BACHELOR´S THESIS / ENGINEERING 2024

Com de responsable és la teva marca? Identificació de marques en escombraries generades pels humans

Abril Masgrau Turró

Universitat Pompeu Fabra
Barcelona

Com de responsable és la teva marca? Identificació de marques en escombraries generades pels humans

TREBALL FI DE GRAU DE

Abril Masgrau Turró

Director: Horacio Saggion

Grau en Enginyeria Matemàtica en Ciència de Dades

Curs 2023-2024

Universitat Pompeu Fabra Barcelona
Escola d'Enginyeria

Als meus pares, pel seu suport i per la paciència

Agraïments

M'agradaria començar per donar les gràcies al meu tutor Horacio, per haver-me ajudat amb el treball, guiar-me en tot moment i per respectar el meu ritme. També vull donar les gràcies a la meva família i amics pel seu suport continu i paciència, sobretot en els moments estressants. Per últim, m'agradaria agrair al meu gos i els meus gats per fer-me companyia a les nits de TFG.

Resum

Un dels problemes que s'enfronta la societat actualment, és la quantitat d'escombraries que acaben tirades per carrer. Als llocs públics de les ciutats s'hi poden observar llaunes, bosses de plàstic, envasos… que generalment pertanyen a marques. Aquest projecte té com a objectiu desenvolupar un sistema de reconeixement d'imatges per tal que, a partir d'una imatge d'escombraries, detecti a quines marques pertanyen els productes i, per tant, tenir la informació de quines són les més contaminants.
Utilitzant els logos, s'estudien i avaluen dos mètodes per identificar les marques, detecció d'objectes i reconeixement de text. El projecte proposa un sistema que combina els dos mètodes per tal de millorar la detecció.
Finalment, es proporciona un prototip d'aplicació que través de la col·laboració dels usuaris, es poden penjar imatges d'escombraries, reconèixer les marques i crear estadístiques per conscienciar tant als ciutadans com a les empreses responsables.

Resumen

Uno de los problemas que se enfrenta la sociedad actualmente, es la cantidad de residuos que acaban tirados por la calle. En los lugares públicos de las ciudades se pueden observar latas, bolsas de plástico, envases… que generalmente pertenecen a marcas. Este proyecto tiene el objetivo de desarrollar un sistema de reconocimiento de imágenes con el fin de que a partir de una imagen de basura, detecte a qué marcas pertenecen los productos y, por lo tanto, tener la información de cuáles son las más contaminantes.
Utilizando los logos, se estudian y evalúan dos métodos para identificar las marcas, detección de objetos y reconocimiento de texto. El proyecto propone un sistema que combina los dos métodos con tal de mejorar la detección.
Finalmente, proporciona un prototipo de aplicación que, a través de la colaboración de los usuarios, se pueden subir imágenes de residuos, reconocer las marcas y crear estadísticas para concienciar tanto a los ciudadanos como a las empresas responsables.

Abstract

One of the problems that society is currently facing, is the amount of waste that ends up thrown on the streets. In public spaces in cities, you can see cans, plastic bags, packaging, etc. which usually belong to companies. This project aims to develop an image recognition system in order to, from an image of garbage, detect what brands they belong to and have information on which ones pollute the most.
Using the logos, two methods are studied and evaluated to identify the brand, object detection and text recognition. This project proposes a system that combines both methods to improve detection.
Finally, it provides an application prototype where, through the collaboration of users, images of garbage are uploaded, logos and brands are recognized, and then statistics are shown to raise awareness among both citizens and responsible companies.

Índex

Introducció **1**
1.1 Plantejament del problema 1
1.2 Objectiu 1
Estat de l'art **2**
Reconeixement d'imatges **3**
3.1 Convolutional Neural Networks (CNN) 3
Marc teòric detecció d'objectes **5**
4.1 Two-stage detectors 5
4.2 One-stage detectors 6
Marc teòric reconeixement de text en imatges **9**
5.1 Arquitectura sistemes OCR 9
5.2 Models existents 10
Marc pràctic detecció d'objectes **12**
6.1 Eines utilitzades 12
6.2 Procediment 12
6.3 Resultats 14
6.4 Conclusions 21
Marc pràctic reconeixement de text **22**
7.1 Eines utilitzades 22
7.2 Procediment 22
7.3 Resultats 25
7.4 Conclusions 25
Sistema final **26**
8.1 Creació d'un nou conjunt d'imatges 26
8.2 Redacció codi del nou sistema 26
8.3 Resultats 27
Prototip d'aplicació per registrar la distribució d'escombraries i marques **28**
Conclusions i treball futur **30**
10.1 Conclusions 30
10.2 Treballs futurs 30
Apèndix **34**
Apèndix 1: Llista de marques del conjunt d'imatges 34
Apèndix 2: Imatges del conjunt augmentat 34
Apèndix 3: Github projecte 35
Apèndix 4: Entrenament amb YOLOv8 36
Apèndix 5: Mètriques d'avaluació 36
Apèndix 6: Prototip d'aplicació 37

