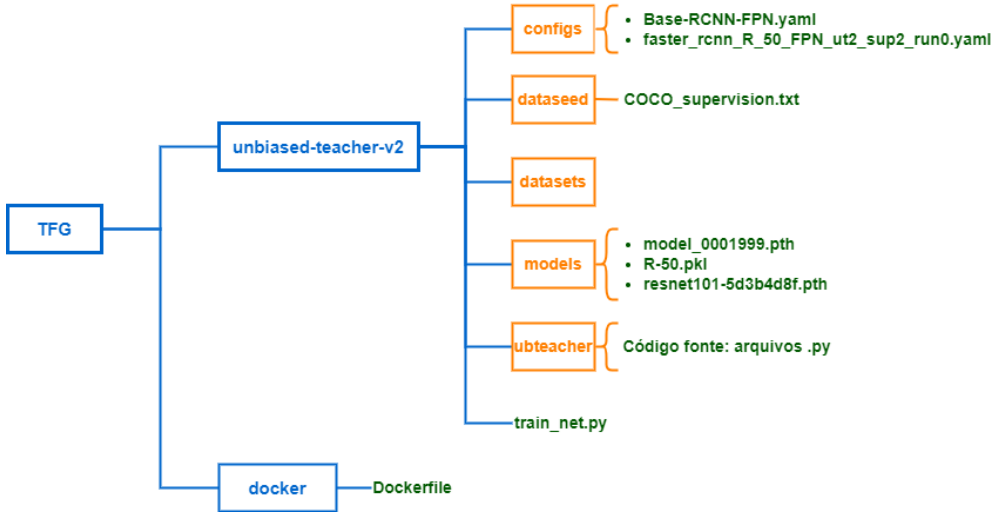



Figura A.1: Organización en carpetas do repositorio deste traballo.

- `docker/`: conten o *Dockerfile* que instala o software necesario para executar os experimentos.
- `unbiased-teacher-v2/`: contén os arquivos necesarios para executar o modelo:

- `configs/`: contén os arquivos de configuración, que permiten configurar os distintos hiperparámetros do modelo.
- `dataseed/`: contén o arquivo `COCO_supervision.txt`, no que se especifican os ids das imaxes que estarán etiquetadas en función da porcentaxe de supervisión elexida. Como isto debe ser aleatorio, proporciónanse 10 "sementes" para cada porcentaxe de supervisión.
- `datasets/`: no repositorio non contén nada, pero este directorio debe conter o conxunto de datos e as súas anotacións. No Anexo B explícase como obter o conxunto de datos e como debe ser a estrutura desta carpeta.
- `models/`: debe conter, polo menos, os modelos preadestrados que se indican na Figura A.1. No repositorio inclúese un enlace para descargar estes modelos, pois son moi pesados para incluilos no repositorio.
- `ubteacher/`: contén todo o código fonte organizado en carpetas.
- `train_net.py`: arquivo que contén o programa principal (*main*) do código fonte.

A continuación, detallamos o contido de `ubteacher/`, que contén todo o código fonte. O esquema de directorios é o seguinte:

```
ubteacher/
├── checkpoint/
├── data/
├── engine/
├── evaluation/
├── layers/
├── modeling/
│   ├── archs/
│   ├── backbone/
│   ├── fcos/
│   ├── meta_arch/
│   ├── proposal_generator/
│   └── roi_heads/
├── solver/
├── utils/
└── configs.py
```

- `checkpoint/`: contén o código asociado á creación de *checkpoints*, é dicir, encárgase de gardar os pesos dos modelos cada certo número de iteracións. Tamén xestiona o caso no que o adestramento parta dun modelo previamente adestrado. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.

