UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

ESCOLA TÉCNICA SUPERIOR DE ENXEÑARÍA

Detección de obxectos e atributos cun conxunto de adestramento reducido

Autor/a:
Héctor Martínez Casares

Titores:
Manuel Mucientes Molina
Daniel Cores Costa

Grao en Enxeñaría Informática

Xullo 2023

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 *Detección de obxectos e atributos cun conxunto de adestramento reducido*, presentada por **D. Héctor Martínez Casares** 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 2023:

Titor/a,	Cotitor/a,	Alumno/a,
MUCIENTES MOLINA MANUEL - 44810714K Firmado digitalmente por MUCIENTES MOLINA MANUEL - 44810714K Fecha: 2023.07.03 16:50:45 +02'00'	Firmado por CORES COSTA DANIEL - ***9056** el día 03/07/2023 con un certificado emitido por AC FNMT Usuarios	34274850C HÉCTOR MARTÍNEZ Firmado digitalmente por 34274850C HÉCTOR MARTÍNEZ Fecha: 2023.07.03 16:25:10 +02'00'
Manuel Mucientes Molina	Daniel Cores Costa	Héctor Martínez Casares

Agradecementos

En primeiro lugar, quixera agraceder aos meus titores, Manuel e Dani, por guiarme e axudarme ao longo de todo o desenvolvemento deste traballo. Por suposto, grazas á miña familia, en especial aos meus pais e á miña parella, Sara, por estar sempre ao meu carón e apoiarme en todo momento. Finalmente, grazas aos meus amigos de toda a vida, os *Gor2*, e aos que fixen durante a carreira, Álvaro e Pablo. Todos vós fixestes máis fácil este camiño.

Resumo

A detección de obxectos en imaxes é un dos grandes problemas a abordar no campo da visión por computador. Os detectores clásicos requiren de enormes cantidades de imaxes etiquetadas para levar a cabo a aprendizaxe; isto supón un inconveniente importante, dado que, en moitas ocasións, a dispoñibilidade de datos é limitada. Para resolvelo, xurdiu a detección *few-shot*, que busca recoñecer obxectos de categorías nunca antes vistas empregando un conxunto de datos reducido. Este traballo está enfocado en estudar como afecta ao funcionamento deste tipo de detectores o feito de complementar a descrición dos obxectos etiquetados con información sobre os seus atributos (cor, patrón, material e reflectancia), ademais da categoría e a posición na imaxe. A anotación manual dunha lista de atributos implica un esforzo menor que o necesario para anotar obxectos novos, polo que unha mellora no rendemento suporía un avance interesante.

Co obxectivo de coñecer a influencia dos atributos, fíxose unha partición *few-shot* dun conxunto de datos de detección con atributos (PACO-LVIS [1]) e implementouse un detector fundamentado en DeFRCN [2] (un modelo *few-shot* baseado en redes neuronais convolucionais) con capacidade para a predición de atributos. A partir da experimentación levada a cabo, descubriuse que, nun escenario de información escasa, a utilización da información sobre os atributos supón unha melloría notable na tarefa de detección.

Índice xeral

Índice de figuras

Índice de táboas

Capítulo 1

Introdución

1.1. Descrición do problema

A **intelixencia artificial** (IA) é unha disciplina dentro das ciencias da computación cuxo propósito é a construción de sistemas informáticos capaces de realizar tarefas cun certo grao de intelixencia. En particular, a **aprendizaxe automática** é a rama da IA que busca dotar ás máquinas da capacidade de aprender, un rasgo esencial de intelixencia que lles permita mellorar o seu desempeño a través da experiencia.

Estreitamente relacionado coa IA, atópase a **visión por computador**, un campo científico centrado no procesamento e análise de imaxes co obxectivo de acadar unha comprensión profunda das mesmas. Un dos retos fundamentais no eido da visión por computador é a **detección de obxectos en imaxes**, un dominio clave nunha gran variedade de casos de uso nos que se busca extraer información dunha imaxe ou vídeo de forma automática, como poden ser a condución autónoma ou a robótica. A detección de obxectos lévase a cabo abordando dúas subtarefas: a localización dos obxectos presentes na escena e máis a clasificación dos mesmos; na figura 1.1 amósase un exemplo das detecciónss realizadas sobre unha imaxe. É precisamente debido á elevada complexidade do problema, que habitualmente se recorre a solucións baseadas na aprendizaxe automática.

Dentro da aprendizaxe automática, atópase a **aprendizaxe supervisada**, unha técnica que involucra o adestramento dun modelo a partir dun conxunto de datos acompañados da saída desexada. Os detectores de obxectos tradicionais requiren dun gran número de imaxes etiquetadas para realizar a optimización do modelo, superando usualmente varios miles de obxectos por cada categoría de interese. Isto limita enormemente as potenciais aplicacións destes sistemas, xa que o proceso de anotación é manual, co esforzo que isto implica. Por outra banda, existen contextos nos que a escaseza de información dificulta a obtención de exemplos de adestramento, co que resulta verdadeiramente complicado recoñecer obxectos de clases pouco comúns empregando os detectores clásicos.

Figura 1.1: Deteccións realizadas por un modelo Faster R-CNN [3] (R101-DC5) do Model Zoo de Detectron2 [4] sobre unha imaxe do conxunto de datos COCO [5].

Recentemente, xurdiron novas técnicas de aprendizaxe para detectores que empregan un conxunto moi reducido de exemplos de adestramento, da orde de decenas de obxectos etiquetados por categoría. Este TFG está centrado neste novo campo de investigación, coñecido coma **detección de obxectos *few-shot***. En concreto, estúdase como afecta ao funcionamento deste tipo detectores o feito de contar cunha descrición máis completa para cada un dos obxectos etiquetados. Especificamente, utilízanse descricións que determinan, ademais das características tradicionais (categoría e posición na imaxe), outros datos adicionais coñecidos como **atributos**, que caracterizan aos obxectos en función da súa cor, patrón, material e reflectancia.

En escenarios con poucos datos dispoñibles, o enriquecemento das anotacións a través da incorporación manual de información sobre atributos é unha opción perfectamente viable. Da mesma maneira, naqueles contextos nos que se conta con abundantes datos, a ampliación das descricións dos obxectos segue sendo unha alternativa menos custosa que a anotación de novos obxectos, xa que, en lugar de definir caixas adicionais que delimitan as posicións de cada un deles, soamente é necesario engadir unha etiqueta correspondente ao atributo nas anotacións dispoñibles.

1.2. Hipótese e obxectivos

A hipótese a probar é a seguinte: complementar a descrición dos obxectos etiquetados con información sobre os seus atributos (cor, patrón, material e reflectancia), ademais da categoría e posición, mellora a calidade da detección, en particular, nun contexto *few-shot*.

O obxectivo xeral do traballo é analizar os efectos da inclusión dos atributos nun detector de obxectos *few-shot*. Para cubrilo, propóñense os obxectivos concretos a continuación:

- **Obx. 1:** Definir un conxunto de datos *few-shot* para a detección de obxectos con anotacións sobre atributos.
- **Obx. 2:** Implementar un detector capaz de aprender con poucos exemplos, que poida determinar non só a categoría e a posición, senón que tamén poida predicir unha lista de atributos.
- **Obx. 3:** Estudar se a dispoñibilidade de exemplos de adestramento con información máis rica pode axudar á aprendizaxe, mellorando a localización e a clasificación fronte a un detector tradicional.

1.3. Estrutura da memoria

Segundo a versión do Regulamento do Traballo de Fin de Grao en Enxeñaría Informática que aplica para a convocatoria de xullo de 2023, este TFG enmárcase no Tipo A (Desenvolvemento dunha idea, prototipo ou modelado teórico dun Sistema Informático que constitúa unha contribución ás técnicas da Informática), polo que os contidos a partir deste primeiro capítulo organízanse da seguinte maneira:

No **capítulo 2**, correspondente co estado do coñecemento, revísanse brevemente algunhas das técnicas actuais relacionadas co problema a abordar.

No **capítulo 3** detállase a metodoloxía empregada, tratando o problema formalmente, examinando de forma exhaustiva as técnicas empregadas e xustificando as decisións de deseño aplicadas.

O **capítulo 4** comprende a descrición dos materiais, equipos informáticos e recursos software utilizados ao longo do traballo.

O **capítulo 5** correspóndese co plan de probas levado a cabo, que verifica a funcionalidade e correctitude global do experimento a través dos resultados.

No **capítulo 6** realízase a discusión de resultados, onde se comproba a hipótese de partida.

O **capítulo 7** serve de conclusión, onde se resumen as principais aportacións do traballo e posibles liñas de traballo futuro.

Capítulo 2

Estado do coñecemento

2.1. Detección de obxectos

Unha das aportacións máis importantes nos últimos anos para a evolución da detección de obxectos en imaxes foi **R-CNN** [6], o primeiro detector que empregou con éxito **redes neuronais convolucionais** (CNNs) para extraer características complexas das imaxes de entrada. Esta iniciativa supuxo un cambio radical que mellorou notablemente os resultados respecto ás aproximacións previas, comunmente baseadas en características máis simples, como por exemplo, as obtidas a partir de histogramas de gradientes orientados (HOGs) [14].

R-CNN forma parte do que se coñece coma **detectores de 2 etapas**, que se caracterizan por xerar, en primeiro lugar, **rexións de interese (RoIs)**, seccións da imaxe con altas probabilidades de conter un obxecto, para posteriormente clasificar a categoría e axustar o cadro delimitador (*bounding box*) por medio de regresión. En particular, R-CNN extrae as características de cada proposta procesándoas indivualmente a través dunha CNN, o cal pode chegar a ser moi custoso e, a continuación, clasifica as rexións utilizando unha máquina de vectores de soporte (SVM) linear.

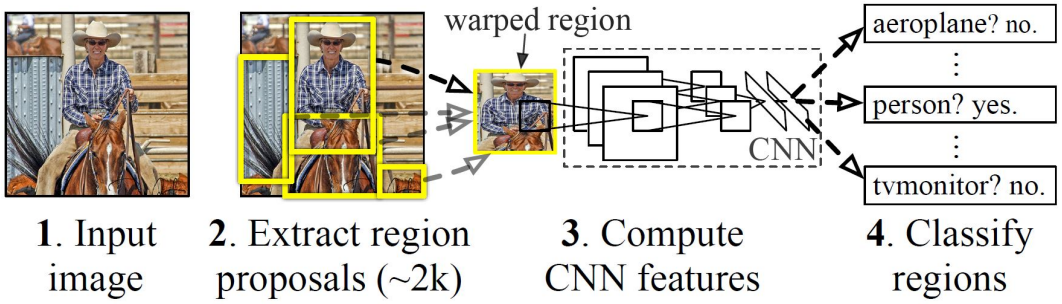


Figura 2.1: Arquitectura de R-CNN. Imaxe extraída de [6].

R-CNN sentou as bases dos posteriores avances nesta materia e un dos seus

sucesores foron as **SPPnets** [15], propostas para mellorar o rendemento mediante a obtención dun único mapa de características da imaxe completa, construído coa axuda dunha CNN. Non obstante, o feito de contar con adestramentos illados para a extracción de características, clasificación con SVMs e regresión dos cadros delimitadores, supoñía unha gran restrición en canto ao rendemento.

Para solucionar este problema xurdiu **Fast R-CNN** [7], que ademais de computar un só mapa de características para a imaxe completa, unificou o adestramento combinando o erro das múltiples tarefas.

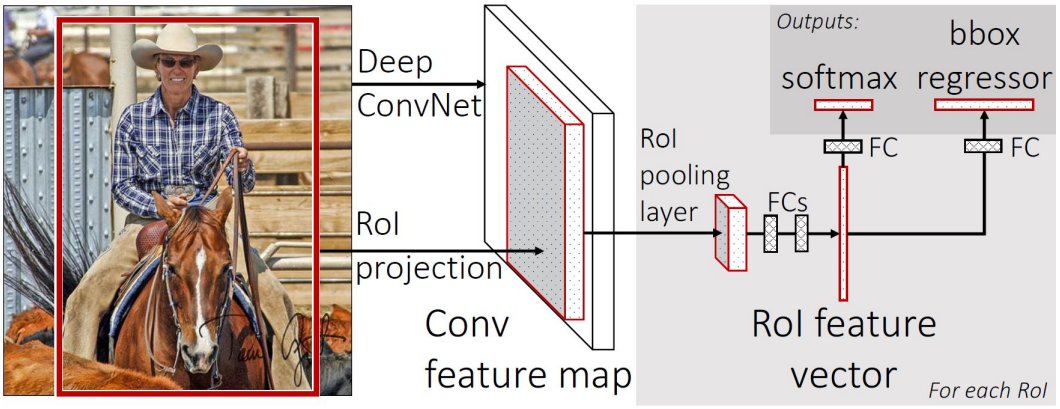
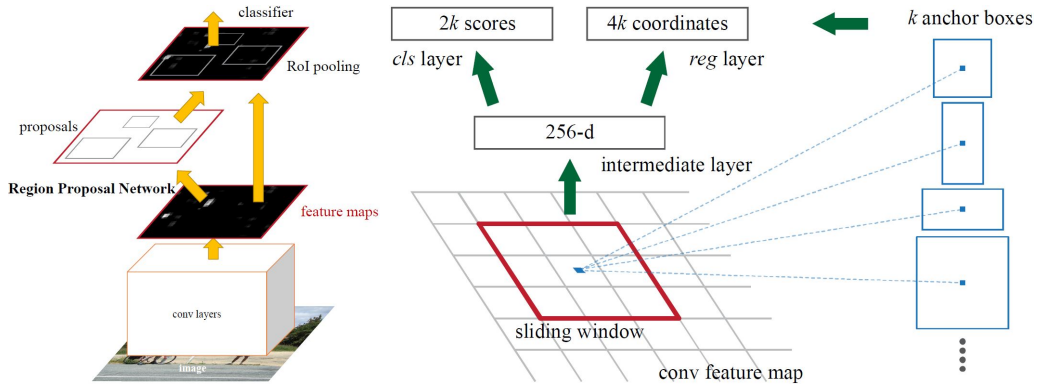


Figura 2.2: Arquitectura de Fast R-CNN. Imaxe extraída de [7].

Coa aparición de Fast R-CNN, a maior limitación a nivel computacional pasou a ser a proposta de rexións de interese, que habitualmente se levaba a cabo mediante técnicas como Selective Search [16] ou EdgeBoxes [17], que consumían unha parte importante do tempo das deteccións. **Faster R-CNN** [3] introduciu un novo mecanismo para a xeración de propostas chamado *Region Proposal Network* (**RPN**), que aproveita os mapas de características extraídos pola CNN encargada da extración de características (*backbone*) e engade algunha capa convolucional adicional para predicir as RoIs a partir de cadros predefinidos (*anchors*), o que resulta nun proceso cun custo computacional moi inferior ao dos métodos anteriores. A figura 2.3 amosa como se integra o módulo RPN na arquitectura do detector.

En contraposición aos detectores baseados en propostas, existen os **detectores de 1 etapa**, que simplifican o proceso de detección prescindindo precisamente da fase de obtención das propostas, o que os converte en métodos moito máis áxiles. Neste ámbito, un dos principais expoñentes é **YOLO** [18], que emprega unha única CNN para predicir os cadros deliminitadores e as categorías directamente a partir da imaxe, a gran velocidade, pero de forma menos precisa en comparación coas técnicas xa revisadas. Outra alternativa é **SSD** [19], que consegue deteccións aínda máis rápidas que YOLO e mellores precisións grazas á utilización de marcos por defecto. Este tipo de métodos son útiles para aplicacións en tempo real.

A pesar de que este traballo está centrado nos detectores baseados en redes



(a) Arquitectura de Faster R-CNN

(b) Arquitectura de RPN

Figura 2.3: Faster R-CNN. Imaxes extraídas de [3].

neuronais convolucionais, cabe destacar que recentemente xurdiu outra liña de investigación enfocada nos detectores baseados en ***vision transformers*** (ViT) [20]. Algúns dos detectores máis importantes neste ámbito son **DETR** [21], **Deformable DETR** [22], **DetReg** [23] ou **imTED** [24], entre outros. Aínda que se alcanzaron resultados moi competentes no campo da detección de obxectos, dado o custo computacional de executar estes modelos, neste traballo non se exploran as alternativas deste tipo.

2.2. Detección de obxectos *few-shot*

A **aprendizaxe *few-shot*** é un tipo de aprendizaxe automática que consiste en aprender conceptos nunca antes vistos empregando moi pouca información de adestramento sobre os mesmos.

En particular, a **detección de obxectos *few-shot*** ten como obxectivo a detección de obxectos pertencentes a categorías das que se dispón dun número moi reducido de exemplos de adestramento. Neste problema interveñen dous conxuntos disxuntos de categorías: as categorías base, para as que se conta cun gran número de imaxes etiquetadas (conxunto de datos base) e as categorías novas, para as que se dispón dunha mostra moito máis pequena de imaxes etiquetadas (conxunto de datos novel).