Llista de figures

Figura 1: Arquitectura de les Xarxes Neuronals Convolucionals. [6]..... 3
Figura 2: Exemple de funcionament de la capa de convolució.[8]..... 4
Figura 3: Exemple de Max i Avg Pooling. [9]..... 4
Figura 4: Arquitectura YOLO[14]..... 6
Figura 5: Fórmula Intersection over Union. [15]..... 7
Figura 6: Estructura de YOLO. [14]..... 7
Figura 7: Arquitectura Tesseract OCR. [20].....10
Figura 8: Arquitectura EasyOCR. [23].....11
Figura 9: Exemple d'imatges del dataset inicial..... 13
Figura 10: Exemple anotació del logo de Milka..... 13
Figura 11: Matriu de confusió del primer entrenament.....15
Figura 12: Evolució de les mètriques al llarg dels epochs del segon entrenament..... 16
Figura 13: Matriu de confusió normalitzada segon entrenament..... 16
Figura 14: Exemples de detecció correctes..... 17
Figura 15: Exemples de deteccions errònies..... 17
Figura 16: Evolució de les mètriques al llarg dels epochs del tercer entrenament.....18
Figura 17: Matriu de confusió normalitzada tercer entrenament..... 18
Figura 18: Corba de precisió i sensibilitat.....19
Figura 19: Evolució de les mètriques al llarg dels epochs del quart entrenament..... 20
Figura 20: Matriu de confusió normalitzada quart entrenament..... 20
Figura 21: Exemples de les deteccions del quart entrenament.....21
Figura 22: Exemple de detecció Tesseract OCR..... 23
Figura 23: Exemple 1 de detecció PaddleOCR..... 23
Figura 24: Exemple 2 de detecció PaddleOCR..... 23
Figura 25: Exemple de detecció EasyOCR..... 24
Figura 26: Exemple de combinació dels dos mètodes per obtenir el resultat final..... 27

Llista de taules

Taula 1: Nombre de productes per cada marca en el dataset inicial i augmentat..... 14
Taula 2: Mètriques de l'últim epoch del segon entrenament..... 15
Taula 3: Mètriques de l'últim epoch del tercer entrenament..... 17
Taula 4: Mètriques de l'últim epoch del quart entrenament..... 19
Taula 5: Taula de resultats pels diferents models provats.....25
Taula 6: Nombre de productes per cada marca en el dataset extra.....26
Taula 7: Mètriques per cada un dels mètodes estudiats.....27

Capítol 1

Introducció

1.1 Plantejament del problema

Aquests últims anys ha augmentat exageradament la producció d'envasos i en conseqüència, la contaminació de la natura. Cada cop es produeixen més envasos de plàstic com ampolles, bosses, llaunes... i la majoria d'ells són d'un sol ús. Quan els tirem a les escombraries poden acabar a l'abocador, ser incinerats o reciclats. Tanmateix, per causes naturals com el vent o pluja o simplement per la despreocupació dels humans, aquests residus també poden arribar a escampar-se pel medi ambient.

Segons les dades que va proporcionar Greenpeace el 2018, del nombre total de plàstics que es produeixen, solament un 9% es recicla, un 12% s'incinera i un 79% acaba en abocadors o en el medi ambient. A més veiem a les ciutats que hi ha localitzacions concretes amb una alta concentració d'envasos, per exemple als parcs hi abunden llaunes de refrescos, bosses de patates, de pipes... . Pels carrers es troben moltes escombraries generades pels humans i que no només representa un problema pel medi ambient sinó que visualment deterioren la ciutat i a més pot tenir repercussions per les empreses a les quals pertanyen els envasos. [1]