- `data/`: contén todo o código relacionado coa carga do conxunto de datos, a súa distribución en mini-lotes e a aplicacións de transformacións fortes e débiles. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.
- `engine/`: contén a clase `UBRCNNTeacherTrainer`, que xestiona o adestramento da arquitectura *Teacher-Student*. Así, chama aos adestramentos do *Teacher* e do *Student*, crea as pseudo-etiquetas empregando as confianzas, chama á actualización do *Teacher* mediante EMA, etc. Esta clase modificouse principalmente para engadir a estratexia de limiar flexible.
- `evaluation/`: contén o código asociado á avaliación no dataset de COCO. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.
- `layers/`: contén a implementación de distintas funcións de perda e outras operacións necesarias. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.
- `modeling/`: contén a implementación das distintas capas da rede neuronal. Esta formada polas seguintes carpetas:
  - `archs/`: contén a implementación de ResNet [7]. Esta carpeta foi engadida para implementar o PCB, pois o seu clasificador emprega ResNet como arquitectura.
  - `backbone/`: contén a implementación de FPN [12], pois esta é a arquitectura que usa o extractor de características (*backbone*) do detector. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.
  - `fcos/`: carpeta incluida no detector orixinal, pero que non empregamos no noso detector. Implementa unha versión de Unbiased Teacher v2 para detectores libres de rexións.
  - `meta_arch/`: contén a clase `TwoStagePseudoLabGeneralizedRCNN`, que procesa un mini-lote de adestramento ou inferencia do modelo: chama ao preprocesado das imaxes, ao extractor de características, á RPN e á cabeceira e obtén a perda total ou o resultado da inferencia. Esta clase foi modificada para engadir o procesamento das características a través dos módulos GDL. Para isto, tamén foi necesario engadir un novo arquivo (`gdl.py`), coa implementación de GDL extraída de [2]. Ademais disto, nesta carpeta tamén se xestiona o duplicado do modelo en *Teacher* e *Student*.
  - `proposal_generator/`: contén a implementación da RPN. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.

  - `roi_heads/`: implementa as funcións da cabeceira R-CNN. Esta carpeta foi modificada principalmente para implementar a asignación de etiquetas empregando SimOTA. Ademais, tamén inclúe a modificación na que a R-CNN ignora certas rexións, realizada sobre a estratexia de limiar flexible.

  Ademais destas carpetas, por unha parte, `modeling/` contén código relacionado coa función de regresión, que non se modificou. Por outra, incluiuse o arquivo `calibration_layer.py`, que contén a implementación do PCB.

- `solver/`: contén código asociado ao axuste da taxa de aprendizaxe. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.

- `utils/`: contén funcións de utilidade para a manipulación das caixas. Esta carpeta non se modificou, polo que todo o contido foi implementado no detector orixinal.

- `configs.py`: arquivo que establece os hiperparámetros da configuración. Modificouse para engadir os hiperparámetros necesarios para as propostas implementadas.

# Apéndice B

# Manual de usuario

Neste manual incluimos a información necesaria para que calquera usuario poida replicar de maneira sinxela os experimentos realizados neste traballo. En primeiro lugar, explicaremos como instalar o *software* necesario para realizar as probas. A continuación, explicaremos como obter os datasets necesarios. Finalmente, detallaremos como realizar as execucións e como configurar os diferentes hiperparámetros que permiten replicar os distintos experimentos.

## B.1. Instalación do *software*

Para poder executar os experimentos descritos neste traballo son necesarios os seguintes prerrequisitos:

- Linux con **Python** $\geq$ 3,7 [55].
- **PyTorch** $\geq$ 1,10 [57] e **torchvision** compatible coa versión de PyTorch.
- **CUDA** 11.1.1 [61].
- **Detectron2** $\geq$ 0,6 [60].
- Bibliotecas de Python adicionais: Matplotlib, OpenCV, Scikit-learn e OmegaConf.
- Bibliotecas libgl1 e libglib2.0-0.

Para facilitar a instalación dos prerrequisitos e evitar problemas de dependencias, proporciónase o seguinte *Dockerfile*, incluido no repositorio de GitHub:

```
1 #Imaxe base de CUDA 11.1 con Ubuntu 20.04
2 FROM nvidia/cuda:11.1.1-cudnn8-devel-ubuntu20.04
3 ENV DEBIAN_FRONTEND=noninteractive
4
```

```
#Ferramentas basicas e dependencias
RUN apt-get update && \
    apt-get install -y \
    wget \
    git \
    build-essential \
    libssl-dev \
    libffi-dev \
    python3.8 \
    python3.8-dev \
    python3-pip \
    python3-setuptools \
    python3-wheel \
    libgl1 \
    libglib2.0-0
#pip para Python 3.8
RUN wget https://bootstrap.pypa.io/get-pip.py && \
    python3.8 get-pip.py && \
    rm get-pip.py

#torch e torchvision
RUN pip install torch==2.3.1 torchvision==0.18.1 -f
   https://download.pytorch.org/whl/cu111/ \
torch_stable.html


#Paquetes adicionais necesarios para Detectron2 e
   omegaconf
RUN pip install cython pyyaml==5.4.1 matplotlib
   opencv-python omegaconf scikit-learn

#Instalamos librerias adicionais necesarias para
   algunhas operacions de Detectron2
RUN apt-get install -y libopencv-dev

#Instalamos Detectron2
RUN python3.8 -m pip install
   'git+https://github.com/facebookresearch/' \
'detectron2.git'

#Clonamos o repositorio TFG dende GitHub
RUN git clone
   https://github.com/cristinalopezamado/TFG.git /TFG
```

```
42
43 #PYTHONPATH ten a ubicacion do codigo
44 ENV PYTHONPATH
      /detectron2:/TFG/unbiased-teacher-v2:$PYTHONPATH
45 CMD ["python3"]
```