Actualmente, neste campo séguense principalmente dous paradigmas: a **meta-aprendizaxe** (*meta-learning*) e o **axuste fino** (*fine-tuning*). Por un lado, a meta-aprendizaxe busca "aprender a aprender", que no dominio *few-shot* significa aprender unha función de similaridade que calcule a distancia entre o obxecto que se quere clasificar e o soporte de cada unha das categorías; de forma que, a partir desa similaridade e, utilizando diferentes técnicas, se determine a clase do

obxecto. En cambio, o axuste fino é unha técnica máis simple dentro da **transferencia de coñecemento** (*transfer learning*) [25] que, neste ámbito, consiste en adestrar un modelo empregando o conxunto de datos base, para despois axustar lixeiramente os seus pesos empregando o conxunto novel.

Xa existen numerosos detectores que tratan de resolver o problema da detección de obxectos *few-shot*, en primeiro lugar, revísanse aqueles enfocados na meta-aprendizaxe. Un exemplo de detector dunha única etapa é **Meta-YOLO** [26], que extrae meta-características a partir do conxunto de datos base e posteriormente as pondera (utilizando a mostra novel) en función da súa importancia, para mellorar a detección sobre as clases novas. Por outra banda, como exemplos de detectores de 2 etapas destacan [27, 28, 29]. **Meta R-CNN** [27], meta-aprende a ponderar as características das RoIs, en vez das da imaxe completa. **FSDetView** [28], pola súa parte, veu a resolver o problema que supón a ponderación de características de clases novel ruidosas que confundan ao detector. Para elo, combina información das características das imaxes de consulta (imaxes sobre as que se aplica o detector) e das representativas das clases. **FewX** [29], a diferencia dos métodos anteriores, trata de mellorar as propostas de RoIs meta-aprendendo un mecanismo para obter similitudes entre os mapas de características das imaxes de consulta e as de soporte (imaxes etiquetadas con obxectos de categorías novas) e, desta forma, reducir os falsos positivos no fondo.

Tamén existen algunhas alternativas centradas na transferencia de coñecemento, como por exemplo **LSTD** [30], un modelo que extrapola o coñecemento do conxunto de datos base ao conxunto novel minimizando as diferenzas entre ambos. Outra alternativa é **TFA** [31], que propón simplemente realizar un axuste fino sobre a última capa dos detectores existentes empregando o conxunto de datos novel, obtendo resultados comparables ou incluso superiores aos mellores métodos baseados en meta-aprendizaxe. Por último, cabe destacar o caso de **DeFRCN** [2], un detector que ten un papel moi importante no desenvolvemento deste traballo e que tamén segue a idea do axuste fino. Este detector está cimentado sobre Faster R-CNN, ao que engade dous módulos adicionais para desacoplar RPN (independente da categoría) da cabeceira do detector (dependente da clase), e para refinar os resultados da clasificación empregando un modelo preadestrado.

2.3. Detección de obxectos con atributos

A detección de obxectos con atributos é unha liña de investigación máis recente que a detección *few-shot* e, polo tanto, non tan estendida como esta. Non obstante, nos últimos tempos presentouse PACO (*Parts and Attributes of Common Objects*) [1], un conxunto de datos de detección que conta con anotacións máis ricas que as dos conxuntos de datos tradicionais, xa que engade información sobre partes e atributos dos obxectos, ademais da categoría e da posición.

Este mesmo traballo veu acompompañado dunha cabeceira con soporte para

atributos que se pode empregar sobre distintos modelos. De feito, a experimentación realizada no propio estudo, probou detectores xa existentes baseados tanto en redes convolucionais, como en *vision transformers*, o que da unha idea da súa versatilidade. Cabe destacar que tanto o conxunto de datos PACO, coma a cabeceira adaptada para este, teñen un peso moi relevante neste traballo de investigación.

Capítulo 3

Metodoloxía

3.1. Definición do problema

No problema da detección de obxectos *few-shot*, interveñen dous conxuntos de datos: un *dataset* de gran tamaño con abundantes imaxes etiquetadas (D_{base}), e outro dataset dunha extensión moito menor (D_{novel}) con escasas de instancias anotadas por categoría. As categorías base (C_{base}) en D_{base} e as categorías novas (C_{novel}) en D_{novel}, non se solapan, é dicir, $C_{base} \cap C_{novel} = \varnothing$.

D_{novel} está conformado por un conxunto de *K-shots* asociados a cada categoría $C_i \in C_{novel}$, de xeito que un *shot* contén K obxectos anotados pertencentes a C_i. O tamaño dos *shots* (K) non está estrictamente determinado, con todo, debe tomar valores reducidos, habitualmente menores a 100. Seguindo investigacións previas, neste traballo defínese $K = \{1, 2, 3, 5, 10, 30\}$. A especificación dos *shots* de varios tamaños permite analizar como afectan as distintas cantidades de instancias (de clases novas) á aprendizaxe dos modelos.

Os detectores que tratan este problema aprenden a partir de ambos conxuntos de datos (D_{base} e D_{novel}). Non obstante, existen dúas alternativas en canto á avaliación dos resultados: FSOD (*Few-Shot Object Detection*) e G-FSOD (*Generalized* FSOD). A primeira, enfócase unicamente no rendemento sobre C_{novel}, mentres que a segunda realiza a avaliación sobre $C_{base} \cup C_{novel}$, é dicir, contempla a capacidade dos modelos de non esquecer as clases base. Este traballo céntrase en FSOD.

3.2. Conxunto de datos *few-shot* con atributos

Dado que este traballo se enmarca nun escenario *few-shot*, podería ser factible anotar manualmente os atributos dunhas decenas de obxectos, non obstante, precísase dun conxunto de test o suficientemente grande como para lograr uns resultados consistentes a partir dos que extraer conclusións. É por este motivo que se parte de PACO (*Parts and Attributes of Common Objects*) [1] para levar

a cabo a construción dun conxunto de datos *few-shot* orientado á detección de obxectos con atributos. PACO inclúe anotacións de partes de obxectos e atributos sobre conxuntos de datos de imaxe e de vídeo, non obstante, neste traballo non se utiliza a información de vídeo nin de partes de obxectos.

Concretamente, as anotacións de atributos (sobre obxectos en imaxes) que proporciona PACO, están incluídas nun subconxunto das anotacións do *dataset* LVIS [32]. A porción das anotacións de LVIS enriquecidas con información acerca de atributos, coñécese coma PACO-LVIS, que é o conxunto de datos que se utiliza neste traballo. Ademais, cabe resaltar que todas as anotacións de PACO-LVIS fan uso de imaxes pertencentes ao *dataset* COCO 2017 [5].

O conxunto de datos sobre o que se traballa contén 55 atributos diferentes que poden ser asignados a calquera obxecto, estes 55 valores están divididos en 4 tipos de atributos: cor, patrón, material e reflectancia. Na táboa 3.1 amósase o vocabulario completo.

Tipo de atributo	**Atributos**
Cor	*black, light_blue, blue, dark_blue, light_brown, brown, dark_brown, light_green, green, dark_green, light_grey, grey, dark_grey, light_orange, orange, dark_orange, light_pink, pink, dark_pink, light_purple, purple, dark_purple, light_red, red, dark_red, white, light_yellow, yellow, dark_yellow*
Patrón	*plain, striped, dotted, checkered, woven, studded, perforated, floral, logo, text*
Material	*stone, wood, rattan, fabric, crochet, wool, leather, velvet, metal, paper, plastic, glass, ceramic*
Reflectancia	*opaque, translucent, transparent*

Táboa 3.1: Taxonomía dos atributos.

Para adaptar este *dataset* ao problema da detección de obxectos *few-shot*, é necesario levar a cabo unha partición que separe as categorías base das categorías novas, de xeito que ambos grupos sexan disxuntos. Respecto ás categorías base, usaranse (no adestramento do detector) todas as imaxes etiquetadas das que dispón o conxunto de adestramento de PACO-LVIS, pola contra, para as categorías novas, farase unha selección aleatoria de varios soportes cun número moi reducido de imaxes etiquetadas, coñecidos como *shots*, que suporán a única información a partir da que o detector aprenda a recoñecer obxectos das categorías nunca antes vistas.

En primeiro lugar, débese determinar a **división entre clases base e novas**, para elo, tomouse como referencia a partición que habitualmente se realiza sobre o *dataset* COCO. Este conxunto de datos conta con 80 categorías que se separan en 60 base e 20 novas, onde as categorías novas coinciden coas clases do *dataset*

PASCAL VOC [33]. PACO-LVIS, pola súa parte conta con 75 categorías que non se corresponden coas de COCO, polo que non se pode replicar esta partición directamente. Non obstante, selecciónanse tamén 20 clases novas, deixando como base as 55 restantes.

Antes de levar a cabo a selección, establecéronse unha serie de condicións necesarias para que unha categoría poida pertencer ao conxunto novel:

- As categorías novas deben contar, polo menos, con 30 instancias no conxunto de adestramento para poder configurar un *shot* de dito tamaño.
- As categorías novas deben contar, polo menos, con 100 instancias no conxunto de test, de xeito que se evita que unha única predición (acertada ou errónea) teña unha repercusión demasiado alta nos resultados da avaliación.
- As categorías novas non deben estar estreitamente relacionadas con ningunha das clases base. Desta maneira, trátase de atenuar a posible influencia da aprendizaxe dunha categoría base sobre a aprendizaxe dunha clase novel.

O procedemento de selección de clases novas levado a cabo foi o seguinte: primeiramente, seleccionáronse aquelas categorías coincidentes con algunha das de PASCAL VOC, asegurando que cumprisen coas restricións definidas. A continuación, dado que non foron suficientes como para completar o conxunto de 20 clases novas, seleccionáronse aquelas categorías cun menor número de instancias de adestramento (cumprindo igualmente as condicións mencionadas), así, dado que só se utilizan, como máximo, 30 instancias, procurouse desperdiciar a menor cantidade de información posible.

Seguindo o procedemento anterior e, dada a distribución de instancias de adestramento e test por clase, na táboa 3.2 amósanse as estadísticas de todas as categorías e os resultados da selección novel en negriña.

Unha vez escollidas as categorías novas, selecciónanse aleatoriamente instancias de cada unha delas para confeccionar os *shots* de tamaños: 1, 2, 3, 5, 10 e 30. Para completar esta tarefa, seguiuse o método descrito en para COCO en [31].

3.3. DeFRCN

Decidiuse empregar DeFRCN [2] como punto de partida para levar a cabo unha das tarefas esenciais deste traballo, a construción dun detector de obxectos *few-shot* capaz de predicir, ademais da localización e categoría dos obxectos, unha lista de atributos. Escolleuse este detector debido a que logra resultados de alta calidade no problema da detección *few-shot* ao mesmo tempo que mantén unha arquitectura considerablemente sinxela en comparación con outras alternativas, o que reduce as necesidades de cómputo para levar a cabo este traballo.

Categoría	Nº instancias adestramento	Nº instancias test	Categoría	Nº instancias adestramento	Nº instancias test
ball	**739**	**107**	**microwave**	**1054**	**191**
basket	3758	731	**mirror**	**3347**	**573**
belt	3464	676	**mouse**	**1753**	**384**
bench	4165	807	mug	1695	368
bicycle	**4344**	**969**	napkin	3814	918
blender	301	57	**newspaper**	**1137**	**202**
book	32249	7022	pan	604	242
bottle	7554	1766	pen	897	237
bowl	4993	940	pencil	530	83
box	7501	1389	pillow	5838	1290
broom	143	20	pipe	4500	786
bucket	1296	273	plate	4992	1042
calculator	58	27	pliers	49	24
can	1356	343	remote_control	2301	483
car	**9978**	**2349**	plastic_bag	3492	738
carton	201	36	**scarf**	**1250**	**259**
cellular_phone	2743	575	scissors	1299	269
chair	10970	2368	screwdriver	87	64
clock	**2564**	**622**	shoe	8849	1970
crate	1757	403	slipper	118	30
cup	4415	834	**soap**	**841**	**263**
dog	**2542**	**531**	sponge	114	18
drill	23	2	**spoon**	**1996**	**426**
drum	56	13	stool	558	128
earphone	**732**	**143**	**sweater**	**1794**	**365**
fan	**708**	**156**	**table**	**2647**	**595**
glass	6121	1205	tape	551	42
guitar	301	52	telephone	906	203
hammer	35	13	television_set	2097	397
handbag	3723	713	tissue_paper	575	84
hat	6661	1357	**towel**	**2077**	**427**
helmet	4593	884	trash_can	2585	548
jar	1856	304	tray	2272	472
kettle	125	31	vase	4748	1055
knife	**3351**	**735**	wallet	119	29
ladder	**947**	**167**	**watch**	**2581**	**551**
lamp	3950	854	wrench	75	80
laptop	2702	551			

Táboa 3.2: Distribución de instancias por clase. En negriña, as categorías novas.

Tendo en conta que DeFRCN foi un pilar fundamental no desenvolvemento deste traballo, é imprescindible coñecer en profundidade as súas características, funcionamento e aportacións.

DeFRCN é un detector *few-shot* cuxo procedemento de aprendizaxe está baseado no paradigma do axuste fino. Este procedemento consiste nunha primeira etapa de extracción de coñecemento (transferible) a partir do abondoso conxunto de datos D_{base}, trala que se realiza unha segunda fase de adaptación rápida sobre o soporte D_{novel}, que dispón dunha cantidade moi baixa de instancias para cada unha das categorías nunca antes vistas.

Este detector está construído en base a Faster R-CNN [3], sobre o que propón un par de módulos adicionais para resolver certos problemas. Antes de introducir as problemáticas, convén lembrar o funcionamento de Faster R-CNN: en primeiro lugar, as imaxes son procesadas a través dunha rede convolucional (*backbone*) encargada de extraer un mapa de características que será compartido como entrada de RPN (*Region Proposal Network*) e da cabeceira do detector. Pola súa parte, RPN xera un conxunto de rexións de interese (RoIs independentes da clase) a través da clasificación (segundo as probabilidades de conter ou non un obxecto) e regresión (axuste das dimensións) de cadros predefinidos (*anchors*). Finalmente, a cabeceira computa a saída do detector facendo uso do mapa de características compartido e máis das RoIs; concretamente, clasifica a categoría do obxecto e axusta de novo as dimensións do cadro delimitador. En canto á optimización, o erro total calcúlase como a combinación do erro de RPN e da cabeceira, tal e como se amosa a continuación:

$$\mathcal{L}_{total} = (\mathcal{L}^{cls}_{rpn} + \mathcal{L}^{reg}_{rpn}) + \eta \cdot (\mathcal{L}^{cls}_{head} + \mathcal{L}^{reg}_{head}) \tag{3.1}$$

onde η representa o hiperpámetro que balancea o peso do erro de cada un dos compoñentes, que, á súa vez, se calcula como a suma do erro de clasificación e regresión en cada caso. Na figura 3.1 ilústranse o fluxo de información cara adiante (mapas de características e propostas) e a propagación cara atrás do erro a través do gradiente.

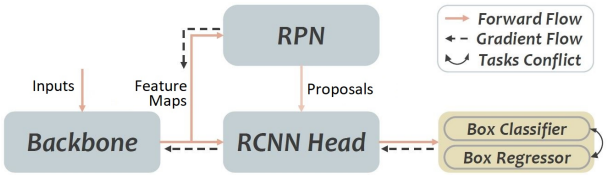


Figura 3.1: Fluxo de información en Faster R-CNN. Imaxe extraída de [2].

A presenza do ***backbone* compartido** resulta problemática debido a que a optimización de dito compoñente emprega información do erro de RPN (independente da categoría) e da cabeceira (dependente da categoría), que pode chegar a ser contraditoria. Por outra banda, a **aprendizaxe multitarefa** (localización e clasificación dos obxectos) representa outro inconveniente, xa que, por unha banda, a parte da cabeceira encargada da localización require características invariantes á translación, xa que trata de recoñecer obxectos independentemente

da súa posición na imaxe; pola contra, a parte da cabeceira dedicada á clasificación necesita características covariantes á translación, para poder determinar a ubicación dos obxectos con exactitude.

DeFRCN resolve o problema derivado do *backbone* compartido implementando as ***Gradient Decoupled Layers*** (GDLs) e a problemática correspondente á aprendizaxe multitarefa a través do ***Prototypical Calibration Block*** (PCB). Transforma o fluxo de información tal e como se amosa na figura 3.2 integrando en Faster R-CNN os módulos mencionados da maneira en que se ilustra na figura 3.3.

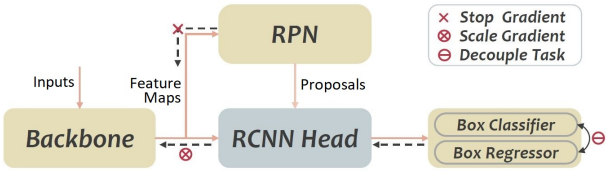


Figura 3.2: Fluxo de información en DeFRCN. Imaxe extraída de [2].

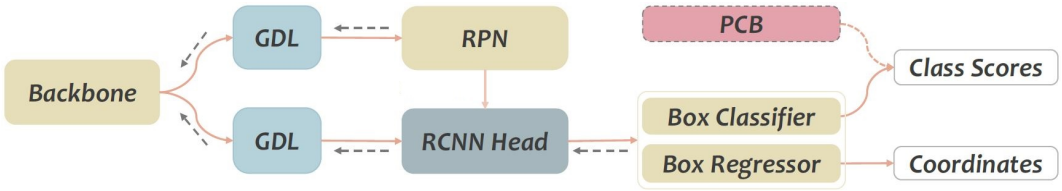


Figura 3.3: Incorporación das GDLs e do PCB sobre Faster R-CNN. As liñas laranxas indican fluxo de información cara adiante e as liñas de puntos, o fluxo do gradiente cara atrás. Imaxe extraída de [2].