Les empreses les quals generen gran quantitat d'envasos, s'haurien d'implicar no només en fer els envasos reciclables o compostables sinó també en assegurar que arriben a les escombraries correctes i no acabin al medi ambient. Per exemple a Nova York, es va multar a l'empresa PepsiCo per la contaminació dels seus productes al riu Buffalo. Un 17% dels envasos recollits al riu pertanyien a PepsiCo i van ser considerats perjudicials per al medi ambient encara que PepsiCo es presenta com una empresa conscienciada pel medi ambient. [2]

Per tant, hi ha molts interessats en reduir la quantitat d'escombraries que es troben escampades a les ciutats i al medi ambient. Per això aquest projecte planteja la idea per identificar localitzacions concretes on s'acumulen envasos de les mateixes marques. Per exemple, trobar llocs de Barcelona on hi ha moltes escombraries que corresponen a la marca Coca-Cola. Això permetria conscienciar les empreses i aquestes podrien augmentar la seva campanya per reciclar en aquests llocs concrets. O també podria ser d'utilitat per l'ajuntament o altres associacions que lluiten pel medi ambient.

1.2 Objectiu

L'objectiu d'aquest projecte, és experimentar amb diversos mètodes de reconeixement d'imatges per poder reconèixer els logos que apareixen a diverses imatges de productes de brossa trobats per carrer, i així identificar quines empreses contribueixen amb la contaminació. Investigar quines tècniques pel reconeixement d'imatges existeixen i entrenar-les o modificar-les per tal que reconeguin logos i avaluar quin dels models té millors resultats.

Un cop trobat un sistema per reconèixer els logos, fer un prototip d'una aplicació per poder fer una anàlisi de la distribució de la brossa i maques a les ciutats.

Capítol 2

Estat de l'art

Ja fa uns anys que en el sector de visió per computadors, s'utilitzen majoritàriament arquitectures d'aprenentatge profund, especialment les CNN (Convolutional Neural Networks). Pel que fa a la detecció d'objectes, els models més utilitzats són Faster R-CNN [12], YOLOv8 [24], SSD (Single-Shot Detector) [13]..., al llarg del projecte s'aniran explicant amb més detall. Per la part de reconeixement de text, hi ha algoritmes especialitzats en detectar a on hi ha text com el CRAFT, EAST, DB… i altres que fan la funció de reconèixer el text com les CRNN, SRN… . Actualment, hi ha molts enginys que uneixen aquests algoritmes per crear una arquitectura que localitzen i reconeixen text, com per exemple TesserractOCR [28], EasyOCR [23] , PaddleOCR [21] i molts altres.
Aquest projecte se centra en reconèixer logos de marques a partir d'una imatge, per tant, a continuació s'analitzen quins avenços hi ha hagut fins ara respecte al problema de *Logo Detection* i detecció d'objectes a les escombraries.

El 2019, O. Orti et al. [3] explora la idea de detectar logos en temps real a les xarxes socials. Desenvolupa un sistema que agafa imatges provinents de plataformes digitals i llavors reconeix i localitza una sèrie de marques. Aquest sistema utilitza dues Xarxes Neuronals Convolucionals, SSD InceptionV2 i Faster Atrous InceptionV4. Per la part experimental, treballa amb dues marques, Estrella Damm i el Futbol Club Barcelona i conclou que SSD és un bon algoritme per la rapidesa que proporciona però no té un bon rendiment amb objectes petits.

El 2020, Wang, J., i companys [4] publiquen un article que proposa la creació d'un *dataset* format per 3.000 logos diferents i 158,652 imatges. A més de crear un dels *datasets* més grans i diversos de logos, també proposa un mètode anomenat Logo_Yolo. Aquest mètode utilitza YOLOv3 amb la incorporació de pèrdua focal per balancejar el desequilibri entre les classes, i pèrdua CIoU (*Complete Intersection over Union*) que fa millorar la precisió. L'article també fa una comparació avaluant diversos mètodes com Faster R-CNN, SSD, RetinaNet, YOLOv3, Logo-Yolo… . El que obté una precisió més alta és Logo-Yolo seguit per YOLOv3.

Pel que fa a detectar i identificar objectes a les escombraries, White et al. (2020) [33], desenvolupa WasteNet, un model de classificació d'escombraries per tal de separar els residus segons el seu tipus. Aquest model se centra en classificar en categories la brossa per tal de poder saber a quines escombraries van. L'article dissenya un model de xarxa neuronal profunda, que utilitza les CNN per tal de classificar les imatges en les categories de plàstic, cartó, paper, vidre… . Els resultats de WasteNet tenen una precisió alta, millorant altres models com VGG16, ResNet169… .