Así, creando unha imaxe de Docker a partir deste *Dockerfile*, instálanse todos os prerrequisitos e clónase o repositorio de GitHub co código. Para crear unha imaxe de Docker, simplemente hai que situarse no directorio do *Dockerfile* e executar:

```
docker build -t nome_imaxe
```

Finalmente, créase un contedor executando:

```
docker run --shm-size=32g -it --name nome_contedor -e
   DETECTRON2_DATASETS=/datasets --gpus all -v ruta/a/
   conxunto_datos:/datasets nome_imaxe /bin/bash
```

A continuación, detallamos as opcións deste comando:

- `--shm-size`: establece o tamaño en gigas da memoria compartida que pode usar o contedor.
- `-it`: o contedor execútase en modo interactivo, é dicir, despois de iniciar o contedor ábrese unha sesión interactiva de *bash*.
- `--name`: permite establecer o nome do contedor.
- `-e`: permite establecer variables de contorna. Neste caso, defínese a variable de contorna `DETECTRON2_DATASETS` coa ruta ao conxunto de datos (dentro do contedor). Daremos máis detalles de como preparar o conxunto de datos na seguinte sección.
- `--gpus all`: indica que se lle van a pasar ao contedor todos os dispositivos GPU dispoñibles. Se se quere especificar que GPUs pode usar o contedor, o comando sería `--gpus '"device=X,Y,Z"'`, onde `X`, `Y`, `Z` son os ids das GPUs que se empregarán.
- `-v ruta/a/conxunto_datos:/datasets`: monta un volume dentro do contedor. Neste caso, estase a montar o directorio que contén os datos dentro do directorio `/datasets` do contedor. Isto non é obrigatorio, pois poderían copiarse os datos directamente no contedor. Non obstante, é recomendable se xa se teñen gardados os datos na máquina para aforrar espazo.
- `nome_imaxe`: nome da imaxe que se usará para crear o contedor. Neste caso, a imaxe creada anteriormente a partir do *Dockerfile*.

Unha vez creado o contedor, podemos acceder a el co comando:

```
docker attach nome_contedor
```

## B.2. Conxunto de datos

A continuación, especificamos como obter o conxunto de datos de COCO e como colocalo na estrutura de carpetas para poder executar o código correctamente.

O conxunto de datos utilizado é o conxunto de adestramento e validación de COCO 2017. Para descargar as imaxes, é necesario executar:

```
wget http://images.cocodataset.org/zips/train2017.zip
wget http://images.cocodataset.org/zips/val2017.zip
unzip train2017.zip
unzip val2017.zip
```

Para obter as anotacións, executamos:

```
wget http://images.cocodataset.org/annotations/
   annotations_trainval2017.zip
unzip annotations_trainval2017.zip
```

Unha vez descargadas, as imaxes e as anotacións deben colocarse na árbore de directorios do seguinte xeito:

```
unbiased_teacher_v2/
└── datasets/
    └── coco/
        ├── train2017/
        ├── val2017/
        └── annotations/
            ├── instances_train2017.json
            └── instances_val2017.json
```

Unha alternativa a isto, tal e como se viu na sección anterior, é definir a variable de contorna `DETECTRON2_DATASETS` coa ruta ao conxunto de datos, que debe seguir a estrutura da carpeta `datasets/` anterior.

## B.3. Execución de adestramentos

A continuación, detallamos como executar os adestramentos que permiten replicar os experimentos presentadas neste traballo.

En primeiro lugar, explicaremos como empregar os arquivos de configuración `Base-RCNN-FPN.yaml` e `faster_rcnn_R_50_FPN_ut2_sup2_run0.yaml` dispoñibles na carpeta `configs/Faster-RCNN/`. Estes arquivos conteñen numerosos hiperparámetros que se poden configurar para adestrar o modelo. Explicaremos soamente os máis relevantes, facendo especial énfasis naqueles que permiten configurar os experimentos realizados neste traballo. Para realizar outro tipo de modificacións na configuración, recomendamos ler a documentación de Unbiased Teacher v2 e Detectron2.