3.3.1. *Gradient Decoupled Layers*

Como se pode observar na figura 3.3, introdúcense dúas GDLs que permiten realizar diferentes tratamentos de información entre o *backbone* e RPN, así como entre o *backbone* e a cabeceira. Cada unha das GDLs é un compoñente realmente simple que segue o esquema da figura 3.4.

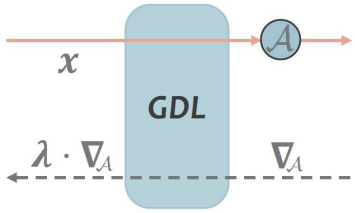


Figura 3.4: Esquema dunha *Gradient Decoupled Layer*. Imaxe extraída de [2].

Durante a propagación da información cara adiante, os mapas de características computados pola CNN son procesados a través dunha capa densamente conectada cun parámetro adestrable por cada canle e máis un *bias* (na figura 3.4 esta capa represéntase como $\mathcal{A}$). Desta forma, créanse dous espazos de características distintos, adaptados ás necesidades de RPN e da cabeceira, xa que a optimización realízase, no caso de $\mathcal{A}_{rpn}$ empregado o erro de RPN e, no caso de $\mathcal{A}_{head}$, empregando o erro da detección.

Por outra banda, na retropropagación, a única operación que se realiza é o produto do gradiente procedente da capa posterior (sexa RPN ou a cabeceira), por unha constante $\lambda \in [0, 1]$, cuxo resultado se transmite directamente cara a capa predecesora, neste caso, o *backbone*. Polo tanto, a optimización dos parámetros adestrables do *backbone*, representados por θ_b, ven descrita pola seguinte ecuación:

$$\theta_b \leftarrow \theta_b - \gamma \left(\lambda_1 \frac{\partial \mathcal{L}_{rpn}}{\partial \theta_b} + \lambda_2 \frac{\partial \mathcal{L}_{head}}{\partial \theta_b} \right) \tag{3.2}$$

onde γ é a taxa de aprendizaxe e λ_1 e λ_2, os coeficientes de desacoplamento para RPN e para a cabeceira respectivamente.

Tal e como se pode observar na figura 3.2, o gradiente procedente de RPN descártase ($\lambda_1 = 0$), co que o erro na proposta de rexións de interese non se ten en conta para a optimización da CNN compartida. Por outra banda, o gradiente proveniente da cabeceira, escálase (en diferentes proporcións para o adestramento sobre D_{base} e para o axuste fino empregando D_{novel}; máis detalles sobre a configuración dos parámetros no apartado 5.1.3).

3.3.2. *Prototypical Calibration Block*

O módulo PCB foi introducido para refinar a confianza das deteccióǹs facendo uso dun clasificador preadestrado. Este compoñente segue o esquema presentado na figura 3.5.

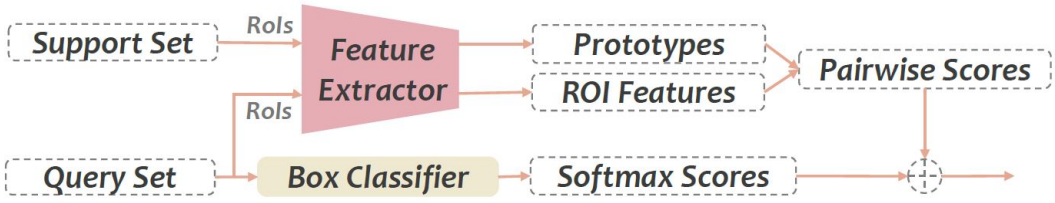


Figura 3.5: Esquema do módulo *Prototypical Calibration Block*. Imaxe extraída de [2].

A idea básica do PCB é axustar, na fase de inferencia, a confianza das deteccióǹs de obxectos pertencentes a algunha categoría $c \in C_{novel}$, comparando o mapa de características do obxecto detectado, cos mapas de características dos obxectos desa mesma categoría dispoñibles no soporte (D_{novel}). Para levar a cabo

esta idea, utilízase o extractor de características dun clasificador preadestrado sobre o conxunto de datos ImageNet [34].

O procedemento a seguir é o seguinte: dada unha detección $y'_i = (c_i, b_i, s_i)$, onde c_i é a categoría predita, b_i o cadro delimitador predito e s_i a confianza correspondente; en primeiro lugar extráense as características (x_i) da RoI que contén o obxecto detectado (na figura 3.5, estas son as *ROI Features*). A continuación, extráense as características daquelas rexións das imaxes do soporte $\mathcal{S}$ que conteñen obxectos da clase c_i , ditas rexións deben atravesar unha capa de *RoIAlign* [11] para homoxeneizar as súas dimensións. Os mapas de características obtidos a partir do soporte agréganse a través da media de todos eles e o resultado son os prototipos (p_i). Unha vez calculados x_i e p_i, compútase a similaridade coseno como:

$$sim = \frac{x_i \cdot p_i}{\|x_i\|\|p_i\|} \tag{3.3}$$

Finalmente, refínase a confianza (*score*) seguindo:

$$s'_i = \alpha \cdot s_i + (1 - \alpha) \cdot sim \tag{3.4}$$

sendo α un hiperparámetro de balanceo. Desta forma, dado que o módulo PCB non comparte ningún parámetro co resto do detector, desacóplanse as tarefas de clasificación e localización, mellorando o rendemento nun contexto *few-shot*.

3.3.3. Arquitectura

Tal e como se mencionou anteriormente, DeFRCN emprega Faster R-CNN coma marco básico de detección, non obstante, isto non supón unha restrición á hora de configurar a arquitectura específica do detector.

Neste caso, utilízase **ResNet-101** [8] como extractor de características, unha arquitectura que permite construír redes convolucionais profundas de alta calidade grazas á introdución de "atallos" (do inglés *shortcut connections*). Na figura 3.6 amósase o diagrama dun bloque residual, que contén cunha conexión que salta capas contiguas co obxectivo de reducir o efecto do problema do desvanecemento de gradiente [35], unha dificultade que se manifesta durante a retropropagación do erro. Neste proceso, cada un dos pesos da rede recibe unha actualización proporcional á derivada parcial da función de custo respecto ao propio peso e, dado que os pesos das capas iniciais en redes profundas teñen moi pouca influencia, sofren cambios moi sutís, co que se dificulta a converxencia do adestramento deste tipo de redes.

Na táboa 3.3 especifícase o esquema de ResNet-101. Como se pode observar, interveñen principalmente dous tipos de capas: a **convolución**, encargada da extracción de características e o ***pooling***, responsable da diminución da dimensionalidade dos datos a nivel de anchura e altura.

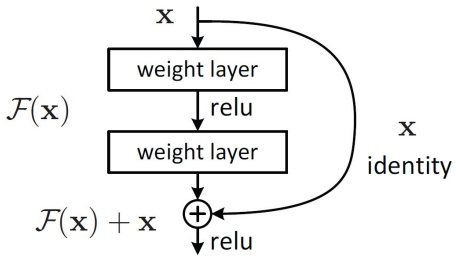


Figura 3.6: Bloque residual cun salto que realiza un mapeo de identidade. Imaxe extraída de [8].

Por unha banda, o ***pooling*** consiste basicamente na agrupación de grupos de valores adxacentes nun único valor, o cal se consegue mediante o uso de filtros de tamaño $f \times f$ (sen parámetros adestrables) que se desprazan polo mapa de características (segundo un *stride* determinado) calculando unha función de agregación sinxela, como por exemplo o máximo (*Max Pool*), ou a media (*Average Pool*). Na figura 3.7 amósase un exemplo.

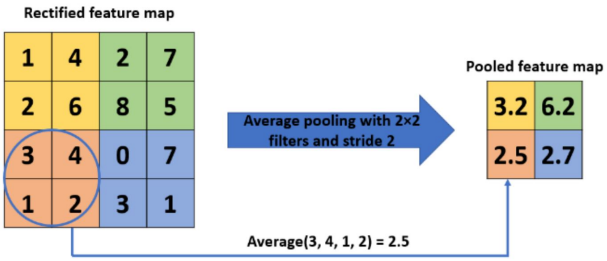


Figura 3.7: *Average Pool* utilizado un filtro 2×2 con *stride* 2. Esta configuración reduce á metade o tamaño de calquera mapa de características. Imaxe extraída de [9].

Por outra banda, nas **capas convolucionais**, aplícanse un conxunto de filtros de tamaño $f \times f$ (con parámetros adestrables) que se desprazan ao longo da imaxe, calculando o produto escalar entre o filtro e unha porción do mapa de características, cuxo resultado é procesado por unha función de activación. Esta operación formalízase tal e como amosa a ecuación 3.5.

$$Y_j = g\left(b_j + \sum_i K_{ij} \circledast Y_i\right) \tag{3.5}$$

onde Y_j é a saída da neurona j, g é unha función de activación non lineal (neste caso ReLU [36]), b_j é un *bias* e K_{ij} representa o filtro aplicado sobre a saída da neurona anterior (Y_i) a través da convolución (denotada coma $\circledast$).

Polo tanto, trala convolución, obténse outro mapa características cunha profundidade igual ao número de filtros aplicados. Esta operación ilústrase en 3.8.

block name	output size	101-layer
conv1	112×112	7×7, 64, stride 2
conv2_x	56×56	3×3 max pool, stride 2
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5_x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool

Táboa 3.3: Arquitectura ResNet-101. Táboa extraída de [8].

Dada a configuración da rede (que aplica, en cada bloque, un maior número de filtros sobre mapas de características máis reducidos), a medida que se procesan os datos, as capas máis alonxadas da entrada son activadas por características cada vez máis complexas.

Relacionado coa convolución, ResNet presenta bloques "pescozo de botella" (do inglés *bottleneck*), que, como se pode observar na táboa 3.3, consisten en tres capas con filtros 1×1, 3×3 e 1×1 respectivamente, onde os filtros 1×1 son utilizados, no primeiro caso, para reducir a profundidade dos datos e, no segundo caso, para restablecela, de xeito que se reduce a complexidade da convolución realizada polo filtro 3×3.

Retomando a arquitectura específica de DeFRCN, non todos os bloques de ResNet-101 se empregan na extracción de características da imaxe completa, realmente, o *backbone* contén só os catro primeiros bloques (conv1, conv2_x, conv3_x, conv4_x), deixando o último (conv5_x) para a extracción de características das rexións de interese (tralo *RoI Pooler*), co que forma parte da cabeceira do detector.

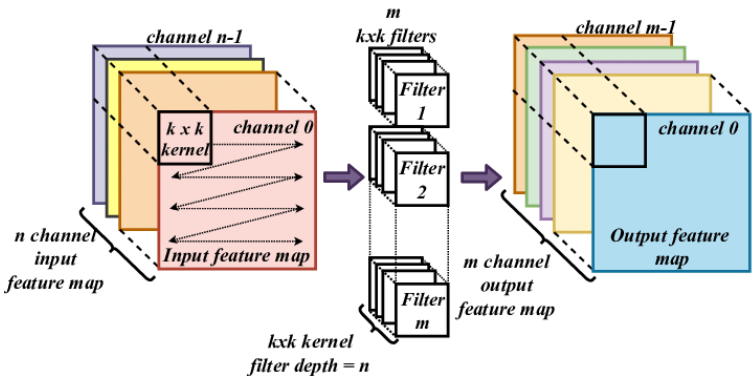


Figura 3.8: Convolución de m filtros sobre un mapa de características de profundidade n. Imaxe extraída de [10]

A imaxe 3.9 ilustra a arquitectura de DeFRCN.

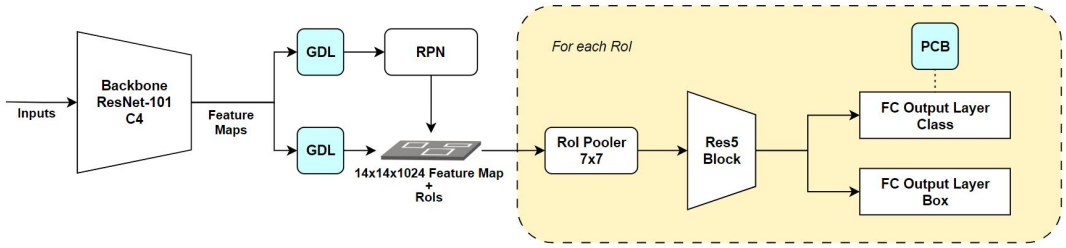


Figura 3.9: Arquitectura do detector DeFRCN.

Cabe destacar que o obxectivo do *RoI Pooler* é producir, para calquera RoI (independentemente das súas dimensións), un mapa de características de tamaño fixo. É necesario unificar o tamaño dos mapas de características para que poidan ser utilizados como entradas dos seguintes compoñentes da cabeceira.

Neste caso particular, o tamaño resultante é de 7×7, tal e como se indica na figura 3.9. A pesar de que o compoñente encargado desta tarefa é comunmente coñecido como *RoI Pooler*, a técnica que utiliza DeFRCN é *RoIAlign*.

A diferencia do *RoIPool*, que plasma as RoIs sobre o mapa de características axustándose á cuadrícula do mesmo, para posteriormente agregar os valores adxacentes (habitualmente mediante o máximo), *RoIAlign* mantén as proporcións das RoIs intactas, e utiliza interpolación bilineal [37] para computar catro valores sobre os que agregar o resultado, conseguindo unha precisión superior. Un diagrama desta operación amósase en 3.10.

3.4. Detector de PACO

Tal e como se adiantou na sección 2.3, o traballo de investigación que deu lugar ao conxunto de datos PACO [1], empregado neste traballo, aportou tamén

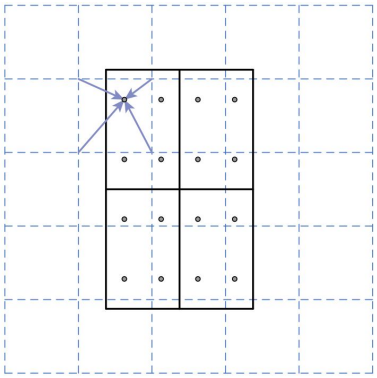


Figura 3.10: RoIAlign: a liña descontinua representa un mapa de características e a liña continua representa unha RoI cuxo tamaño se fixa a 2×2 agregando os catro puntos calculados mediante interpolación bilineal en cada "celda". Imaxe extraída de [11].

unha cabeceira con soporte para a predición de atributos que se pode utilizar sobre distintos detectores xa existentes. Dita cabeceira foi, xunto con DeFRCN, un elemento de grande axuda para o cumprimento do segundo dos obxectivos específicos presentados na sección 1.2. A figura 3.11 presenta un diagrama da súa arquitectura.

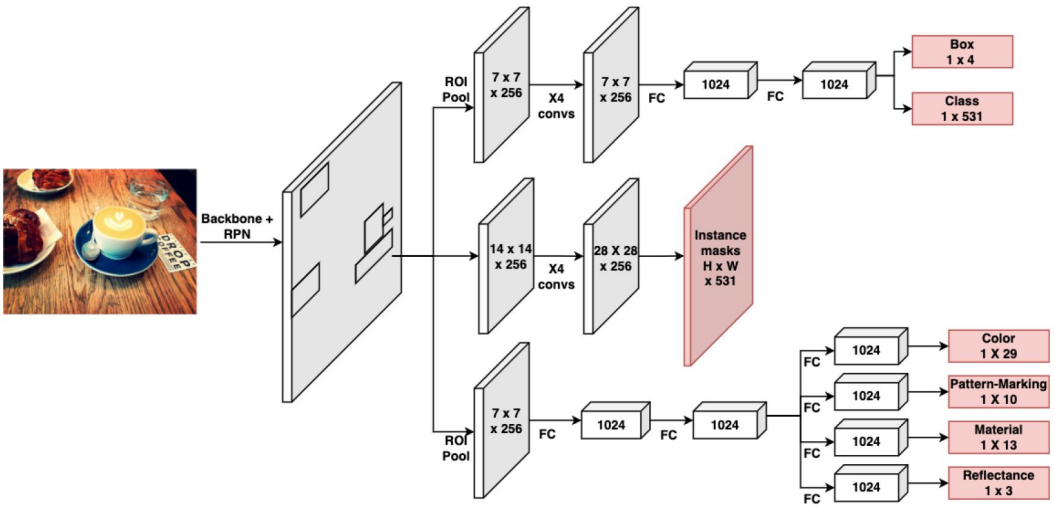


Figura 3.11: Arquitectura da cabeceira de PACO. Imaxe extraída de [1].

Como se pode observar, a cabeceira dispón de tres partes claramente diferenciadas. A primeira delas (en orde descendente) é a encargada da detección de obxectos (clasificación e localización), a segunda está dedicada á segmentación, un problema que non se trata ao longo deste traballo, o cal consiste na división de imaxes en distintas rexións a través dunha clasificación a nivel de píxel. Por

último, a terceira ramificación é a encargada da predición dos atributos, a única que se aproveita no desenvolvemento deste traballo.