Tornant als logos, K. Rezkiani, I. Nurtanio and Syafaruddin, 2022 [36], van proposar un mètode, que utilitza YOLOv4 per tal de reconèixer logos de diferents organitzacions i institucions. Amb l'objectiu de classificar els documents de forma automàtica, l'article es centra en la recopilació de diplomes universitaris formant un *dataset* amb els logos de les diferents universitats. Amb l'entrenament del conjunt d'imatges i YOLOv4, els resultats quant a precisió són prometedors, amb el 93.73%.

Capítol 3

Reconeixement d'imatges

El reconeixement d'imatges s'entén com el procés d'identificar objectes o característiques a una imatge o vídeo. L'objectiu és que els ordinadors puguin interpretar els inputs visuals d'una manera similar a la dels humans, identificant i classificant objectes.

Hi ha molts tipus de sistemes que estan dissenyats per reconèixer diferents objectes com cares de persones, text, patrons, vores... . Cada un d'aquestes tècniques s'especialitza en diferents característiques i n'hi ha de més específics que altres. En línies generals són els següents *Object Detection, Text Recognition, Feature Recognition, Pattern Recognition, Facial Recognition, Scene Understanding, Image Localization*... . [5]

Com ja s'ha explicat a l'inici, l'objectiu que es planteja en aquest projecte, és la implementació d'un sistema que a partir d'imatges d'escombraries o deixalles identifiqui diverses marques d'envasos que hi apareixen. Per fer aquesta tasca que s'han de detectar marques, hi ha principalment dues tècniques que poden ser les més adequades, reconeixement de text i detecció d'objectes. Pel cas de reconeixement de text funcionaria a partir de detectar les marques a partir del nom i en el cas de la detecció d'objectes utilitzaria tot o part dels logos.

Els dos casos tenen avantatges i limitacions, per exemple en el cas d'utilitzar reconeixement de text limita els productes on no apareix el nom de l'empresa. Per tant, els logos on no apareix el nom com Starbucks o Carrefour no podrien ser detectats. Per altra banda, utilitzant detecció d'objectes ens limita la quantitat de logos segons el nombre de marques amb les quals s'entrena el model.

En el llarg del projecte, s'utilitzen molts d'algorismes i tècniques que utilitzen les Xarxes Neuronals Convolucionals, les CNN, així doncs abans de continuar amb els tipus de reconeixement d'imatges, s'expliquen en què consisteixen aquestes xarxes.

3.1 Convolutional Neural Networks (CNN)

Les Xarxes Neuronals Convolucionals, en anglès *Convolutional Neural Networks* (CNN), son arquitectures de *deep learning* que s'utilitzen per al reconeixement i processament d'imatges. Aquest tipus de xarxes neuronals artificials tenen una gran habilitat en el reconeixement de patrons en imatges. Estan formades principalment per tres capes. Inicialment, consta d'un input, que si assumim que treballem amb imatges, l'input és una matriu de mida *height* x *width* x 3, altura, amplada i profunditat (valors RGB) representant píxels. A continuació s'expliquen les capes principals [6], [7], [8].

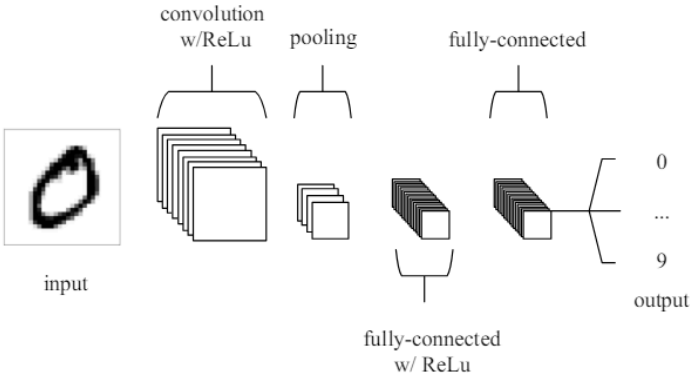


Figura 1: Arquitectura de les Xarxes Neuronals Convolucionals. [6]

1. La capa de convolució, *convolution layer* consisteix en aplicar una sèrie d'operacions a l'input utilitzant uns filtres per trobar patrons o característiques en les regions de la imatge. Aquests filtres, també denominats *kernel*s, poden tenir diferents dimensions i van recorrent la imatge d'input Com mostra la **Figura 2**, calculen el producte escalar i van creant una nova matriu que serà l'output anomenat *feature map*. Després de l'operació de convolució, s'aplica una funció d'activació ReLU (Rectified Lineal Unit) que estableix a zero els valors negatius i per tant, introdueix una no linealitat al model i restringeix els valors resultants.