Por un lado, `Base-RCNN-FPN.yaml` contén a configuración relativa ao detector Faster R-CNN empregado:

```
MODEL:
  META_ARCHITECTURE: "GeneralizedRCNN"
  BACKBONE:
    NAME: "build_resnet_fpn_backbone"
  RESNETS:
    OUT_FEATURES: ["res2", "res3", "res4", "res5"]
  FPN:
    IN_FEATURES: ["res2", "res3", "res4", "res5"]
  ANCHOR_GENERATOR:
    SIZES: [[32], [64], [128], [256], [512]]
    ASPECT_RATIOS: [[0.5, 1.0, 2.0]]
  RPN:
    IN_FEATURES: ["p2", "p3", "p4", "p5", "p6"]
    PRE_NMS_TOPK_TRAIN: 2000
    PRE_NMS_TOPK_TEST: 1000
    POST_NMS_TOPK_TRAIN: 1000
    POST_NMS_TOPK_TEST: 1000
    ENABLE_DECOUPLE: True
    BACKWARD_SCALE: 0.0
  ROI_HEADS:
    NAME: "StandardROIHeads"
    IN_FEATURES: ["p2", "p3", "p4", "p5"]
    ENABLE_DECOUPLE: True
    BACKWARD_SCALE: 1.0
  ROI_BOX_HEAD:
    NAME: "FastRCNNConvFCHead"
    NUM_FC: 2
    POOLER_RESOLUTION: 7
  ROI_MASK_HEAD:
    NAME: "MaskRCNNConvUpsampleHead"
    NUM_CONV: 4
    POOLER_RESOLUTION: 14
```

```
33
34    SIMOTA_ASSIGNMENT: False
35    SIMOTA_COEF_REG: 1.5
36    SIMOTA_RADIUS: 10
37    SIMOTA_Q_DYNAMICK: 20
38  DATASETS:
39    TRAIN: ("coco_2017_train",)
40    TEST: ("coco_2017_val",)
41  SOLVER:
42    IMS_PER_BATCH: 16
43    BASE_LR: 0.02
44    STEPS: (60000, 80000)
45    MAX_ITER: 90000
46    CHECKPOINT_PERIOD: 2000
47  INPUT:
48    MIN_SIZE_TRAIN: (640, 672, 704, 736, 768, 800)
49  VERSION: 2
50  TEST:
51    EVAL_PERIOD: 3000
52    PCB_ENABLE: False
53    PCB_MODELPATH: "models/resnet101-5d3b4d8f.pth"
54    PCB_MODELTYPE: "resnet"
55    PCB_ALPHA: 0.50
56    PCB_UPPER: 1.0
57    PCB_LOWER: 0.05
58
59  OUTPUT_DIR: "/ruta/output/"
```

Por outro lado, `faster_rcnn_R_50_FPN_ut2_sup2_run0.yaml` contén a configuración relativa ao adestramento semi-supervisado:

```
1  _BASE_: "../Base-RCNN-FPN.yaml"
2  MODEL:
3    META_ARCHITECTURE: "TwoStagePseudoLabGeneralizedRCNN"
4    WEIGHTS: "models/R-50.pkl"
5    MASK_ON: False
6    RESNETS:
7      DEPTH: 50
8    PROPOSAL_GENERATOR:
9      NAME: "PseudoLabRPN"
10   RPN:
11     POSITIVE_FRACTION: 0.25
```

```
    LOSS: "CrossEntropy"
  ROI_HEADS:
    NAME: "StandardROIHeadsPseudoLab"
    LOSS: "FocalLoss_BoundaryVar"
  ROI_BOX_HEAD:
    BBOX_REG_LOSS_TYPE: "nlloss"
    CLS_AGNOSTIC_BBOX_REG: True
SOLVER:
  LR_SCHEDULER_NAME: "WarmupMultiStepLR"
  STEPS: (180000,)
  MAX_ITER: 180000
  IMG_PER_BATCH_LABEL: 32
  IMG_PER_BATCH_UNLABEL: 32
  BASE_LR: 0.01
DATALOADER:
  SUP_PERCENT: 2.0
  RANDOM_DATA_SEED: 0
  #Para facer probas mais rapido, reducimos o numero
     de datos para entrenar
  REDUCE_UNLABEL_DATA: False
  #Porcentaxe de datos utilizados
  UNLABEL_PERCENT_USE: 20.0
DATASETS:
  CROSS_DATASET: False
  TRAIN: ("coco_2017_train",)
  TEST: ("coco_2017_val",)
SEMISUPNET:
  Trainer: "ubteacher_rcnn"
  BBOX_THRESHOLD: 0.7
  FLEXIBLE_THRESHOLD: False
  FLEXMATCH_NMS: False
  IOU_FLEXMATCH_NMS: 0.5
  MIN_BBOX_THRESHOLD: 0.0
  MAX_BBOX_THRESHOLD: 1.0
  RANDOM_MAX_THRESHOLD: False
  FLEX_WARMUP: False
  IGNORE_REGIONS: False
  THRESHOLD_IGNORE_REGIONS: 0.7
  NUM_ITERS_BEFORE_FLEX: 0
  TEACHER_UPDATE_ITER: 1
  BURN_UP_STEP: 2000
  EMA_KEEP_RATE: 0.9996
  UNSUP_LOSS_WEIGHT: 3.0
```

```
  UNSUP_REG_LOSS_WEIGHT: 1.0
TEST:
  EVAL_PERIOD: 3000
  EVALUATOR: "COCOeval"
  VAL_LOSS: False
INPUT: # scale jittering (follow soft teacher)
  MIN_SIZE_TRAIN: (400, 1200)
  MIN_SIZE_TRAIN_SAMPLING: "range"
```