Respecto á predición dos atributos, en primeiro lugar, os datos son transformados polo *RoI Pooler*, que neste caso tamén aplica *RoIAlign*. Posteriormente, son procesados secuencialmente a través de dúas capas densamente conectadas (*Fully Connected*, FC) de 1024 neuronas. Finalmente, a información rediríxese cara as capas FC de saída, cada unha destinada a un dos catro tipos de atributos que ten PACO. Estas capas de saída dispoñen de tantas neuronas coma valores posibles para o tipo de atributo co que se corresponden, máis outra neurona adicional, que se inclúe como representación do valor "outro" en cada caso.

3.5. Detector implementado

3.5.1. Arquitectura

A idea posta en práctica para a construción dun detector de obxectos *few-shot* coa capacidade de predicir atributos consiste, fundamentalmente, en acoplar, no modelo DeFRCN, a parte da cabeceira de PACO encargada da predición dos atributos.

Esta integración non é un procedemento inmediato. Para valorar a súa complexidade é importante ter en conta que esta tarefa non trata simplemente de substituír a cabeceira de DeFRCN pola de PACO, senón que a cabeceira de DeFRCN debe ser profundamente modificada para facer posible a súa división en dúas ramificacións paralelas, responsables da detección de obxectos e do soporte para os atributos respectivamente. Ademais, unha dificultade engadida é o feito de que o código de cada un dos detectores non utiliza as mesmas versións dalgunhas dependencias software, co que existen numerosas incompatibilidades que impiden unha transición directa entre compoñentes de ambos modelos. Na figura 3.12 móstrase a arquitectura proposta.

Como se pode apreciar, respéctanse as arquitecturas dos dous modelos combinados, xa que non é necesario alterar ningún dos elementos para completar a implementación do detector. De feito, o tamaño dos mapas de características producidos polo *RoI Pooler* é o mesmo (7×7), non obstante, impleméntanse dous compoñentes distintos (coa mesma configuración) para realizar dita operación; de xeito que se facilita calquera tipo de actualización no futuro que implique unha diferencia entre ambos.

Cabe destacar que, a diferencia do que se amosa na figura 3.11 para PACO, neste caso, os mapas de características procedentes do *backbone* contan con 1024 canles, non con 256.

Finalmente, é interesante repasar as capas de saída do modelo, xa que son as pezas encargadas de devolver os resultados. Seguindo a figura 3.12 en orde descendente, a primeira capa de saída correspóndese coa clasificación da categoría dos

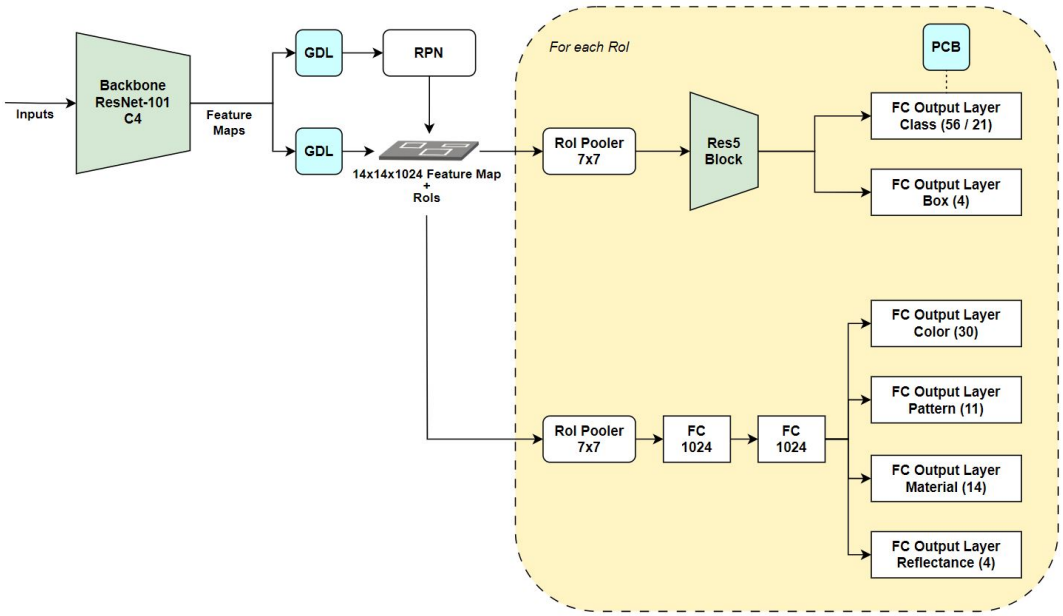


Figura 3.12: Arquitectura do detector implementado.

obxectos, terá 56 neuronas durante o adestramento e a avaliación base (tamaño de $C_{base} + 1$ neurona para determinar que se trata do fondo) e 21 neuronas durante o axuste fino e a avaliación novel (tamaño de $C_{novel} + 1$). Neste caso, os valores finais procésanse a través da función **SoftMax** para obter as probabilidades para cada unha das categorías posibles, esta función defínese como:

$$h_i(x, W, b) = \frac{e^{f_i(x,W,b)}}{\sum_{j=1}^{K} e^{f_j(x,W,b)}} \tag{3.6}$$

onde K é o número de neuronas e h_i representa a confianza da rede de que o obxecto pertenza á clase correspondente á neurona i, a cal recibe as entradas x e as procesa xunto cos pesos (W) e o *bias* (b) as través da operación $f(x, W, b)$. Así, a categoría con maior probabilidade será a predita polo detector coa confianza correspondente.

Seguindo a mesma orde, a segunda capa de saída é a que se ocupa do axuste das dimensións do cadro delimitador dos obxectos. Este cadro está expresado en formato XYXY [38], o cal está composto por 4 valores que representan os límites esquerdo, superior, dereito e inferior do cadro delimitador respectivamente. Dado que se trata dun problema de regresión de 4 cifras, esta capa resólveo directamente con 4 neuronas, cuxos valores non precisan de ningún tipo de transformación.

Finalmente, como xa se comentou no apartado previo, as catro capas de saída dedicadas á predición de cada un dos tipos de atributos teñen tantas neuronas como valores posibles, máis 1 para representar "outro". Do mesmo xeito que na clasificación da categoría, utilízase SoftMax para calcular as probabilidades. Con

todo, neste caso, dado que un mesmo obxecto pode ter asignados varios atributos do mesmo tipo, non interesa seleccionar unicamente aquel con maior confianza, senón recuperar as probabilidades de todos os atributos posibles.

3.5.2. Optimización

Unha vez repasadas as principais características da arquitectura do detector proposto, é conveniente coñecer como se leva a cabo a optimización do modelo. Na retropropagación do erro trátanse de atopar os parámetros óptimos de todos os compoñentes (os pesos denótanse como θ_b para o backbone, θ_{rpn} para RPN, θ_{head_det} para a parte da cabeceira correspondente á detección e θ_{head_att} para a parte da cabeceira correspondente á predición de atributos), que se actualizan seguindo as ecuacións 3.7, 3.8, 3.9 e 3.10 respectivamente.

$$\theta_b \leftarrow \theta_b - \gamma \left(\lambda_1 \frac{\partial \mathcal{L}_{rpn}}{\partial \theta_b} + \lambda_2 \frac{\partial (\mathcal{L}_{head\_det} + \mathcal{L}_{head\_att})}{\partial \theta_b} \right) \tag{3.7}$$

$$\theta_{rpn} \leftarrow \theta_{rpn} - \gamma \frac{\partial \mathcal{L}_{rpn}}{\partial \theta_{rpn}} \tag{3.8}$$

$$\theta_{head\_det} \leftarrow \theta_{head\_det} - \gamma \frac{\partial \mathcal{L}_{head\_det}}{\partial \theta_{head\_det}} \tag{3.9}$$

$$\theta_{head\_att} \leftarrow \theta_{head\_att} - \gamma \frac{\partial \mathcal{L}_{head\_att}}{\partial \theta_{head\_att}} \tag{3.10}$$

onde γ é a taxa de aprendizaxe, $\mathcal{L}_{rpn}$ é o erro de RPN, $\mathcal{L}_{head\_det}$ é o erro da sección da cabeceira correspondente á detección, $\mathcal{L}_{head\_att}$ é o erro da sección da cabeceira encargada da predición de atributos e λ_1, λ_2 son os coeficientes de desacoplamento de cada unha das GDLs.

Convén saber que o algoritmo de optimización empregado é o **descenso de gradiente estocástico** (SGD) [39], cun *mini-batch* de tamaño 16, o que significa que o detector procesa 16 imaxes antes de realizar a actualización dos pesos.

Para comprender en profundidade as ecuacións anteriormente descritas é necesario definir a función de custo de cada un dos compoñentes. En primeiro lugar, o erro de RPN sobre unha imaxe calcúlase como:

$$\mathcal{L}_{rpn}(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i \mathcal{L}_{rpn}^{cls}(p_i, p_i') + \frac{1}{N_{reg}} \sum_i p_i' \mathcal{L}_{rpn}^{reg}(t_i, t_i') \tag{3.11}$$

onde i representa o índice dun dos cadros predefinidos (*anchors*), p_i é a probabilidade predita de que o cadro i conteña un obxecto (p_i' é a probabilidade real, 0 ou 1) e t_i representa o axuste realizado sobre o cadro predefinido (t_i' é o cadro delimitador real). N_{cls} (tamaño do *mini-batch*) e N_{reg} (número de *anchors*) son

termos de normalización. O cálculo dos erros de clasificación ($\mathcal{L}_{rpn}^{cls}$) e de regresión ($\mathcal{L}_{rpn}^{reg}$) especifícase trala definición das funcións de custo, xa que é idéntico en todos os compoñentes.

Pola súa parte, o erro nas deteccións realizadas nunha imaxe calcúlase a través da seguinte función de custo:

$$\mathcal{L}_{head\_det}(\{p_i\},\{t_i\}) = \sum_i \mathcal{L}_{head\_det}^{cls}(p_i, p_i') + \mathcal{L}_{head\_det}^{reg}(t_i, t_i') \tag{3.12}$$

onde i representa, neste caso, o índice dunha das propostas realizadas por RPN, p_i contén as probabilidades preditas para cada unha das categorías posibles (p_i' é a clase real) e t_i representa o axuste realizado sobre o cadro da proposta i (t_i' é o cadro delimitador real).

Por último, a función de custo da sección da cabeceira correspondente á predición de atributos é a seguinte:

$$\begin{aligned}\mathcal{L}_{head\_att}(\{p_i^{cor}\},\{p_i^{pat}\},\{p_i^{mat}\},\{p_i^{ref}\}) = \sum_i \{ & \mathcal{L}_{head\_att}^{cor}(p_i^{cor}, p_i'^{cor}) \quad + \\ & \mathcal{L}_{head\_att}^{pat}(p_i^{pat}, p_i'^{pat}) \quad + \\ & \mathcal{L}_{head\_att}^{mat}(p_i^{mat}, p_i'^{mat}) \; + \\ & \mathcal{L}_{head\_att}^{ref}(p_i^{ref}, p_i'^{ref}) \quad \}\end{aligned} \tag{3.13}$$

onde i representa, de novo, o índice dunha das propostas realizadas por RPN, $p_i^{cor}, p_i^{pat}, p_i^{mat}$ e p_i^{ref} son as probabilidades preditas para todos os atributos correspondentes a cada un dos tipos (a notación $'$ fai referencia aos atributos reais anotados de cada tipo).

Para finalizar, é importante coñecer como se calculan os erros de clasificación e regresión. No caso da clasificación, que aplica a $\mathcal{L}_{rpn}^{cls}$ en 3.11, a $\mathcal{L}_{head\_det}^{cls}$ en 3.12 e a $\mathcal{L}_{head\_att}^{cor}, \mathcal{L}_{head\_att}^{pat}, \mathcal{L}_{head\_att}^{mat}, \mathcal{L}_{head\_att}^{ref}$ en 3.13, resólvese mediante **entropía cruzada** (do inglés *cross entropy*), que se calcula, para cada proposta ou detección, do seguinte xeito:

$$\mathcal{L}_{cls}(p_i, p_i') = -p_i' log(p_i) \tag{3.14}$$

Mentres que o erro de regresión, que aplica a $\mathcal{L}_{rpn}^{reg}$ en 3.11 e a $\mathcal{L}_{head\_det}^{reg}$ en 3.12, calcúlase mediante a función de custo **L1 suavizada** (*smooth L1*). A función L1 convencional consiste simplemente no valor absoluto da diferencia entre o valor real e o valor predito, mentres que a L1 suavizada calcúlase, para cada proposta ou detección, da seguinte maneira:

$$\mathcal{L}_{reg}(t_i, t_i') = smooth_{L_1}(t_i - t_i') \tag{3.15}$$

$$smooth_{L_1}(x) = \begin{cases} 0{,}5x^2 & se\ |x| < 1 \\ |x| - 0{,}5 & se\ |x| \geq 1 \end{cases} \tag{3.16}$$

Capítulo 4

Materiais

4.1. Configuración hardware

A totalidade das probas levadas a cabo executáronse de forma remota sobre dous servidores xestionados polo Centro Singular de Investigación en Tecnoloxías Intelixentes (CiTIUS), aos que se tivo acceso durante o desenvolvemento.

Un dos servidores, identificado como `ctgpgpu6`, é un Gigabyte G291-281 [40] con dous procesadores Intel Xeon Silver 4214 [41] e 192 GB de memoria RAM. A nivel de GPUs, dispón dunha Nvidia Quadro P6000 de 24 GB [42], unha Nvidia Quadro RTX 8000 de 48 GB [43] e dúas Nvidia A30 de 24 GB [44].

O outro servidor do que se dispuxo, `ctgpgpu11`, é un Gigabyte G482-Z54 [45] con dous procesadores AMD EPYC 7413 [46] e 256 GB de RAM. Respecto ás GPUs, conta con catro Nvidia A100 de 80 GB [47].

4.2. Configuración software

En canto ao sistema operativo, o servidor `ctgpgpu6` emprega CentOS 7 [48], mentres que `ctgpgpu11` utiliza AlmaLinux 9.1 [49].

Debido á natureza compartida dos servidores, así como ás numerosas dependencias que teñen as aplicacións desenvolvidas, optouse por empregar contedores de software, xestionados a través de Docker [50], nas versións 20.10.18 e 20.10.23 respectivamente. Deste xeito, tamén se unifican as versións dos programas auxiliares en ambos servidores.

Para poder explotar as vantaxes das GPUs fronte ás CPUs de propósito xeral, foi necesario contar con CUDA 11.3 [51], unha plataforma de computación que permite paralelizar os procesos a través do lanzamento de numerosos fíos simultáneos.

Respecto á colección de compiladores GNU (GCC) [52], empregouse a versión 9.4.0.

Referente ao código elaborado durante o traballo, salvo os scripts de Bash creados para a automatización de experimentos, todo o código está escrito en Python 3.8.10 [53]. Non obstante, utilizáronse múltiples paquetes adicionais, imprescinbles para o desenvolvemento. Algúns dos máis destacables son: NumPy 1.23.3 [54], PyTorch 1.10.0 [55], Detectron2 0.3 [4], OpenCV 4.7.0 [56], Sci-kit Learn 1.1.2 [57] e LVIS 0.5.3 [32], entre outros.

Capítulo 5

Probas

5.1. Deseño experimental

5.1.1. Métrica para a detección de obxectos

Antes de definir a métrica, é necesario ter en conta certas características do conxunto de datos empregado (construído en base a PACO-LVIS [1], tal e como se describe na sección 3.2). En concreto, é especialmente importante coñecer que cada categoría conta con:

- Un conxunto de **imaxes negativas** nas que se garantiza que non existe ningunha instancia da clase correspondente.
- Un conxunto de **imaxes positivas exhaustivamente anotadas**, nas que se asegura que todos os obxectos da clase están rexistrados.
- Un conxunto de **imaxes positivas non exhaustivamente anotadas**, que conteñen, polo menos unha instancia da clase, pero nas que non se garantiza que todos os obxectos desa clase estean rexistrados.

Outro concepto necesario para poder avaliar a calidade das deteccióńns é a **IoU (*Intersection over Union*)**, unha medida que cuantifica o solapamento entre os cadros delimitadores preditos e o cadros reais anotados. Calcúlase coma o cociente da intersección entre ambos e a súa unión, tal e como se amosa na figura 5.1, de xeito que un valor de 1 representaría unha coincidencia exacta.

Unha vez revisada a IoU, pódese explicar un dos aspectos máis importantes á hora de definir unha métrica de avaliación de deteccióńns, a determinación dos Verdadeiros Positivos (VP), Falsos Positivos (FP) e Falsos Negativos (FN). Dadas as características do conxunto de datos mencionadas, determinámolos do seguinte xeito:

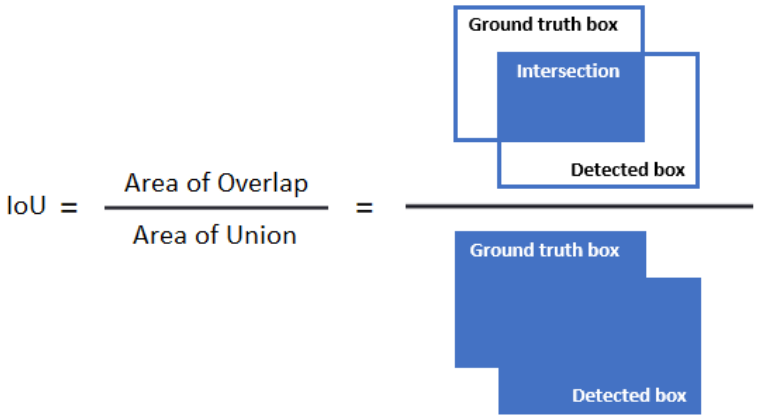


Figura 5.1: Intersección sobre Unión. Imaxe extraída de [12].