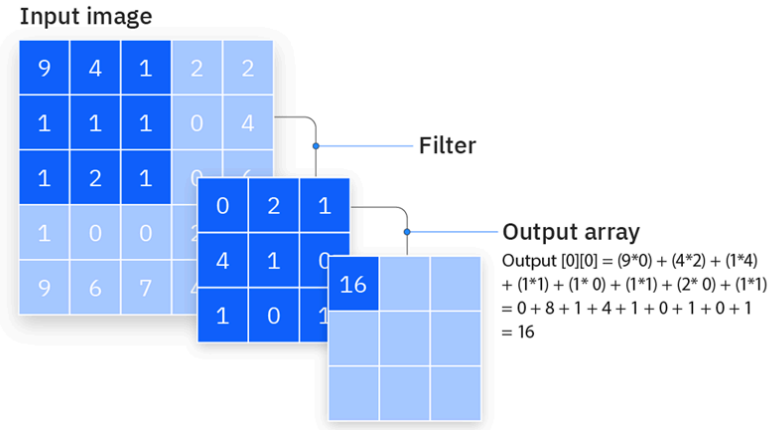


Figura 2: Exemple de funcionament de la capa de convolució.[8]

2. La capa d'agrupació, *pooling layer* té l'objectiu de disminuir les dimensions de l'input reduint el nombre de paràmetres que ha d'aprendre la xarxa i redueix la memòria i evita el *overfitting*. Igual que la capa de convolució, aquesta recorre la imatge i aplica un filtre, però que en aquest cas no té cap pes sinó que té la funció d'agrupar, les més comunes són les següents:

- *Max pooling* o agrupació màxima que el filtre selecciona el píxel amb el valor més alt.
- *Average pooling* o agrupació mitjana que calcula el valor mitjà dins la regió corresponent.

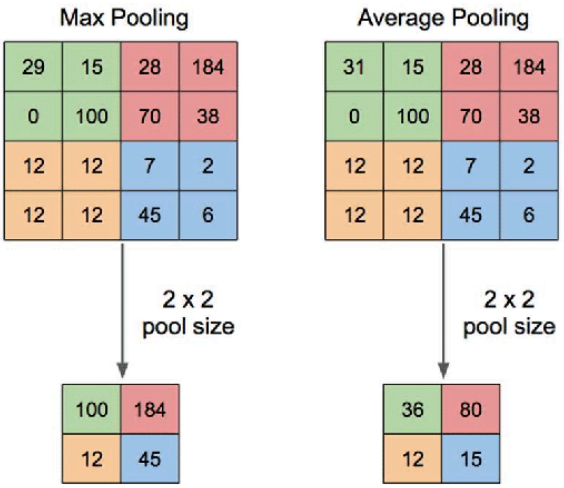


Figura 3: Exemple de Max i Avg Pooling. [9]

3. La capa totalment connectada, *fully connected layer* és l'encarregada de produir una classificació i puntuació de les classes possibles. Basant-se en les característiques que han extret les anteriors capes, generen un vector que conté les probabilitats per cada una de les classes.

Capítol 4

Marc teòric detecció d'objectes

Pels humans és fàcil reconèixer i localitzar objectes que veiem, però pels ordinadors és una tasca complicada on s'han buscat solucions per aproximar aquesta virtut dels humans. La detecció d'objectes, en visió per computadors, és una tècnica per identificar i localitzar instàncies d'objectes en imatges o vídeos. Aquesta tècnica utilitza branques de *machine learning* o *deep learning* com per exemple *image processing*, per detectar la posició i l'àrea en la qual apareix l'objecte. [10]

La detecció d'objectes té moltíssimes utilitats, de les més conegudes serien el reconeixement facial, el rastreig o recompte de persones. Tot i això, hi ha moltes aplicacions relacionades amb la seguretat i salut de les persones, com en medicina i el reconeixement d'anomalies en radiografies que poden ajudar al diagnòstic. O també en conducció autònoma per identificar cotxes, vianants o altres objectes que obstrueixin la via com ja fa la tecnologia ADAS (Advanced driver-assistance system).