A continuación, comentamos os hiperparámetros máis relevantes:

- `MODEL.RPN.ENABLE_DECOUPLE` e `MODEL.ROI_HEADS.ENABLE_DECOUPLE` indican se está activado (`True`) ou non (`False`) o GDL da RPN e da R-CNN respectivamente. Os hiperparámetros `MODEL.RPN.BACKWARD_SCALE` e `MODEL.ROI_HEADS.BACKWARD_SCALE` correspóndense cos valores $\lambda_{rpn}$ e $\lambda_{rcnn}$ da Ecuación (3.3.1).

- `MODEL.SIMOTA_ASSIGNMENT` indica se está activado (`True`) ou non (`False`) a asignación de etiquetas empregando SimOTA. `MODEL.SIMOTA_COEF_REG` correspóndese co parámetro $\gamma$ da Ecuación (3.25), `MODEL.SIMOTA_RADIUS` é o radio máximo permitido para unha asignación *ground truth*-proposta e `MODEL.SIMOTA_Q_DYNAMICK` é o parámetro $q$ descrito en 3.5.1 para a estimación dinámica de $k$.

- `DATASETS.TRAIN` e `DATASETS.VAL` especifican os conxuntos de datos de adestramento e validación respectivamente.

- `SOLVER_BASE_LR` establece a taxa de aprendizaxe (*learning rate*).

- `SOLVER.CHECKPOINT_PERIOD` establece cada cantas iteracións se garda unha copia do modelo adestrado ata o momento (*checkpoint*).

- `TEST.EVAL_PERIOD` establece cada cantas iteracións se fai unha validación.

- `TEST.PCB_ENABLE` indica se se emprega (`True`) ou non (`False`) o módulo PCB. `TEST.PCB_ALPHA` correspóndese co parámetro $\alpha$ da Ecuación (3.20). `TEST.PCB_MODELPATH` correspóndese cos pesos do modelo preadestrado que se empregan para a PCB. Neste traballo, empregouse unha resnet 101 adestrada con ImageNet, que se pode descargar no enlace incluido na carpeta `models/`.

- `OUTPUT_DIR` indica o directorio onde se gardan os resultados do adestramento.

- `MODEL.WEIGHTS` indica o modelo preadestrado do que se parte. No repositorio de GitHub, inclúese unha carpeta `models/`, que se pode usar para

gardar estes modelos. O repositorio proporciona enlaces para descargar estes modelos preadestrados. O modelo `R-50.pkl` correspóndese co modelo preadestrado con ImageNet, que é o modelo de partida empregado nestas probas. Tamén se inclúe un modelo `model_0001999`, que se corresponde co modelo despois de 2000 iteracións de adestramento. Resulta útil empregalo se se queren facer diversas probas cambiando só a parte semi-supervisada do adestramento.