- **Imaxes negativas**: as deteccióńs de obxectos dunha categoría preditas nunha imaxe anotada negativamente para dita clase, considéranse Falsos Positivos.
- **Imaxes positivas exhaustivamente anotadas**: as deteccióńs de obxectos dunha categoría preditas nunha imaxe positiva anotada exhaustivamente para dita clase, considéranse Verdadeiros Positivos en caso de solaparse (por riba dun umbral de IoU determinado) con algunha das anotacións; en caso contrario, trátanse coma Falsos Positivos.
- **Imaxes positivas non exhaustivamente anotadas**: as deteccióńs de obxectos dunha categoría preditas nunha imaxe positiva non anotada exhaustivamente para dita clase, considéranse Verdadeiros Positivos en caso de solaparse (por riba dun umbral de IoU determinado) con algunha das anotacións; en caso contrario, non se contabilizan.

Por outra banda, considéranse Falsos Negativos aqueles obxectos anotados que non foron detectados. Cos termos descritos ata o momento, pódese definir, para cada clase, a precisión, que determina a proporción de deteccióńs correctas e o *recall*, que determina a proporción de obxectos detectados:

$$Precision = \frac{VP}{VP + FP} \tag{5.1}$$

$$Recall = \frac{VP}{VP + FN} \tag{5.2}$$

Desta maneira, empregando precisión e *recall*, xunto coas puntuacións de confianza de cada detección, calcúlase o **AP (*Average Precision*)** de cada categoría, que se define como a área baixo a curva precisión-*recall*. Habitualmente, esta integral aproxímase mediante interpolación, que se realiza discretizando a

curva nun número determinado de valores de *recall*. Especificamente, aproxímase a precisión (p) para un valor de *recall* (r) como a máxima precisión para calquera $r' > r$, tal e como se formaliza na ecuación 5.3. Deste xeito, a diferenza da curva orixinal, a interpolación é sempre decrecente.

$$p_{interp}(r) = \max_{r'>r} p(r') \tag{5.3}$$

Así, suavízanse as oscilacións na precisión que se producen a medida que aumenta o *recall*, de xeito que o AP é menos sensible ás pequenas variacións. A figura 5.2 mostra un exemplo sinxelo no que se amosa: en amarelo, a curva orixinal; en verde, a interpolación realizada sobre 11 puntos de *recall* e, en vermello, os 11 valores de precisión sobre os que se calcula a media para obter o AP.

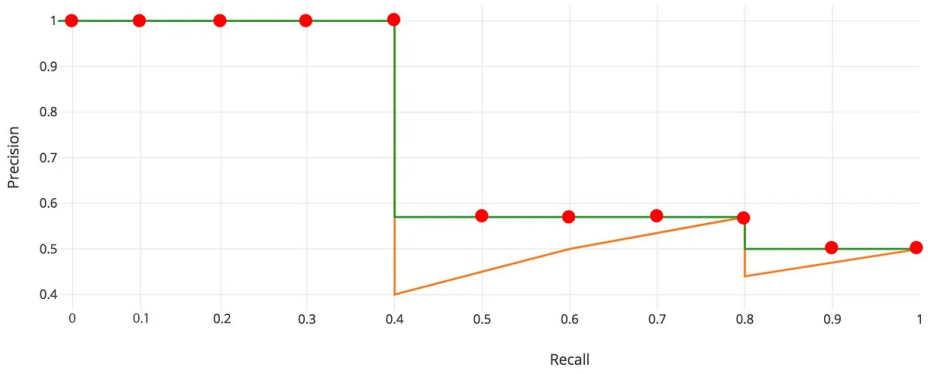


Figura 5.2: Cálculo do AP mediante interpolación de 11 puntos da curva precision-*recall*. Imaxe extraída de [13].

Cabe resaltar que os resultados obtidos neste traballo calcúlanse seguindo o estándar definido en COCO, que realiza a interpolación tomando 101 puntos de *recall* (de 0 a 1 con paso 0,01) e computa os seguintes valores para cada clase:

- **AP**: métrica principal que se calcula como a media do AP para os umbrais de IoU entre 0,5 e 0,95 con paso de 0,05.
- **AP50**: AP con umbral de IoU de 0,5.
- **AP75**: AP con umbral de IoU de 0,75.
- **APs**: AP (0,5:0,95) para obxectos pequenos: área $\leq 32^2$.
- **APm**: AP (0,5:0,95) para obxectos medianos: $32^2 <$ área $\leq 96^2$.
- **APl**: AP (0,5:0,95) para obxectos grandes: área $> 96^2$.

Por último, destacar que, en xeral, repórtanse os valores medios de todas as clases para cada unha das métricas, o que se coñece coma mAP. Non obstante, para simplificar, mantense a nomenclatura anterior.

5.1.2. Métrica para a predición de atributos

Para medir a calidade da predición de atributos, avalíase cada combinación de categoría e atributo predita, o que representa unha valoración realmente esixente.

Unha particularidade do conxunto de datos empregado é que non ten anotados todos atributos de todos os obxectos, polo que a métrica debe contemplar este fenómeno. As anotacións realizadas para cada obxecto consisten nunha lista de atributos e máis 4 indicadores que sinalan se os atributos de cada un dos tipos (cor, patrón, material e reflectancia) foron anotados. Polo tanto, considéranse anotacións positivas todos aqueles atributos contidos na lista e, anotacións negativas, aqueles atributos que non se inclúen na lista sempre e cando o indicador correspondente ao seu tipo marque que todos atributos desa variedade están anotados.

Séguese utilizando o AP coma métrica de avaliación e, neste caso, dada unha predición (c, a), onde c representa a categoría predita e a indica un dos atributos preditos, os Verdadeiros Positivos e Falsos Positivos xestiónanse do seguinte modo:

- **Imaxes negativas**: todas as predicións en imaxes negativas para a clase c considéranse Falsos Positivos.
- **Imaxes positivas exhaustivamente anotadas**: as detección cuxo cadro delimitador se solapa (por riba dun umbral de IoU) co dalgún obxecto real anotado co atributo a, considéranse Verdadeiros Positivos, en caso de que o obxecto estea anotado negativamente para dito atributo trátanse como Falsos Positivos e se o obxecto non ten atributos anotados, non se contabiliza a detección. En cambio, aquelas detección que non se solapan con ningún dos obxectos anotados, considéranse Falsos Positivos.
- **Imaxes positivas non exhaustivamente anotadas**: o tratamento das detección realizadas sobre imaxes deste tipo é igual ao mencionado para as imaxes exhaustivamente anotadas para a clase c, agás no caso daquelas detección cuxo cadro delimitador non se solapa con ningún dos obxectos anotados, xa que neste caso non son contabilizadas.

Da mesma forma que na experimentación de PACO, para garantir uns resultados representativos, soamente se computa o AP para unha combinación (c, a) se o conxunto de test contén polo menos unha instancia da clase c anotada positivamente co atributo a e outras 40 instancias desa mesma clase anotadas negativamente para dito atributo.

Por outra banda, existen algunhas combinacións de categoría e atributo moi pouco frecuentes, as cales poden ter unha alta influencia nos resultados. Para reducir esta varianza, agréganse os resultados a nivel de atributo, computando a media do AP entre todas as categorías avaliadas sobre o mesmo.

Para representar os resultados dos experimentos levados a cabo, repórtase o AP medio entre todos os atributos (AP_{att}), así como os valores medios entre os atributos de cada tipo: cor (AP_{att}^{cor}), patrón (AP_{att}^{pat}), material (AP_{att}^{mat}) e reflectancia (AP_{att}^{ref}).

5.1.3. Configuración das probas

Antes de describir as probas realizadas e os resultados da experimentación, é importante coñecer as condicións nas que se realizan e os parámetros establecidos para as execucións do detector proposto, tanto de adestramento, como de test.

En canto á configuración do adestramento, é necesario sinalar que o *backbone* está preadestrado sobre o conxunto de datos ImageNet [34] e lembrar que, tal e como se explicou na sección 3.5, o algoritmo de optimización utilizado é SGD (*Stochastic Gradient Descent*), cun tamaño de *mini-batch* de 16 imaxes.

Outros hiperparámetros de interese son os que definen o comportamento dos compoñentes de DeFRCN. No caso das GDLs, cada unha ten asociado un coeficiente de desacoplamento, de forma que se poden escalar os gradientes procedentes de RPN (mediante λ_1) e da cabeceira (mediante λ_2) de forma independente durante a retropropagación do erro. λ_1 se establece a 0 e λ_2 toma o valor 0,75 durante o adestramento base e 0,01 durante o axuste fino. No caso do PCB, α é o hiperparámetro que determina a magnitude do refinamento da confianza das detección, o cal se establece a 0,5.

En canto á taxa de aprendizaxe (γ), da que depende a velocidade da aprendizaxe, toma o valor 0,02 ao longo do adestramento base e redúcese á metade, 0,01 no axuste fino.

Un dato imprescindible para poder replicar calquera dos experimentos é o número de iteracións de adestramento, que está estreitamente relacionado coas baixadas da taxa de aprendizaxe (*Learning Rate*, RT) que se realizan ao alcanzar un número determinado de iteracións. Na aprendizaxe automática, unha maior cantidade de iteracións non ten por que dar lugar a un mellor modelo, de feito, o natural é que a aprendizaxe se estanque, ou incluso se cometa un sobreaxuste que empeore o rendemento, unha vez superados certos límites. Neste traballo, axustouse experimentalmente o número de iteracións máis apropiado para o adestramento base, que foi complementado con dúas baixadas da taxa de aprendizaxe, as cales provocan unha optimización máis sutil, co fin de retardar o estancamento. Obtivéronse 90 000 iteracións e baixadas de LR ás 70 000 e ás 80 000. Para o axuste fino, replicouse a configuración de DeFRCN para COCO:

- 1-shot: 1000 iteracións con baixada de LR ás 800.
- 2-shot: 1500 iteracións con baixada de LR ás 1200.
- 3-shot: 2000 iteracións con baixada de LR ás 1600.

- 5-shot: 2500 iteracións con baixada de LR ás 2000.
- 10-shot: 2500 iteracións con baixada de LR ás 2000.
- 30-shot: 4000 iteracións con baixada de LR ás 3200.

Outro aspecto crucial é coñecer como se realiza o axuste fino sobre D_{novel}. Antes de comezar este proceso, co obxectivo de aprender sobre clases nunca antes vistas usando poucos exemplos, reinícianse os pesos das capas de saída correspondentes á detección (clasificación e localización), de forma que se adestran unicamente en base ao soporte *few-shot*. No caso das capas de saída de predición de atributos, decidiuse (e probouse) que a mellor opción é manter os pesos obtidos durante o adestramento base e conxelar a optimización desta parte da cabeceira durante o axuste fino. As alternativas serían manter os pesos e realizar a optimización, co que se incurriría na sobreaprendizaxe; ou ben reiniciar os pesos e tratar de adestrar as capas durante o axuste fino. Esta última opción daría lugar a un nesgo moi alto, debido á incapacidade para extraer coñecemento sobre 55 posibles atributos a partir de datos con información insuficiente sobre os mesmos, xa que os atributos non están necesariamente balanceados nos *shots* de D_{novel}.

Por último, cabe destacar o protocolo seguido para a avaliación FSOD, que valora o rendemento do detector sobre as clases novas. Para garantir a consistencia dos resultados, imítase o que se fai en [26] para COCO. É dicir, execútanse 10 repeticións de adestramento (axuste fino) e test sobre D_{novel}, empregando en todas elas unha mesma partición (selección aleatoria de instancias para cada *shot* empregando unha mesma semente). Finalmente, repórtanse a media e a desviación típica de todos os resultados producidos.

5.2. Experimento 1: Detección *few-shot* sen atributos

A finalidade deste primeiro experimento é analizar o rendemento do detector proposto sobre o conxunto de datos construído nun contexto *few-shot*, no que, polo momento, se omite a intervención dos atributos. Non se realiza o adestramento da sección da cabeceira do detector responsable da predición dos atributos e, por suposto, tampouco se utiliza nas fases de inferencia e avaliación.

Dado que o obxectivo xeral da investigación levada a cabo é o estudo dos efectos da inclusión de atributos nun detector de obxectos *few-shot*, é imprescindible coñecer, en primeiro lugar, a calidade das deteccióǹs do modelo sen facer uso desa información adicional. Polo tanto, esta experimentación representa a liña base cuxos resultados serán tomados como referencia para poder determinar as consecuencias da introdución dos atributos.

Os resultados de todas as probas divídense en base e FSOD. Os resultados base correspóndense coa avaliación do detector tralo adestramento realizado sobre

D_{base}, na que se valoran soamente as categorías base (C_{base}). Por outra banda, os resultados FSOD correspóndense coa avaliación do detector tralo axuste fino realizado sobre D_{novel}, valorando unicamente as clases novas (C_{novel}). Dado que este traballo está especialmente enfocado na extracción de coñecemento sobre un conxunto reducido de imaxes etiquetadas, préstaselle máis atención aos resultados FSOD.

Para comezar, nas táboas 5.1 e 5.2, móstranse os resultados (base e FSOD) da avaliación do detector, calculando a media entre as categorías C_{base} e C_{novel} respectivamente. Repórtanse o AP e as súas variantes indicadas na sección 5.1 e os nomes das métricas se preceden de 'b' para base e 'n' para novel.

bAP	**bAP50**	**bAP75**	**bAPs**	**bAPm**	**bAPl**
22,30	35,79	23,23	12,06	29,00	39,90

Táboa 5.1: Resultados base da detección (sen atributos) computando a media das categorías en C_{base}.

En canto ás métricas xerais (AP, AP50 e AP75), a menos esixente é AP50 (que contabiliza todas as detecciónss que se solapan por riba dun 50 % cun obxecto real), polo que é razoable que teña a puntuación máis alta; non obstante, a métrica que se toma como referencia principal é o AP (0,5:0,95), por ser máis robusta. Respecto ás métricas que agrupan obxectos de distintos tamaños, o comportamento tamén é predicible, xa que, naturalmente, os obxectos máis grandes son máis fáciles de detectar.

		nAP	**nAP50**	**nAP75**	**nAPs**	**nAPm**	**nAPl**
1-shot	$\bar{x}$	1,50	3,21	1,18	0,80	1,30	2,50
	σ	0,06	0,09	0,13	0,04	0,12	0,11
2-shot	$\bar{x}$	3,18	6,32	2,98	1,31	3,82	7,35
	σ	0,09	0,14	0,17	0,04	0,17	0,57
3-shot	$\bar{x}$	4,69	9,28	4,60	2,10	6,08	8,76
	σ	0,06	0,15	0,12	0,10	0,12	0,36
5-shot	$\bar{x}$	6,69	13,24	6,34	3,24	9,15	11,98
	σ	0,05	0,13	0,10	0,07	0,07	0,19
10-shot	$\bar{x}$	9,81	19,19	9,20	4,68	15,15	16,86
	σ	0,07	0,17	0,08	0,08	0,19	0,23
30-shot	$\bar{x}$	11,80	23,49	10,79	6,07	17,52	18,51
	σ	0,05	0,12	0,15	0,06	0,13	0,17

Táboa 5.2: Resultados FSOD da detección (sen atributos) computando a media das categorías en C_{novel}. Repórtanse a media e a desviación típica de 10 execucións do axuste fino para cada *shot*.

No contexto *few-shot*, loxicamente, **os modelos axustados con *shots* de**

maior tamaño teñen unha calidade superior, o cal se pode observar atendendo a calquera das métricas de detección.

A capacidade de aprendizaxe do modelo neste escenario de escasa información pode evidenciarse, por exemplo, atendendo aos resultados do axuste fino con 30 instancias por categoría. No *30-shot*, tomando en consideración o AP das categorías novas (nAP), este alcanza un valor de máis do 50 % do AP das categorías base (bAP) e, reparando no AP50, o resultado acada entorno aos dous tercios do conseguido no adestramento base. A pesar de que os resultados base e FSOD non son estrictamente comparables debido a que o test non se realiza sobre as mesmas imaxes, tendo en conta a gran disparidade no tamaño entre D_{base} e D_{novel}, pode considerarse un éxito. Cabe destacar que, **neste caso, a diferenza entre a calidade das detecciónsde obxectos medianos e grandes redúcese considerablemente**, sobre todo, nos *shots* máis numerosos.

Respecto á desviación típica entre as distintas repeticións da avaliación FSOD, non se detecta que o tamaño do *shot* sea un condicionante. Non obstante, pódese observar claramente que, como cabe esperar, a menor variabilidade en termos absolutos, prodúcese ao empregar a métrica máis esixente, o AP. En calquera caso, os valores para o resto de métricas non son demasiado grandes, o cal é un bo sinal.