Una manera de categoritzar les diverses formes d'enfocar la detecció d'objectes és entre *machine learning* i *deep learning*. Utilitzant *machine learning*, s'han de seleccionar manualment les característiques a tenir en compte per fer les prediccions. Les tècniques més comunes són el de Viola-Jones per detectar cares humanes, *Aggregate channel features* (ACF) i *Histogram of Oriented Gradients* (HOG). Per altra banda, utilitzant *deep leaning*, les característiques es seleccionen de forma automàtica. [10]

Utilitzant *deep learning*, es poden dividir en dos tipus de detecció d'objectes segons la xarxa que utilitzen, detectors d'una etapa (*One-Stage Detectors*) i de dues etapes (*Two-Stage Detectors)*.

4.1 Two-stage detectors

Aquests mètodes acostumen aconseguir més precisió en els resultats, però alhora solen ser més lents. La seva arquitectura consisteix principalment en dues parts, proposar regions i classificar objectes. La primera part es basa en generar i proposar regions de la imatge on pot haver-hi un objecte, i així centrar-se en les zones òptimes, conegudes com a *regions of interest* (RoI). A la segona part, es redefineixen les localitzacions dels objectes i es classifiquen per classes amb una puntuació. Hi ha diversos algoritmes que utilitzen aquesta arquitectura i com s'ha explicat anteriorment, les CNN, tenen un rol vital en la majoria de les tècniques de reconeixement d'imatges, a continuació, es mencionen algoritmes que són two-stage detectors. [11]

Region-based Convolutional Network method (R-CNN)

L'algoritme R-CNN utilitza *selective search* per extreure un nombre menor de regions proposades. A través de *Region Proposal Networks*, fa una selecció de quines zones està ubicat l'objecte i un cop trobades aquestes regions, s'utilitza un CNN per extreure'n les característiques i llavors classificar les regions. [12]

Fast Region-based Convolutional Network method (Fast R-CNN)

A diferència del R-CNN, la xarxa primer processa la imatge amb diverses capes convolucionals i d'agrupació i llavors per cada regió proposada, es passa a una capa totalment connectada i retorna l'output. Aquest avanç es va fer amb la intenció de reduir el temps que demora el R-CNN. [12]

4.2 One-stage detectors

Aquests mètodes calculen la classificació de les regions directament, sense generar les regions proposades. Això disminueix el temps, però alhora també disminueix la precisió. Els seus algorismes més utilitzats són el SSD i YOLO.

Single Shot Detection (SSD)

Els SDD és un algoritme de detecció d'objectes que es caracteritza per la seva velocitat i eficiència i, per tant s'acostuma a utilitzar en aplicacions a temps real. Utilitza diverses capes de convolució amb resolucions espacials decreixents i una última capa per classificar els objectes i les *bounding boxes*. [13]

You Only Look Once (YOLO)

YOLO [14] es un algorisme que utilitza Xarxes Neuronals per detectar objectes en temps real que es caracteritza per la seva velocitat i precisió. Els seus autors el classifiquen com a un problema de regressió de *bounding boxes*, en comptes d'un problema de detecció d'objectes. Al contrari que els models que hem vist anteriorment, YOLO analitza la imatge sencera durant l'entrenament i, per tant, té en compte informació de context de les classes i les seves aparences. La seva arquitectura és la següent:

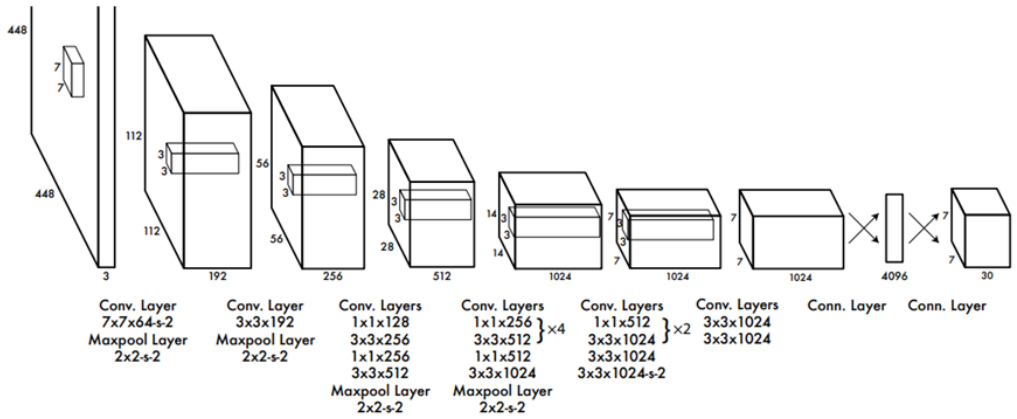


Figura 4: Arquitectura YOLO[14]