- `SOLVER.MAX_ITER` indica o número máximo de iteracións que durará o adestramento.
- `SOLVER.IMG_PER_BATCH_LABEL` e `SOLVER.IMG_PER_BATCH_UNLABEL` son o tamaño de lote (*batch*) para os datos etiquetados e sen etiquetar respectivamente.
- `DATALOADER.SUP_PERCENT` indica a porcentaxe de imaxes etiquetadas que se empregará sobre o total de imaxes. Neste traballo, todos os experimentos foron realizados empregando un 2 % dos datos.
- `DATALOADER.REDUCE_UNLABEL_DATA` empregouse nalgunhas probas para facer adestramentos máis curtos empregando só unha porcentaxe dos datos dispoñibles. Así, se `DATALOADER.REDUCE_UNLABEL_DATA` é `True`, empregarase o `DATALOADER.UNLABEL_PERCENT_USE` dos datos sen etiquetar dispoñibles.
- Se `SEMISUPNET.FLEXIBLE_THRESHOLD` é `False`, `SEMISUPNET.BBOX_THRESHOLD` establece o limiar fixo para filtrar as pseudo-etiquetas do *Teacher*.
- Se `SEMISUPNET.FLEXIBLE_THRESHOLD` é `True` emprégase a estratexia de limiar flexible en lugar de limiar fixo. Neste caso:
  - `SEMISUPNET.BBOX_THRESHOLD` establece o limiar fixo a partir do cal se obteñen os limiares dinámicos.
  - `SEMISUPNET.FLEXMATCH_NMS_THRESHOLD` indica se se aplica NMS despois de obter as pseudo-etiquetas. No caso de que este parámetro sexa `True`, `SEMISUPNET.IOU_FLEXMATCH_NMS` indica o valor de IoU empregado para aplicar NMS.
  - `SEMISUPNET.MIN_BBOX_THRESHOLD` establece un limiar mínimo para empregar unha pseudo-etiqueta.
  - `SEMISUPNET.MAX_BBOX_THRESHOLD` establece un limiar máximo a partir do cal sempre se emprega unha pseudo-etiqueta.
  - Se `SEMISUPNET.RANDOM_MAX_THRESHOLD` é `True`, emprégase a estratexia de limiar máximo con selección aleatoria descrita en 5.5 empregando `SEMISUPNET.MAX_BBOX_THRESHOLD` como limiar máximo.

  - Se `SEMISUPNET.FLEX_WARMUP` é `True`, engádese a etapa inicial de quentamento descrita en 5.5.
  - Se `SEMISUPNET.IGNORE_REGIONS` é `True`, emprégase a estratexia de limiar flexible na que se ignoran as rexións entre certo limiar e o limiar dinámico descrita en 5.5. `SEMISUPNET.THRESHOLD_IGNORE_REGIONS` indica o limiar empregado para ignorar estas rexións.
  - `SEMISUPNET.NUM_ITERS_BEFORE_FLEX` indica o número de iteracións de aprendizaxe mutuo con limiar fixo que se realizan antes de aplicar a estratexia de limiar flexible.

- `SEMISUPNET.BURN_UP_STEP` indica o número de iteracións que dura a etapa de Burn-In. Se se parte dun modelo preadestrado co Burn-In, esta debe valer 0.
- `SEMISUPNET.EMA_KEEP_RATE` correspóndese co parámetro $\alpha$ da Ecuación (3.16) empregado para calcular a EMA.

Deste modo, para replicar os experimentos realizados ou executar variacións destes, só é necesario modificar os arquivos de configuración. Unha vez establecidos os arquivos de configuración, para lanzar un adestramento, execútase o seguinte comando dende a carpeta `unbiased-teacher-v2/`:

```
python3 train_net.py \
      --num-gpus 4 \
      --config configs/Faster-RCNN/coco-standard/
         faster_rcnn_R_50_FPN_ut2_sup2_run0.yaml \
       SOLVER.IMG_PER_BATCH_LABEL 32 \
       SOLVER.IMG_PER_BATCH_UNLABEL 32
```

onde `--num-gpus` indica o número de GPUs que se empregarán, `--config` indica o arquivo de configuración empregado e `SOLVER.IMG_PER_BATCH_LABEL` e `SOLVER.IMG_PER_BATCH_UNLABEL` indica os tamaños de lote supervisado e non supervisado respectivamente.

Por outra parte, para realizar unha avaliación dun modelo, execútase:

```
python3 train_net.py \
    --eval-only \
    --num-gpus 4 \
    --config configs/Faster-RCNN/coco-standard/
       faster_rcnn_R_50_FPN_ut2_sup2_run0.yaml \
    SOLVER.IMG_PER_BATCH_LABEL 32 \
    SOLVER.IMG_PER_BATCH_UNLABEL 32 \
    MODEL.WEIGHTS <modelo_adestrado>.pth
```

onde `<modelo_adestrado>.pth` é o modelo que queremos avaliar.

# Bibliografía


[1] Y.-C. Liu, C.-Y. Ma e Z. Kira, "Unbiased teacher v2: Semi-supervised object detection for anchor-free and anchor-based detectors," en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

[2] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu e C. Zhang, "Defrcn: Decoupled faster r-cnn for few-shot object detection," en *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[3] Z. Ge, S. Liu, Z. Li, O. Yoshie e J. Sun, "Ota: Optimal transport assignment for object detection," en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

[4] S. Ren, K. He, R. Girshick e J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, vol. 28, 2015.