A continuación, é de utilidade obter os resultados da avaliación para cada categoría, xa que, máis adiante, serán necesarios para estudar os efectos da incorporación dos atributos sobre cada clase. Neste caso, repórtase unicamente a métrica principal (AP). Esta información, recopílase nas táboas 5.3 e 5.4.

cat	bAP	cat	bAP	cat	bAP	cat	bAP	cat	bAP
trash_can	37,91	bucket	32,52	guitar	5,86	pen	6,16	slipper	2,82
handbag	18,92	calculator	33,70	hammer	0,00	pencil	8,16	sponge	15,33
basket	26,86	can	16,72	hat	17,41	pillow	34,17	stool	30,78
belt	23,69	carton	12,40	helmet	34,12	pipe	7,26	tape	0,81
bench	21,37	cellular_phone	36,12	jar	27,14	plate	39,21	telephone	21,18
blender	44,31	chair	11,10	kettle	31,19	pliers	0,00	television_set	63,54
book	13,66	crate	11,47	lamp	26,51	remote_control	30,58	tissue_paper	13,00
bottle	29,36	cup	26,75	laptop	59,66	plastic_bag	18,52	tray	17,44
bowl	42,53	drill	0,00	mug	34,72	scissors	31,08	vase	38,92
box	15,26	drum	6,34	napkin	19,71	screwdriver	3,32	wallet	18,49
broom	19,50	glass	39,75	pan	12,04	shoe	21,12	wrench	15,74

Táboa 5.3: Resultados base da detección (sen atributos) para cada categoría pertencente a C_{base}.

Como se pode observar na táboa 5.3, existen clases que o detector non é capaz de recoñecer tralo adestramento base. Concretamente, o AP é 0 para *drill, hammer* e *pliers*, que son as tres categorías para as que se dispón dun menor número de instancias anotadas; de feito, en ningún dos casos se chega aos 50 exemplos de adestramento. Estas categorías non puideron ser seleccionadas como categorías novas debido a que incumprían a condición establecida de contar con, como mínimo, 100 instancias de test para obter resultados consistentes na

avaliación FSOD. Por este mesmo motivo, os datos sobre estas clases non son os máis representativos.

En xeral, pódese apreciar unha alta variabilidade entre os resultados de cada clase, un fenómeno no que probablemente inflúe, entre outras cousas, o feito de que os datos de adestramento non están balanceados. Á hora de configurar o conxunto de datos, decidiuse non equilibrar o número de instancias de todas as clases porque implicaría unha perda de información inadmisible. Ademais, **non se pode determinar que exista unha correlación total entre o número de instancias de adestramento e o AP**; de feito, tomando como referencia as categorías *book* (clase da que se dispón dun maior número de instancias anotadas, con 32249) e *calculator* (categoría que conta con soamente con 58 instancias de adestramento), o AP obtido para a primeira e claramente inferior, de 13, 66, fronte a 33, 70 para a segunda.

1-shot						2-shot					
cat	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**
ball	5,86	0,31	mirror	0,52	0,06	ball	12,95	0,69	mirror	0,84	0,09
bicycle	2,23	0,16	mouse	0,58	0,12	bicycle	2,68	0,21	mouse	15,1	0,52
car	0,21	0,08	newspaper	0,79	0,18	car	0,41	0,1	newspaper	1	0,18
clock	3,21	0,23	scarf	0,17	0,02	clock	3,81	0,31	scarf	0,25	0,02
dog	1,54	0,17	soap	0	0	dog	5,25	0,26	soap	0,52	0,12
earphone	0	0	spoon	0,61	0,19	earphone	0,17	0,14	spoon	1,44	0,09
fan	1,36	0,18	sweater	0,12	0,04	fan	1,09	0,16	sweater	1,32	0,23
knife	1,35	0,1	table	4,29	0,12	knife	1,69	0,09	table	4,25	0,09
ladder	0,06	0,07	towel	1,07	0,14	ladder	0,9	0,04	towel	0,71	0,24
microwave	5,96	0,39	watch	0	0	microwave	9,18	0,87	watch	0	0

3-shot						5-shot					
cat	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**
ball	13,99	0,57	mirror	1,56	0,09	ball	15,7	0,46	mirror	2,62	0,14
bicycle	3,43	0,11	mouse	16,98	0,63	bicycle	6,38	0,17	mouse	28,21	0,49
car	0,45	0,04	newspaper	1	0,11	car	0,45	0,05	newspaper	1,68	0,17
clock	4,26	0,21	scarf	0,6	0,09	clock	14,58	0,3	scarf	0,7	0,07
dog	6,85	0,27	soap	0,5	0,11	dog	7,6	0,35	soap	0,77	0,25
earphone	0,21	0,14	spoon	1,57	0,15	earphone	0,61	0,09	spoon	1,99	0,12
fan	5,15	0,44	sweater	1,06	0,15	fan	3,55	0,28	sweater	1,85	0,2
knife	1,8	0,12	table	3,19	0,13	knife	2,84	0,09	table	4,6	0,12
ladder	0,56	0,21	towel	0,94	0,16	ladder	1,4	0,2	towel	2,34	0,18
microwave	29,71	0,7	watch	0	0	microwave	35,87	0,66	watch	0	0

10-shot						30-shot					
cat	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**
ball	23,56	0,42	mirror	3,91	0,18	ball	24,35	0,35	mirror	3,21	0,13
bicycle	8,06	0,14	mouse	33,64	0,29	bicycle	7,86	0,17	mouse	36,62	0,27
car	4,73	0,27	newspaper	4,2	0,25	car	9,41	0,15	newspaper	6,67	0,23
clock	13,65	0,25	scarf	2,41	0,19	clock	19,05	0,16	scarf	3,05	0,1
dog	12,34	0,38	soap	5,47	0,42	dog	17,3	0,31	soap	7,46	0,26
earphone	0,22	0,05	spoon	3,33	0,15	earphone	0,88	0,18	spoon	5	0,18
fan	12,59	0,43	sweater	6,38	0,25	fan	12,18	0,3	sweater	10,09	0,39
knife	4,82	0,16	table	7,45	0,12	knife	6,53	0,22	table	9,39	0,16
ladder	0,82	0,16	towel	5,15	0,17	ladder	2,58	0,11	towel	6,52	0,18
microwave	42,91	0,49	watch	0,5	0,21	microwave	46,68	0,25	watch	1,09	0,06

Táboa 5.4: Resultados FSOD da detección (sen atributos) para cada categoría pertencente a C_{novel}. Repórtanse a media e a desviación típica de 10 execucións do axuste fino para cada shot.

Respecto ao axuste fino, cabe destacar que non existe variabilidade entre o

número de instancias utilizadas no adestramento de cada clase, aínda que sí que varía a cantidade de exemplos de test. Atendendo aos resultados dispoñibles na táboa 5.4, e seguindo a análise realizada anteriormente, pódese ver como, agás excepcións, o detector recoñece mellor os obxectos das categorías en C_{novel}, canto máis grande é o *shot*.

Algunhas das clases para as que existe unha maior diferenza entre os resultados dos *shots* máis reducidos e dos máis numerosos, son *car, mouse, soap* ou *sweater*. Todas elas teñen un $AP < 1$ no *shot* máis pequeno, e superan folgadamente os 5 puntos de AP en *30-shot*. Outro caso a destacar é o da categoría *microwave*, que no *shot* de tamaño 5, xa alcanza un AP de máis de 35 puntos, o cal supera á maioría das clases base, para as que se dispón dun número de exemplos de adestramento incomparablemente máis grande.

5.3. Experimento 2: Detección *few-shot* con atributos

Este segundo experimento realízase nunhas condicións exactamente idénticas ás do anterior, coa diferenza fundamental de que, neste caso, si que se explota a información dos atributos. Durante o adestramento do detector, optimízase a cabeceira ao completo (aínda que cabe lembrar que a parte correspondente aos atributos conxélase durante o axuste fino), de xeito que, en inferencia, é capaz de predicir os atributos de cada obxecto detectado.

Durante a fase de test do modelo, avalíanse tanto a tarefa de detección, como a predición dos atributos. En particular, a partir dos resultados de detección deste experimento, poderase verificar ou desmentir a hipótese inicial deste traballo, presentada na sección 1.2. Dita hipótese consiste na suposición de que complementar a descrición dos obxectos etiquetados con información sobre os seus atributos (ademais da categoría e posición), mellora a calidade da detección, en especial, nun contexto few-shot.

Antes de comprobar se, efectivamente, mellora a calidade da detección, convén analizar os resultados da predición dos atributos. Non se debe obviar este paso, xa que, no caso de que o detector non conseguise aprender a realizar esta tarefa con certa calidade, deduciríase que non se extrae coñecemento da información adicional de forma axeitada, polo que non tería tanto sentido estudar o posible cambio no rendemento da detección de obxectos.

De igual maneira que no anterior experimento, preséntanse, en tódolos casos, os resultados base e FSOD. Os resultados da predición dos atributos recóllense nas táboas 5.5 e 5.6, nas que se indican as métricas para atributos presentadas anteriormente: o AP_{att} xeral de todos os atributos, e os de cada tipo.

Para realizar a análise dos resultados da predición de atributos, débese ter en consideración que os valores para as métricas desta tarefa e os das métricas da

$\mathbf{bAP}_{att}$	$\mathbf{bAP}^{cor}_{att}$	$\mathbf{bAP}^{pat}_{att}$	$\mathbf{bAP}^{mat}_{att}$	$\mathbf{bAP}^{ref}_{att}$
12,37	10,43	12,95	9,77	16,31

Táboa 5.5: Resultados base da predición de atributos.

		$\mathbf{nAP}_{att}$	$\mathbf{nAP}^{cor}_{att}$	$\mathbf{nAP}^{pat}_{att}$	$\mathbf{nAP}^{mat}_{att}$	$\mathbf{nAP}^{ref}_{att}$
1-shot	$\bar{x}$	0,93	0,99	0,83	0,84	1,07
	σ	0,29	0,14	0,04	0,15	1,15
2-shot	$\bar{x}$	1,51	1,39	1,43	1,60	1,61
	σ	0,11	0,10	0,10	0,21	0,48
3-shot	$\bar{x}$	1,71	1,90	2,03	1,91	0,99
	σ	0,13	0,16	0,10	0,38	0,11
5-shot	$\bar{x}$	2,73	2,44	2,72	2,91	2,86
	σ	0,21	0,14	0,12	0,23	0,75
10-shot	$\bar{x}$	2,94	3,13	3,42	2,97	2,24
	σ	0,20	0,09	0,08	0,24	0,72
30-shot	$\bar{x}$	4,45	3,61	4,43	3,82	5,92
	σ	0,34	0,11	0,21	0,20	1,32

Táboa 5.6: Resultados FSOD da predición de atributos. Repórtanse a media e a desviación típica de 10 execucións do axuste fino para cada *shot*.

detección de obxectos non son comparables en ningún caso. Como xa se mencionou na sección 5.1, a métrica da predición dos atributos ten unha esixencia moi superior, xa que depende das detección realizadas e combina a clase do obxecto detectado cos atributos preditos.

Na táboa 5.5 pódese observar como os atributos asociados á reflectancia son os que o modelo predice con maior efectividade. Esta diferenza respecto aos demais tipos explícase, entre outras cousas, debido a que a reflectancia é o tipo que conta con menos valores posibles, sendo tan só tres (opaco, translúcido e transparente).

Na configuración das probas presentada na sección 5.1.3, explicouse que, durante o axuste fino, non se modifica a parte da cabeceira encargada da predición dos atributos; non obstante, é conveniente estudar a capacidade de xeneralización do detector á hora de predicir atributos sobre categorías nunca antes vistas.

Neste aspecto, atendendo á media da métrica global amosada na táboa 5.6, pódese observar como aumenta consistentemente en función do tamaño do *shot* utilizado (destacando o crecemento entre os *shots* 1 e 2, 3 e 5, e 10 e 30), de forma que se alcanza, no *30-shot* un AP_{att} que representa o 36 % do obtido sobre as clases base, non obstante, como se mencionou no primeiro experimento, nAP e bAP non son directamente comparables. Considerando que non se realiza ningunha aprendizaxe sobre os atributos anotados das clases novas, pódese considerar un gran logro. Prestando atención á desviación típica que presentan os resultados das diferentes execucións, si que se pode ver unha variabilidade considerable, en

todo caso, entra dentro dos valores razoables.

Tras comprobar que o modelo proposto aprendeu a predicir atributos dun xeito adecuado, é o momento de examinar a influencia dos atributos na detección de obxectos. Polo tanto, en primeiro lugar, na táboa 5.7 preséntanse os resultados base da avaliación da tarefa de detección (onde non se ten en conta o erro dos atributos, o cal si que tivo influencia durante o adestramento), calculando a media entre as categorías de C_{base}.

bAP	**bAP50**	**bAP75**	**bAPs**	**bAPm**	**bAPl**
21,88	35,21	23,02	11,78	28,61	39,91

Táboa 5.7: Resultados base da detección (con atributos) computando a media das categorías en C_{base}.

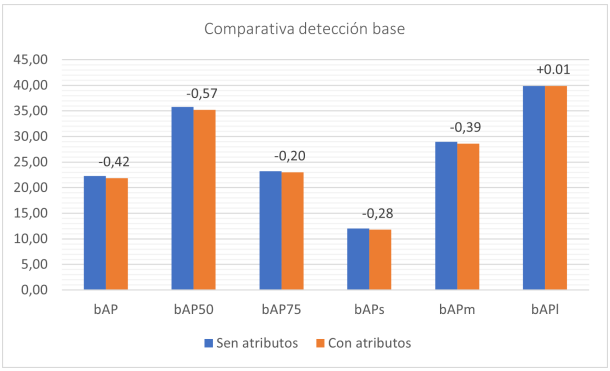


Figura 5.3: Comparativa entre os resultados da detección base dos modelos sen atributos e con atributos.

Atendendo á táboa 5.7, pódese observar que os resultados tralo adestramento base son moi similares aos obtidos mediante o modelo sen atributos. Para comparar dun xeito máis directo ambas alternativas, preséntase a figura 5.3. Nela, **non se pode apreciar que a integración dos atributos teña un impacto positivo no rendemento do detector sobre as clases base**. A pesar de que este resultado non parece demasiado prometedor, isto non quere dicir que no contexto de escasa información (*few-shot*) suceda o mesmo, polo que, a continuación, débese analizar o efecto dos atributos tralo axuste fino. Para elo, disponse da táboa 5.8.

A diferenza dos resultados base, tralo axuste fino, a calidade das deteccións sobre as clases novas é claramente mellor neste caso (facendo uso da información dos atributos), que no experimento anterior (sen explotar dita información). As puntuacións obtidas en cada unha das métricas xerais (AP, AP50 e AP75) son máis altas para todos os tamaños de *shot*, e tamén o son, agás excepcións, para as métricas que agrupan obxectos de distintos tamaños (APs, APm e APl). Non

		nAP	nAP50	nAP75	nAPs	nAPm	nAPl
1-shot	$\bar{x}$	1,98	4,07	1,62	0,90	2,07	4,60
	σ	0,05	0,12	0,08	0,03	0,08	0,52
2-shot	$\bar{x}$	3,39	6,58	3,22	1,60	4,10	8,20
	σ	0,07	0,13	0,15	0,06	0,11	0,15
3-shot	$\bar{x}$	4,87	9,50	4,66	2,39	6,26	10,27
	σ	0,05	0,09	0,12	0,09	0,09	0,09
5-shot	$\bar{x}$	7,10	13,57	6,92	3,48	9,45	13,68
	σ	0,06	0,10	0,10	0,08	0,18	0,16
10-shot	$\bar{x}$	10,02	19,39	9,44	4,92	14,66	17,35
	σ	0,09	0,15	0,13	0,08	0,17	0,24
30-shot	$\bar{x}$	12,34	24,18	11,56	6,72	17,88	18,91
	σ	0,08	0,19	0,14	0,06	0,15	0,14

Táboa 5.8: Resultados FSOD da detección (con atributos) computando a media das categorías en C_{novel}. Repórtanse a media e a desviación típica de 10 execucións do axuste fino para cada *shot*.

semella que esta melloría derive dunha maior capacidade para detectar obxectos dalgún tamaño concreto, polo que se pode centrar a análise na métrica principal, o AP.

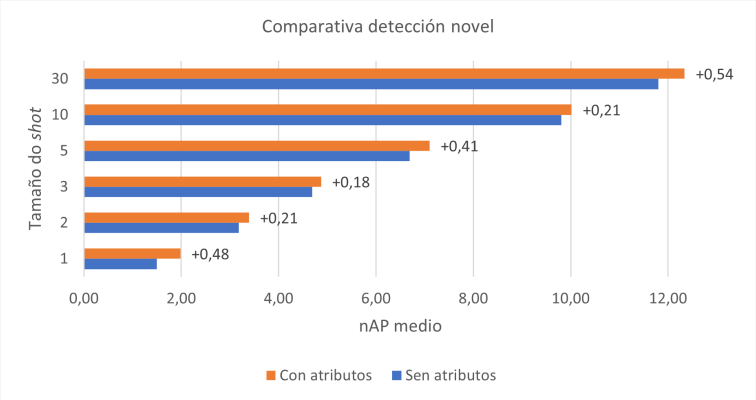


Figura 5.4: Comparativa entre os resultados da detección FSOD dos modelos sen atributos e con atributos.

Na figura 5.4 amósase unha gráfica na que se pode apreciar a diferenza entre ambos modelos respecto ao AP medio en cada un dos *shots*. Concretamente, a alternativa con atributos consegue uns crecementos nesta métrica de, aproximadamente: 0,5, 0,2, 0,2, 0,4, 0,2 e 0,5 puntos para os *shots* de tamaño: 1, 2, 3, 5, 10 e 30 respectivamente.