L'arquitectura està inspirada en GoogLeNet i té un total de 24 capes de convolució seguides per 2 capes totalment connectades. Abans de les capes de convolució, la imatge de l'input es reajusta per tenir una mida de 448x448. Com s'observa a la **Figura 4**, es van alternant capes de convolució amb filtres de 1x1 per reduir el nombre de canals i seguidament s'apliquen capes convolucionals de 3x3. L'output final és un tensor de prediccions de mida 7x7x30.

<u>Funcionament de la detecció unificada</u>

Inicialment es divideix cada imatge en SxS quadrícules del mateix tamany. Cada una d'aquestes cel·les és responsable de detectar els objectes que el seu centre estigui dintre la cel·la.

Llavors cada una calcula B *bounding boxes* i una puntuació de confiança per cada una d'elles. Aquesta puntuació significa com de segur està el model de que a la *bounding box* apareix un objecte i com de precís és la predicció de la *bounding box*, pot anar de 0.0 a 1.0. Per tant, obtenim *bounding boxes* constituïdes de 5 xifres, la posició x, posició y, amplada, altura i puntuació. Per calcular la puntuació s'utilitza la següent fórmula:

On IOU és la divisió entre la intersecció de l'àrea predita i la real entre l'àrea de la seva unió.

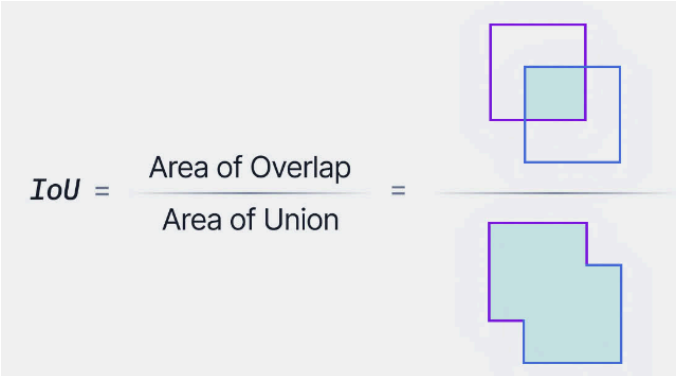


Figura 5: Fórmula Intersection over Union. [15]

Addicionalment, cadascuna de les cel·les de la quadrícula, calcula C probabilitats condicionals Pr(Class **i** | Object), és a dir, donat un objecte dins la cel·la, quina probabilitat té que sigui de la classe *i*, on *i* és una de les classes amb les que s'entrena el model. Aquestes probabilitats són diferents per cada una de les cel·les. És important recalcar que cada cel·la només pot predir una classe i això és una gran carència de YOLO.

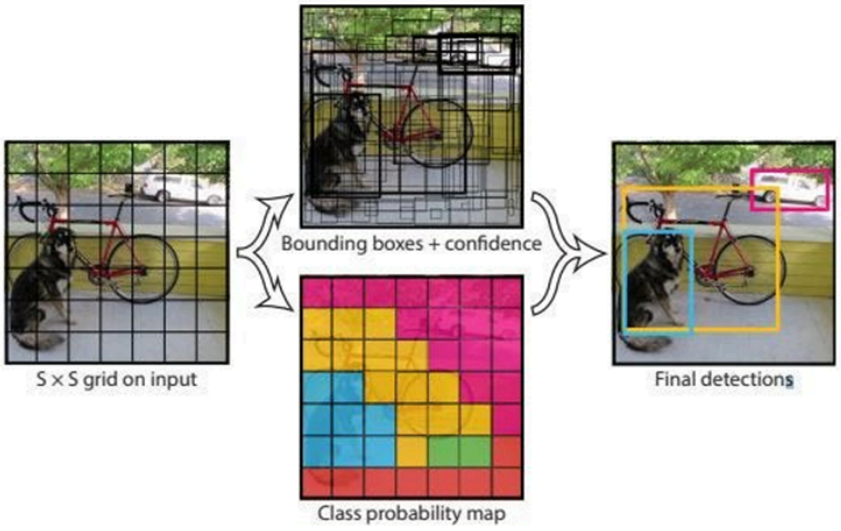


Figura 6: Estructura de YOLO. [14]

A la **Figura 6**, s'observen les *bounding boxes* i les puntuacions de confiança juntament amb els resultats i prediccions. Com que hi ha SxS cel·les, l'output de les prediccions és un tensor de S·S·(B*5 + C), ja que cada *bounding box* té 5 components i a més tenim les probabilitats de les classes.