[5] S. Saha, *A Guide to Convolutional Neural Networks — the ELI5 way*, Consultado o 13 de xuño de 2024, 2018. URL: `https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/`.

[6] A. Krizhevsky, I. Sutskever e G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, pp. 84–90, 2017.

[7] K. He, X. Zhang, S. Ren e J. Sun, "Deep residual learning for image recognition," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[8] S. Xie, R. Girshick, P. Dollár, Z. Tu e K. He, "Aggregated residual transformations for deep neural networks," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[9] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[10] G. Huang, Z. Liu, L. Van Der Maaten e K. Q. Weinberger, "Densely connected convolutional networks," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[11] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell e S. Xie, "A convnet for the 2020s," en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[12] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan e S. Belongie, "Feature pyramid networks for object detection," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[13] R. Girshick, J. Donahue, T. Darrell e J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[14] R. Girshick, "Fast r-cnn," en *Proceedings of the IEEE international conference on computer vision*, 2015.

[15] J. Redmon, S. Divvala, R. Girshick e A. Farhadi, "You only look once: Unified, real-time object detection," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[16] J. Redmon e A. Farhadi, "YOLO9000: better, faster, stronger," en *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[17] J. Redmon e A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He e P. Dollár, "Focal loss for dense object detection," en *Proceedings of the IEEE international conference on computer vision*, 2017.

[19] W. Liu, D. Anguelov, D. Erhan et al., "Ssd: Single shot multibox detector," en *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, 2016.

[20] Z. Tian, C. Shen, H. Chen e T. He, "FCOS: A simple and strong anchor-free object detector," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, pp. 1922–1933, 2020.

[21] S. Zhang, C. Chi, Y. Yao, Z. Lei e S. Z. Li, "Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection," en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[22] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov e S. Zagoruyko, "End-to-end object detection with transformers," en *European conference on computer vision*, 2020.

[23] P. Gao, M. Zheng, X. Wang, J. Dai e H. Li, "Fast convergence of detr with spatially modulated co-attention," en *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[24] X. Zhu, W. Su, L. Lu, B. Li, X. Wang e J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[25] H. Zhang, F. Li, S. Liu et al., "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," *arXiv preprint arXiv:2203.03605*, 2022.

[26] Y. Wang, Z. Liu e S. Lian, "Semi-supervised Object Detection: A Survey on Recent Research and Progress," *arXiv preprint arXiv:2306.14106*, 2023.

[27] J. Jeong, S. Lee, J. Kim e N. Kwak, "Consistency-based semi-supervised learning for object detection," *Advances in neural information processing systems*, vol. 32, 2019.

[28] G. Li, X. Li, Y. Wang, Y. Wu, D. Liang e S. Zhang, "Pseco: Pseudo labeling and consistency training for semi-supervised object detection," en *European Conference on Computer Vision*, 2022.

[29] J. Hoffman, S. Guadarrama, E. S. Tzeng et al., "LSDA: Large scale detection through adaptation," *Advances in neural information processing systems*, vol. 27, 2014.

[30] X. Wang, T. E. Huang, T. Darrell, J. E. Gonzalez e F. Yu, "Frustratingly simple few-shot object detection," *arXiv preprint arXiv:2003.06957*, 2020.

[31] K. Sohn, Z. Zhang, C.-L. Li, H. Zhang, C.-Y. Lee e T. Pfister, "A simple semi-supervised learning framework for object detection," *arXiv preprint arXiv:2005.04757*, 2020.

[32] Q. Zhou, C. Yu, Z. Wang, Q. Qian e H. Li, "Instant-teaching: An end-to-end semi-supervised object detection framework," en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

[33] Y.-C. Liu, C.-Y. Ma, Z. He et al., "Unbiased Teacher for Semi-Supervised Object Detection," en *International Conference on Learning Representations*, 2021.

[34] M. Xu, Z. Zhang, H. Hu et al., "End-to-end semi-supervised object detection with soft teacher," en *Proceedings of the IEEE/CVF international conference on computer vision*, 2021.

[35] Z. Wang, Y.-L. Li, Y. Guo e S. Wang, "Combating noise: semi-supervised learning by region uncertainty quantification," *Advances in Neural Information Processing Systems*, vol. 34, pp. 9534–9545, 2021.