É imprescindible destacar que a baixa desviación típica entre os resultados das 10 execucións (dispoñible na táboa 5.8), demostra que a mellora é consis-

tente, xa que non se trata dunha única execución puntual na que se obteñen resultados lixeiramente superiores; senón que, **no escenario *few-shot*, o detector proposto ten un rendemento recorrentemente máis alto usando a información dos atributos, que sen utilizala**.

Unha vez repasados os resultados facendo a media de todas as clases, pode ser interesante analizar como afecta a incorporación dos atributos sobre cada categoría. Para comezar, na táboa 5.9 preséntanse os resultados de detección tralo adestramento base por categoría.

cat	bAP	cat	bAP	cat	bAP	cat	bAP	cat	bAP
trash_can	38,52	bucket	30,16	guitar	8,24	pen	6,49	slipper	0,00
handbag	18,77	calculator	32,03	hammer	0,00	pencil	7,10	sponge	18,04
basket	26,84	can	15,18	hat	17,74	pillow	33,70	stool	27,49
belt	23,01	carton	7,33	helmet	32,29	pipe	7,49	tape	2,19
bench	21,29	cellular_phone	35,40	jar	28,13	plate	39,97	telephone	22,41
blender	45,64	chair	11,23	kettle	26,68	pliers	0,00	television_set	63,48
book	13,57	crate	11,32	lamp	26,04	remote_control	30,29	tissue_paper	13,39
bottle	28,42	cup	28,26	laptop	60,27	plastic_bag	18,94	tray	17,98
bowl	41,90	drill	0,00	mug	37,03	scissors	30,22	vase	38,24
box	15,03	drum	7,92	napkin	19,82	screwdriver	2,57	wallet	18,75
broom	13,84	glass	39,75	pan	11,06	shoe	20,16	wrench	11,48

Táboa 5.9: Resultados base da detección (con atributos) para cada categoría pertencente a C_{base}.

Como xa se viu anteriormente, o uso dos atributos non ten unha repercusión perceptible na detección trala aprendizaxe sobre D_{base}, polo que os datos por clase non aportan demasiada información, máis alá de confirmar que a distribución do AP ao longo das clases é practicamente igual ao primeiro experimento. Para finalizar, na táboa 5.10 amósanse os resultados FSOD da detección para cada categoría nova.

A pesar de que en cada *shot* hai algunhas categorías novas para as que o rendemento do detector aumenta notablemente (respecto ao modelo sen atributos), non se poden extraer conclusións sobre aquelas clases nas que os atributos teñen unha maior influencia, xa que, en cada *shot*, as categorías cunha mellora máis relevante son diferentes. Ademais, tamén hai algunhas clases cuxo rendemento diminúe e, do mesmo xeito, tamén varían en cada *shot*. Para visualizar este fenómeno, a táboa 5.11 recolle o crecemento do nAP medio do detector con atributos respecto ao modelo do primeiro experimento.

Desta forma, pódese deducir que a melloría descuberta neste segundo experimento non se produce debido a que os atributos sexan moi útiles para recoñecer obxectos dun pequeno subconxunto de C_{novel}. Polo tanto, **dado que a influencia dos atributos non se reduce soamente a unhas poucas clases, pódese concluír que a información sobre atributos axuda a aumentar a calidade da detección *few-shot* dun xeito global**.

1-shot						2-shot					
cat	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**
ball	8,95	0,57	mirror	0,55	0,01	ball	14,68	0,58	mirror	1,13	0,16
bicycle	2,1	0,14	mouse	2,13	0,22	bicycle	1,86	0,11	mouse	14,76	0,7
car	0,17	0,04	newspaper	0,8	0,16	car	0,41	0,06	newspaper	0,66	0,14
clock	7,46	0,47	scarf	0,15	0,07	clock	8,23	0,62	scarf	0,35	0,03
dog	2,25	0,19	soap	0,09	0,14	dog	5,6	0,37	soap	0,6	0,04
earphone	0	0	spoon	0,35	0,14	earphone	0	0	spoon	1,61	0,3
fan	1,52	0,22	sweater	0,28	0,08	fan	1,92	0,22	sweater	0,47	0,09
knife	1,18	0,09	table	4,64	0,2	knife	1,45	0,08	table	4,4	0,16
ladder	0,04	0,07	towel	2,23	0,18	ladder	0,16	0,13	towel	2,27	0,21
microwave	4,69	0,35	watch	0	0	microwave	7,13	0,55	watch	0	0

3-shot						5-shot					
cat	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**
ball	15,39	0,55	mirror	1,68	0,13	ball	15,7	0,53	mirror	3,08	0,13
bicycle	3,07	0,17	mouse	16,26	0,82	bicycle	6,2	0,16	mouse	26,33	0,41
car	0,4	0,03	newspaper	0,76	0,12	car	0,47	0,04	newspaper	1,89	0,14
clock	7,31	0,43	scarf	0,41	0,05	clock	16,21	0,24	scarf	0,71	0,1
dog	6,96	0,28	soap	0,56	0,04	dog	8,62	0,31	soap	1,96	0,46
earphone	0,19	0,07	spoon	1,63	0,12	earphone	0,24	0,07	spoon	2,71	0,14
fan	3,77	0,34	sweater	1,2	0,18	fan	4,65	0,37	sweater	2,3	0,18
knife	2,15	0,1	table	3,71	0,13	knife	3,36	0,07	table	4,63	0,08
ladder	0,16	0,1	towel	1,94	0,2	ladder	0,3	0,06	towel	3,98	0,11
microwave	29,75	0,43	watch	0	0	microwave	38,6	0,4	watch	0	0

10-shot						30-shot					
cat	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**	**cat**	$\bar{x}$ **nAP**	σ **nAP**
ball	22,06	0,62	mirror	3,73	0,11	ball	25,13	0,46	mirror	4,46	0,29
bicycle	7,42	0,11	mouse	31,2	0,18	bicycle	7,38	0,21	mouse	36,38	0,24
car	5,38	0,26	newspaper	5,94	0,22	car	7,82	0,09	newspaper	8,38	0,32
clock	13,22	0,23	scarf	2,79	0,32	clock	19,71	0,17	scarf	2,97	0,15
dog	12,95	0,1	soap	10,27	0,49	dog	19,76	0,4	soap	11,64	0,47
earphone	0,1	0,07	spoon	3,97	0,16	earphone	1,07	0,15	spoon	4,87	0,28
fan	13,24	0,46	sweater	6,93	0,3	fan	12,37	0,5	sweater	10,28	0,3
knife	4,84	0,17	table	7,79	0,11	knife	6,25	0,15	table	10,25	0,15
ladder	0,26	0,05	towel	5,72	0,21	ladder	2,34	0,27	towel	6,99	0,17
microwave	42,35	0,59	watch	0,19	0,03	microwave	47,75	0,45	watch	1,06	0,07

Táboa 5.10: Resultados FSOD da detección (con atributos) para cada categoría pertencente a C_{novel}. Repórtanse a media e a desviación típica de 10 execucións do axuste fino para cada shot.

cat	**Δ nAP 1-shot**	**Δ nAP 2-shot**	**Δ nAP 3-shot**	**Δ nAP 5-shot**	**Δ nAP 10-shot**	**Δ nAP 30-shot**
ball	3,09	1,73	1,4	0	-1,5	0,78
bicycle	-0,13	-0,82	-0,36	-0,18	-0,64	-0,48
car	-0,04	0	-0,05	0,02	0,65	-1,59
clock	4,25	4,42	3,05	1,63	-0,43	0,66
dog	0,71	0,35	0,11	1,02	0,61	2,46
earphone	0	-0,17	-0,02	-0,37	-0,12	0,19
fan	0,16	0,83	-1,38	1,1	0,65	0,19
knife	-0,17	-0,24	0,35	0,52	0,02	-0,28
ladder	-0,02	-0,74	-0,4	-1,1	-0,56	-0,24
microwave	-1,27	-2,05	0,04	2,73	-0,56	1,07
mirror	0,03	0,29	0,12	0,46	-0,18	1,25
mouse	1,55	-0,34	-0,72	-1,88	-2,44	-0,24
newspaper	0,01	-0,34	-0,24	0,21	1,74	1,71
scarf	-0,02	0,1	-0,19	0,01	0,38	-0,08
soap	0,09	0,08	0,06	1,19	4,8	4,18
spoon	-0,26	0,17	0,06	0,72	0,64	-0,13
sweater	0,16	-0,85	0,14	0,45	0,55	0,19
table	0,35	0,15	0,52	0,03	0,34	0,86
towel	1,16	1,56	1	1,64	0,57	0,47
watch	0	0	0	0	-0,31	-0,03

Táboa 5.11: Crecemento por categoría do nAP medio do modelo con atributos respecto ao modelo sen atributos.

Capítulo 6

Discusión dos resultados

Tras revisar os resultados da experimentación, pódense extraer as seguintes conclusións:

1. A calidade das deteccióǹs no escenario *few-shot* crece consistentemente a medida que aumenta o tamaño do *shot*.

2. Os obxectos grandes son claramente os que mellor detecta o modelo tralo adestramento base, non obstante, tralo axuste fino sobre as clases novas, a diferenza entre a capacidade do detector para recoñecer obxectos medianos e grandes redúcese considerablemente.

3. A capacidade dun detector para recoñecer obxectos dunha categoría non depende exclusivamente do número de instancias de adestramento das que se dispoña. Isto aplica tanto ao adestramento base como ao axuste fino *few-shot*. No primeiro caso púidose comprobar que o AP dalgunhas categorías pouco comúns superaba o AP daquelas categorías máis frecuentes e, no escenario *few-shot*, onde o número de instancias de adestramento é o mesmo para todas as clases, os resultados para cada unha delas diferían en gran medida. Neste sentido, dado que o procesado da información de D_{base} e D_{novel} é distinto, non ten sentido comparar o AP das clases novas co das base para este caso particular.

4. A incorporación de información sobre atributos dos obxectos non supón un beneficio evidente no adestramento base. A calidade das deteccióǹs tralo adestramento sobre D_{base} non mellora a pesar de facer uso de información adicional.

5. Complementar a descrición dos obxectos con información sobre atributos repercute positivamente na capacidade do detector para recoñecer obxectos de categorías novas, tralo axuste fino sobre D_{novel}. O AP crece entre 0,2 e 0,5 puntos dependendo do tamaño do *shot*, grazas ao uso dos atributos.

6. A melloría no rendemento que se produce tralo axuste fino debido ao aproveitamento da información dos atributos, non se reduce a un pequeno subconxunto das clases novas sobre as que os atributos teñen un efecto positivo; senón que a melloría é global, xa que, en cada *shot*, as categorías que se ven beneficiadas e prexudicadas son distintas.

7. O detector proposto é capaz de xeneralizar o coñecemento sobre os atributos adquirido no adestramento base, para predicir os atributos de categorías nunca antes vistas. Neste aspecto, destaca o feito de que, durante o axuste fino, non se aprende nada de información sobre os atributos das clases en C_{novel}.

A partir destas conclusións, pódese afirmar que a hipótese de partida presentada na sección 1.2 queda parcialmente confirmada. En primeiro lugar, a conclusión 4 demostra que complementar a descrición dos obxectos con información sobre atributos, non mellora a calidade da detección nun escenario de información abundante. Pola contra, a conclusión 5 confirma a segunda parte da hipótese, demostrando que o uso de dita información adicional si que mellora o rendemento da detección de obxectos nun contexto *few-shot*.

A evidente diferenza da influencia dos atributos no adestramento base fronte ao axuste fino *few-shot* é comprensible dende o punto de vista de que, ante un escenario con moi poucos datos sobre cada categoría (D_{novel}), toda a información adicional que caracterice aos obxectos representa unha grande axuda para a optimización do modelo. Por outra banda, D_{base} conta cun conxunto moi numeroso de imaxes etiquetadas coa categoría e a posición dos obxectos, polo que os atributos quedan nun segundo plano á hora de aprender a tarefa de detección.

Capítulo 7

Conclusións e posibles ampliacións

Ao longo deste traballo estudouse en profundidade o problema da detección de obxectos. Dado que os detectores de obxectos tradicionais precisan dun conxunto moi numeroso de imaxes etiquetadas cuxa anotación manual é, en moitos casos, inviable, a investigación estivo centrada en tratar de mellorar o rendemento dos detectores *few-shot*, que empregan un conxunto reducido de exemplos de adestramento.

Unha posibilidade para enriquecer os datos moito menos custosa que aumentar o número de instancias anotadas, é complementar as descricións dos obxectos cunha lista de atributos, xa que se evita o paso de determinar cadros delimitadores precisos. Ademais, en certos contextos, non é factible anotar máis obxectos debido á non dispoñibilidade de imaxes adicionais. Precisamente, dado o baixo custo da anotación de atributos (especialmente, no contexto *few-shot*), analizouse o efecto da inclusión desta información engadida no rendemento da detección.

Para poder levar a cabo dita análise, fíxose unha partición *few-shot* dun conxunto de datos de detección con atributos (PACO-LVIS [1]) e implementouse un detector baseado en DeFRCN [2] (un modelo *few-shot* baseado en redes convolucionais) con capacidade para a predición de atributos.

Empregando os compoñentes desenvolvidos, púidose comprobar que os atributos non son de grande utilidade para mellorar o rendemento dun detector cando xa se dispón dunha abondosa cantidade de instancias anotadas. Non obstante, no escenario de información limitada sobre categorías nunca antes vistas, a simple introdución dos atributos nas escasas anotacións, supón unha melloría notable, sobre todo, tendo en conta o mínimo esforzo que conleva o enriquecemento dos datos neste contexto.

Aínda que as conclusións obtidas a partir da experimentación son realmente prometedoras, os resultados da detección sobre categorías novas atópanse todavía considerablemente lonxe do rendemento que se consegue ao contar cunha cantidade de instancias por clase un par de magnitudes superior.

Unha liña de traballo futuro interesante podería estar enfocada en intentar sacarlle algo máis de partido á información sobre os atributos dos obxectos. En relación con este concepto, durante o desenvolvemento deste traballo, realizouse un breve estudo preliminar co obxectivo de formular algunha posible ampliación do mesmo.

A idea desenvolvida fundamentábase nun posprocesado sinxelo das predicións realizadas. Concretamente, consistía en incrementar a confianza daquelas detec-cións para as que os atributos preditos tiveran unha alta correlación cos dalgún dos obxectos da mesma clase anotados no *shot*. A suposición realizada para suxerir esta proposta foi a seguinte: se se detecta un obxecto dunha categoría determinada e os seus atributos coinciden cos das instancias de adestramento, probablemente esa detección sexa correcta.

Non se chegou a profundizar demasiado nesta liña, polo que non se obtiveron resultados consistentes, pero pode chegar a ser unha boa opción para procurar o máximo aproveitamento da información dos atributos e, así, mellorar a calidade das detec-cións sen aumentar o tamaño do conxunto de datos.

Apéndice A

Manual técnico

Neste apéndice descríbense todos os requisitos necesarios para configurar unha contorna de execución en GNU/Linux a través da que sexa posible repetir a experimentación levada a cabo.

A totalidade do código fonte empregado ao longo deste traballo pode atoparse nun repositorio público de GitHub, accesible a través de `https://github.com/hectormc01/Repo_TFG`.

Ademais do código, é imprescindible obter os conxuntos de datos protagonistas deste estudo, os cales, debido ao seu tamaño, non se incorporan no propio repositorio. Para preparar os conxuntos de datos, executaremos algúns scripts de Python [53], polo que se recomenda a versión 3.8.10, que foi a empregada durante a experimentación.

En primeiro lugar, indícanse os pasos a seguir para conseguir o conxunto de datos de detección *few-shot* con atributos empregado durante as probas:

1. Descargar as imaxes de COCO [5] train2017 e val2017. Para elo, situarse no directorio `/datasets` do proxecto e crear un novo directorio chamado `coco`, dentro do mesmo, executar:

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

2. Descargar as anotacións de PACO-LVIS [1]. Dentro do proxecto, crear `/datasets/paco/annotations`. Dentro do directorio máis profundo executar:

```
wget https://dl.fbaipublicfiles.com/paco/
   annotations/paco_lvis_v1.zip
unzip paco_lvis_v1.zip
```

3. Eliminar as partes de obxectos das anotacións de PACO-LVIS. Situarse na raíz do proxecto e executar o script creado para dita tarefa como segue:

```
python3 remove_paco_part_categories.py datasets
   /paco/annotations/paco_lvis_v1_train.json
python3 remove_paco_part_categories.py datasets
   /paco/annotations/paco_lvis_v1_test.json
```

4. Crear a partición *few-shot* a partir das anotacións de adestramento de PACO-LVIS. Dende a raíz do proxecto, executar os scripts correspondentes da seguinte maneira:

```
python3 min_30_instances.py datasets/paco/
   annotations/paco_lvis_v1_train.json
python3 pacosplit.py
```

Ademais do conxunto de datos anterior, débese descargar ImageNet [34], necesario para o preadestramento do *backbone* do detector e para o módulo PCB. Para gardar este conxunto de datos, situarse no directorio `/datasets` do proxecto e executar:

```
wget --load-cookies /tmp/cookies.txt "https://docs.
   google.com/uc?export=download&confirm=$(wget --
   quiet --save-cookies /tmp/cookies.txt --keep-
   session-cookies --no-check-certificate 'https://
   docs.google.com/uc?export=download&id=1
   rsE20_fSkYeIhFaNU04rBfEDkMENLibj' -O- | sed -rn
   's/.*confirm=([0-9A-Za-z_]+).*/\1\n/p')&id=1
   rsE20_fSkYeIhFaNU04rBfEDkMENLibj" -O
   ImageNetPretrained.zip && rm -rf /tmp/cookies.
   txt
unzip ImageNetPretrained.zip
```

Unha vez obtido o código e máis os conxuntos de datos, é importante asegurarse de contar coas dependencias software necesarias. Tal e como se indica na sección 4.2, requírese dos seguintes paquetes de Python, nas versións especificadas:

- NumPy 1.23.3 [54]
- PyTorch 1.10.0 [55]
- Detectron2 0.3 [4]
- OpenCV 4.7.0 [56]
- Sci-kit Learn 1.1.2 [57]

- LVIS 0.5.3 [32]

No proxecto inclúese un Dockerfile que se pode utilizar para crear unha imaxe Docker [50] coa que instanciar contedores preparados para a execución deste detector. No manual de usuario (apéndice B), detállase o procedemento para xestionar este tipo de contedores.