[36] Y. Tang, W. Chen, Y. Luo e Y. Zhang, "Humble teachers teach better students for semi-supervised object detection," en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[37] H. Zhou, Z. Ge, S. Liu et al., "Dense teacher: Dense pseudo-labels for semi-supervised object detection," en *European Conference on Computer Vision*, 2022.

[38] P. Wang, Z. Cai, H. Yang et al., "Omni-detr: Omni-supervised object detection with transformers," en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[39] Z. Chen, W. Zhang, X. Wang, K. Chen e Z. Wang, "Mixed pseudo labels for semi-supervised object detection," *arXiv preprint arXiv:2312.07006*, 2023.

[40] B. Zhang, Y. Wang, W. Hou et al., "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," *Advances in Neural Information Processing Systems*, vol. 34, pp. 18 408–18 419, 2021.

[41] Z. Ge, S. Liu, F. Wang, Z. Li e J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[42] *COCO - Common Objects in Context*, Consultado o 13 de xuño de 2024. URL: `https://cocodataset.org/#home`.

[43] Y. Lee, J.-w. Hwang, H.-I. Kim et al., "Localization uncertainty estimation for anchor-free object detection," en *European Conference on Computer Vision*, 2022.

[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li e L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," en *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[45] Sistemas Informáticos Europeos (SIE), *Plataformas SIE Ladon de 8 GPUs*, Consultado o 4 de xuño do 2024. URL: `https://www.sie.es/hpc/plataformas-sie-ladon-de-8-gpus/`.

[46] Intel Corporation, *Procesador Intel Xeon Silver 4214*, Consultado o 4 de xuño do 2024. URL: `https://www.intel.la/content/www/xl/es/products/sku/193385/intel-xeon-silver-4214-processor-16-5m-cache-2-20-ghz/specifications.html`.

[47] Nvidia Corporation, *Nvidia Quadro P6000*, Consultado o 4 de xuño do 2024. URL: `https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/productspage/quadro/quadro-desktop/quadro-pascal-p6000-data-sheet-a4-nv-704590-r1.pdf`.

[48] Nvidia Corporation, *Nvidia Quadro RTX 8000*, Consultado o 4 de xuño do 2024. URL: `https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/quadro-rtx-8000-us-nvidia-946977-r1-web.pdf`.

[49] Nvidia Corporation, *Nvidia A100 Tensor Core GPU*, Consultado o 4 de xuño do 2024. URL: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf.

[50] CentOS Development Team, *The CentOS Project*, Consultado o 4 de xuño do 2024. URL: https://www.centos.org/.

[51] Gigabyte, *GIGABYTE G482-Z54*, Consultado o 4 de xuño do 2024. URL: https://www.gigabyte.com/es/Enterprise/GPU-Server/G482-Z54-rev-100.

[52] AMD, *AMD EPYC 7413*, Consultado o 4 de xuño do 2024. URL: https://www.amd.com/es/products/cpu/amd-epyc-7413.

[53] *AlmaLinux*, Consultado o 4 de xuño do 2024. URL: https://almalinux.org/.

[54] Intel Corporation, *Procesador Intel Xeon Platinum 8352Y*, Consultado o 4 de xuño do 2024. URL: https://www.intel.la/content/www/xl/es/products/sku/212284/intel-xeon-platinum-8352y-processor-48m-cache-2-20-ghz/specifications.html.

[55] Python Software Foundation, *Python*, Consultado o 4 de xuño do 2024. URL: https://www.python.org/.

[56] NumPy, *NumPy documentation*, Consultado o 4 de xuño do 2024. URL: https://numpy.org/doc/stable/index.html.

[57] PyTorch, *PyTorch documentation*, Consultado o 4 de xuño do 2024. URL: https://pytorch.org/docs/stable/index.html.

[58] Matplotlib Development Team, *Matplotlib 3.9.0 documentation*, Consultado o 4 de xuño do 2024. URL: https://matplotlib.org/stable/index.html.

[59] OpenCV, *OpenCV-Python Tutorials*, Consultado o 4 de xuño do 2024. URL: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html.

[60] W. Yuxin, A. Kirillov, F. Massa, W. Lo e R. Girshick, *Detectron2*, 2023. URL: https://github.com/facebookresearch/detectron2.

[61] Nvidia, *CUDA Toolkit Documentation 12.5*, Consultado o 4 de xuño do 2024. URL: https://docs.nvidia.com/cuda/.

[62] *Docker Docs*, Consultado o 4 de xuño do 2024. URL: https://docs.docker.com/.

[63] *Documentation for Visual Studio Code*, Consultado o 4 de xuño do 2024. URL: https://code.visualstudio.com/docs.