Por último, débese ter en conta que o adestramento dun modelo como o proposto neste traballo require dunha capacidade de computación certamente elevada. En particular, para replicar os mesmos experimentos nas condicións exactas, requírese dunha GPU (ou varias) de forma que se dispoña, en total, dun mínimo de 32 GB de RAM. Para sacarlle o máximo partido ás GPUs precísase de CUDA [51], concretamente, ao longo desta investigación utilizouse a versión 11.3

Apéndice B

Manual de usuario

Neste segundo apéndice descríbese o procedemento a seguir para replicar calquera dos experimentos levados a cabo durante a investigación. Presuponse que se dispón do código fonte e dos conxuntos de datos necesarios, e que se cumpren os requisitos de software e hardware básicos presentados no manual técnico (apéndice A).

A alternativa máis sinxela para xestionar as dependencias do proxecto é empregar un contedor de software, en particular, un contedor Docker [50]. A continuación, preséntase un Dockerfile que serve para xerar unha imaxe, a partir da cal crear contedores con todas as dependencias solucionadas.

```
FROM nvidia/cuda:11.3.0-cudnn8-devel

ENV DEBIAN_FRONTEND noninteractive
RUN apt-key adv --keyserver keyserver.ubuntu.com --recv
   -keys A4B469963BF863CC
RUN apt-get update && apt-get install -y \
  python3-opencv ca-certificates python3-dev git wget
     sudo python3-pip && \
  rm -rf /var/lib/apt/lists/*

# install dependencies
# See https://pytorch.org/ for other options if you use
    a different version of CUDA
# RUN pip3 install torch torchvision tensorboard cython
RUN pip3 install cython
RUN pip3 install torch==1.10.0+cu113 torchvision
   ==0.11.1+cu113 -f https://download.pytorch.org/whl/
   cu113/torch_stable.html
RUN pip3 install 'git+https://github.com/cocodataset/
   cocoapi.git#subdirectory=PythonAPI'
```

```
RUN pip3 install 'git+https://github.com/
   facebookresearch/fvcore'
# install detectron2
RUN git clone --branch v0.3 https://github.com/
   facebookresearch/detectron2 detectron2_repo
ENV FORCE_CUDA="1"
# This will build detectron2 for all common cuda
   architectures and take a lot more time,
# because inside `docker build`, there is no way to
   tell which architecture will be used.
ENV TORCH_CUDA_ARCH_LIST="Kepler;Kepler+Tesla;Maxwell;
   Maxwell+Tegra;Pascal;Volta;Turing;Ampere"
RUN pip3 install -e detectron2_repo

RUN pip3 install scipy pandas scikit-image sklearn

# Set a fixed model cache directory.
ENV FVCORE_CACHE="/tmp"
ENV PYTHONPATH="/workspace/detectron"

# Install LVIS
RUN pip install lvis
```

Para crear a imaxe Docker a partir do Dockerfile anterior, executar o seguinte comando:

```
docker build <directorio_dockerfile> -t detectron2
   :03_cu113_pytorch110_lvis
```

Coa opción `-t` indícanse o nome e o etiqueta, que conforman o identificador (nome:etiqueta) da imaxe. Para instanciar contedores, indícase o identificador da imaxe utilizada.

Unha vez a imaxe está dispoñible, pódese instanciar un contedor executando o comando a continuación:

```
docker run -d --gpus all --ipc=host --privileged -
   it --name <nome_contedor> -e "
   CUDA_VISIBLE_DEVICES=0" -e "CUDA_DEVICE_ORDER=
   PCI_BUS_ID" -v <directorio_proxecto>:/workspace
   -v <directorio_saida>:/results -v <
   directorio_imagenet>:/ImageNetPretrained
   detectron2:03_cu113_pytorch110_lvis /bin/bash
```

Algúns detalles de interese:

- `-d`: enviar a segundo plano.
- `--gpus all`: soporte para GPU.
- `--it`: crear un terminal interactivo.
- `-e "CUDA_VISIBLE_DEVICES=0"`: indica a GPU que se vai a utilizar na execución. Pódense indicar varias separadas por comas.
- `-v <directorio_proxecto>:/workspace`: monta o directorio do proxecto (seguindo o manual técnico, sería `Repo\_TFG`) no directorio `/workspace` do contedor.
- `-v <directorio_saida>:/results`: monta no contedor o directorio onde se gardarán tanto os resultados, como os checkpoints realizados ao longo do adestramento.
- `-v <directorio_imagenet>:/ImageNetPretrained`: monta no contedor o directorio onde se almacena o conxunto de datos ImageNet [34] (seguindo o manual técnico, sería `Repo_TFG/datasets/ImageNetPretrained`).
- `/bin/bash`: indícase o shell do contedor.

Unha vez creado o contedor, acceder co comando:

```
docker attach <nome_contedor>
```

Neste punto, xa está todo preparado para comezar coas execucións.

En primeiro lugar, é importante lembrar que na experimentación se probaron dous detectores, un deles adaptado á predición atributos e o outro sen esta capacidade. O proxecto está configurado, por defecto, para executar o modelo con soporte para os atributos, en calquera caso, os cambios a realizar para utilizar un modelo ou outro son os seguintes:

1. No ficheiro de configuración `/workspace/configs/Base-RCNN.yaml`, indicar como cabeceira do detector (`ROI_HEADS.NAME`), "PACOROIHeads", no caso do detector con atributos e "Res5ROIHeads" para o detector sen atributos.
2. No ficheiro `/workspace/main.py`, á hora de instanciar o avaliador de PACO, indicar, para o parámetro `eval_attributes` (liña 27), os valores `True` ou `False` segundo corresponda.

Cabe resaltar que o resto das configuracións descritas a partir deste momento son totalmente comúns para calquera das dúas alternativas anteriores.

No proxecto, dispónse dun script (`/workspace/run_paco.sh`) que, por defecto, executa, tanto o adestramento e a avaliación base, como as 10 repeticións do

axuste fino e avaliación para cada *shot*. A continuación, indícanse as partes do ficheiro resposables de cada tarefa, de xeito que se poden comentar as demais para executalas por separado (tamén sería válido lanzar os comandos directamente dende o terminal, pero habería que substituír os nomes das constantes polos seus valores, que están definidos ao inicio do script):

- **Adestramento e avaliación base**

  O código responsable é o seguinte:

  ```
  CUDA_LAUNCH_BLOCKING=1 python3 main.py --num-
     gpus 1 --config-file configs/paco/
     defrcn_det_r101_base.yaml --opts MODEL.
     WEIGHTS ${IMAGENET_PRETRAIN} OUTPUT_DIR ${
     SAVEDIR}/defrcn_det_r101_base
  ```

  En caso de utilizar máis de 1 GPU, débese modificar a opción `--num-gpus` para sinalar o dato correcto. Esto aplica a calquera das alternativas posteriores.

- **Avaliación base**

  As modificacións necesarias respecto ao anterior caso serían, primeiramente, indicar a opción `--eval-only` e, en segundo lugar, indicar o directorio no que se gardan os pesos do modelo a avaliar (ficheiro coa extensión .pth) a través da opción `MODEL.WEIGHTS`. De forma que o código quedaría así:

  ```
  CUDA_LAUNCH_BLOCKING=1 python3 main.py --num-
     gpus 1 --config-file configs/paco/
     defrcn_det_r101_base.yaml --eval-only --opts
      MODEL.WEIGHTS <directorio_pesos> OUTPUT_DIR
      ${SAVEDIR}/defrcn_det_r101_base
  ```

- **Axuste fino *few-shot* e avaliación FSOD**

  O código responsable é o seguinte:

  ```
  python3 tools/model_surgery.py --dataset paco
     --method remove --src-path ${SAVEDIR}/
     defrcn_det_r101_base/model_final.pth --save-
     dir ${SAVEDIR}/defrcn_det_r101_base
  BASE_WEIGHT=${SAVEDIR}/defrcn_det_r101_base/
     model_reset_remove.pth

  for repeat_id in 0 1 2 3 4 5 6 7 8 9
  do
      for shot in 1 2 3 5 10 30
  ```

```
    do
        python3 tools/create_config.py --
           dataset paco --config_root configs/
           paco --shot ${shot} --seed 0 --
           setting 'fsod'
        CONFIG_PATH=configs/paco/
           defrcn_fsod_r101_novel_${shot}
           shot_seed0.yaml
        OUTPUT_DIR=${SAVEDIR}/
           defrcn_fsod_r101_novel/fsrw-like/${
           shot}shot_seed0_repeat${repeat_id}
        python3 main.py --num-gpus 1 --config-
           file ${CONFIG_PATH} --opts MODEL.
           WEIGHTS ${BASE_WEIGHT} OUTPUT_DIR ${
           OUTPUT_DIR} TEST.PCB_MODELPATH ${
           IMAGENET_PRETRAIN_TORCH}
        rm ${CONFIG_PATH}
    done
done
python3 tools/extract_results.py --res-dir ${
   SAVEDIR}/defrcn_fsod_r101_novel/fsrw-like --
   shot-list 1 2 3 5 10 30
python3 tools/extract_results_per_cat.py --res-
   dir ${SAVEDIR}/defrcn_fsod_r101_novel/fsrw-
   like --shot-list 1 2 3 5 10 30
python3 tools/extract_attr_results.py --res-dir
    ${SAVEDIR}/defrcn_fsod_r101_novel/fsrw-like
    --shot-list 1 2 3 5 10 30
python3 tools/extract_attr_results_per_cat.py
   --res-dir ${SAVEDIR}/defrcn_fsod_r101_novel/
   fsrw-like --shot-list 1 2 3 5 10 30
```

O número de repeticións lanzadas para cada *shot* ven determinado pola lista que percorre `repeat_id` no bucle externo, que se pode modificar a conveniencia. Os tamaños de *shot* probados están definidos na lista que percorre `shot` no bucle interno. Ditos valores deben coincidir cos indicados para o lanzamento dos catro scripts finais de extracción de resultados.

- **Avaliación FSOD**

  As modificacións neste caso son análogas ás descritas para a avaliación base, neste caso, o lanzamento do main quedaría da seguinte forma:

```
python3 main.py --num-gpus 1 --config-file ${
   CONFIG_PATH} --eval-only --opts MODEL.
```

```
WEIGHTS <directorio_pesos> OUTPUT_DIR ${
OUTPUT_DIR} TEST.PCB_MODELPATH ${
IMAGENET_PRETRAIN_TORCH}
```

Finalmente, unha vez preparado o script, execútase (dentro do contedor) co comando:

```
bash /workspace/run_paco.sh <nome_experimento>
```

En caso de que se produza un desborde de memoria na GPU, un dos parámetros que se pode modificar é o tamaño de *batch*, que se establece, por defecto, a 16. No caso de reducilo, teríase que reducir tamén a taxa de aprendizaxe de forma proporcional para manter unhas condicións equivalentes á experimentación levada a cabo neste traballo. Para realizar esta modificación, abrir o ficheiro de configuración correspondente (base ou FSOD dun tamaño de *shot* determinado) que se atopa en `/workspace/configs/paco` e indicar os valores que se desexen para `SOLVER.IMS_PER_BATCH` e `SOLVER.BASE_LR`.

Bibliografía

[1] V. Ramanathan, A. Kalia, V. Petrovic, Y. Wen, B. Zheng, B. Guo, R. Wang, A. Marquez, R. Kovvuri, A. Kadian *et al.*, "Paco: Parts and attributes of common objects," *arXiv preprint arXiv:2301.01795*, 2023.

[2] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang, "Defrcn: Decoupled faster r-cnn for few-shot object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 8681–8690.

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

[4] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 740–755.

[6] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

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

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[9] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," *arXiv preprint arXiv:2009.07485*, 2020.

[10] V. Panchbhaiyye and T. Ogunfunmi, "A fifo based accelerator for convolutional neural networks," https://www.researchgate.net/figure/Convolution-operation-in-a-CNN_fig1_341083522, 2020, data de consulta: 25 de xuño de 2023.

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[12] Baeldung - Tarnum Java SRL, "Intersection over union for object detection," https://www.baeldung.com/cs/object-detection-intersection-vs-union, data de consulta: 21 de xuño de 2023.

[13] Jonathan Hui, "map (mean average precision) for object detection," https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173, data de consulta: 21 de xuño de 2023.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, vol. 1, 2005, pp. 886–893.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[16] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, pp. 154–171, 2013.

[17] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 2014, pp. 391–405.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image

is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[21] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer, 2020, pp. 213–229.

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

[23] A. Bar, X. Wang, V. Kantorov, C. J. Reed, R. Herzig, G. Chechik, A. Rohrbach, T. Darrell, and A. Globerson, "Detreg: Unsupervised pretraining with region priors for object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 605–14 615.

[24] X. Zhang, F. Liu, Z. Peng, Z. Guo, F. Wan, X. Ji, and Q. Ye, "Integral migrating pre-trained transformer encoder-decoders for visual object detection," *arXiv preprint arXiv:2205.09613*, 2022.

[25] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[26] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8420–8429.

[27] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin, "Meta r-cnn: Towards general solver for instance-level low-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9577–9586.

[28] Y. Xiao, V. Lepetit, and R. Marlet, "Few-shot object detection and viewpoint estimation for objects in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3090–3106, 2022.

[29] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai, "Few-shot object detection with attention-rpn and multi-relation detector," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4013–4022.

[30] H. Chen, Y. Wang, G. Wang, and Y. Qiao, "Lstd: A low-shot transfer detector for object detection," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

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

[32] A. Gupta, P. Dollar, and R. Girshick, "Lvis: A dataset for large vocabulary instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5356–5364.

[33] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, pp. 303–338, 2010.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[35] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[36] V.Ñair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[37] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[38] Keras, "Bounding box formats," https://keras.io/api/keras_cv/bounding_box/formats/, data de consulta: 26 de xuño de 2023.

[39] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[40] Gigabyte Technology, "Server g291-281," https://www.gigabyte.com/es/Enterprise/GPU-Server/G291-281-rev-100, data de consulta: 19 de xuño de 2023.

[41] Intel Corporation, "Procesador intel xeon silver 4214," https://www.intel.es/content/www/es/es/products/sku/193385/intel-xeon-silver-4214-processor-16-5m-cache-2-20-ghz/specifications.html, data de consulta: 19 de xuño de 2023.

[42] NVIDIA Corporation, "Data sheet: Quadro p6000," 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, data de consulta: 19 de xuño de 2023.

[43] ——, "Data sheet: Quadro rtx 8000," https://www.nvidia.com/content/dam/en-zz/Solutions/design-visualization/quadro-product-literature/quadro-rtx-8000-us-nvidia-946977-r1-web.pdf, data de consulta: 19 de xuño de 2023.

[44] ——, "Nvidia a30 datasheet," https://www.nvidia.com/content/dam/en-zz/Solutions/data-center/products/a30-gpu/pdf/a30-datasheet.pdf, data de consulta: 19 de xuño de 2023.

[45] Gigabyte Technology, "Server g482-z54," https://www.gigabyte.com/Enterprise/GPU-Server/G482-Z54-rev-100, data de consulta: 19 de xuño de 2023.

[46] Advanced Micro Devices, Inc, "Amd epyc 7413," https://www.amd.com/es/products/cpu/amd-epyc-7413, data de consulta: 19 de xuño de 2023.

[47] NVIDIA Corporation, "Nvidia a100 datasheet," https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf, data de consulta: 19 de xuño de 2023.

[48] The CentOS Project, "The centos project," https://www.centos.org/, data de consulta: 19 de xuño de 2023.

[49] AlmaLinux OS Foundation, "Almalinux os - forever-free enterprise-grade operating system," https://almalinux.org/, data de consulta: 19 de xuño de 2023.

[50] Docker Inc, "Docker: Accelerated, containerized application development," https://www.docker.com/, data de consulta: 19 de xuño de 2023.

[51] NVIDIA Corporation, "Cuda zone," https://developer.nvidia.com/cuda-zone, data de consulta: 19 de xuño de 2023.

[52] Free Software Foundation, Inc, "Gcc, the gnu compiler collection," https://gcc.gnu.org/, data de consulta: 19 de xuño de 2023.

[53] Python Software Foundation, "Welcome to python," https://www.python.org/, data de consulta: 19 de xuño de 2023.

[54] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with numpy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[56] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.

[57] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.