Llavors per cada una de les cel·les hi ha B *bounding boxes* de les quals no totes són rellevants. Per intentar seguir amb només les que són rellevants, determinem un *threshold* que es compara amb la puntuació de confiança i agafa les que tenen valors més grans que el *threshold*. Tot i eliminar-ne algunes, pot passar que vàries d'elles facin referència al mateix objecte, per això YOLO utilitza *Non-max suppression*. Aquesta tècnica elimina les bounding boxes que es solapen severament amb altres, utilitzant també un *threshold*. [14, 16]

Avantatges YOLO

En línies generals n'hi ha quatre punts positius de YOLO [16]:

- Velocitat: YOLO pot processar les imatges a 45 *frames per second* i, per tant, comparant amb altres algoritmes, està molt ben considerat dintre els de processament a temps real. A més, manté una precisió que pot doblar als altres sistemes.
- Precisió: Per la velocitat que té, YOLO té una alta precisió de detecció. Especialment, té pocs errors de *background*.
- Generalització: Especialment, les últimes versions de YOLO, tenen una bona generalització i, per tant, un bon rendiment davant de diversos escenaris o inputs diferents.
- Open Source: A més, YOLO és públic i, per tant, té un lliure accés i permet que tothom hi pugui millorar el model.

Limitacions YOLO

Evidentment, no tot són trets positius, YOLO també té limitacions i carències que li deixen marge de millora [14].

- Com s'ha explicat anteriorment, per cada cel·la, YOLO només pot predir dues *bounding boxes* i a més han de pertànyer a la mateixa classe. Això fa que YOLO tingui dificultats en detectar objectes petits i molt junts.
- Tot i tenir una bona generalització, especialment per versions antigues, si els objectes tenen canvis significatius en les proporcions o configuracions respecte al *dataset* d'entrenament, el model no acaba de generalitzar correctament i pot tenir errors.

Versions YOLO

L'algoritme de YOLO té moltes versions [15], des de YOLOv1 que va sortir el 2015, a YOLOv8 que es va publicar el 2023. Aquest últim és molt recent i per tant s'estan fent canvis constantment. El YOLOv7 va suposar una gran millora específicament en el camp de la detecció d'objectes tant en velocitat com en precisió. Actualment, hi ha YOLOv8 que millora lleugerament la versió v7 pel que fa a velocitat i precisió.

Capítol 5

Marc teòric reconeixement de text en imatges

El reconeixement de text en imatges és el procés d'identificar i extreure text a partir d'una foto, vídeo, document... . Utilitza la tècnica OCR, Optical Character Recognition per reconèixer caràcters concrets i la qual ha evolucionat en els últims anys i pot identificar caràcters de diferents fons, mides i fins i tot escrit a mà. Aquesta tècnica busca passar d'una imatge amb text, a un format de text digital que pugui ser utilitzat per ordinadors per analitzar, buscar, editar… . [17]

Els sistemes OCR més simples utilitzen una base de dades que contenen símbols variats i utilitzant algoritmes els relaciona amb els caràcters de la imatge. Altres sistemes més complexos, tenen en compte altres canvis que pot tenir un text com la il·luminació, el fons, textura… i utilitzen xarxes neuronals.

5.1 Arquitectura sistemes OCR

Hi ha molts enginys i softwares que utilitzen un conjunt d'algoritmes per tal de preprocessar el text, localitzar el text i classificar els caràcters. Cada un utilitza eines i tècniques diverses, però seguint aquesta estructura [17, 18]:

1. *Preprocessing*: Inicialment, s'ha de convertir la imatge en format digital, per exemple, un escaneig. Llavors s'han de preprocessar les imatges per millorar la seva qualitat i reduir la informació innecessària. Això s'aconsegueix a través de vàries tècniques, algunes d'elles són les següents:

    - Corregir l'alineació: A l'escanejar un text o tirar una foto, el text pot quedar mal alineat. És important alinear l'input de manera que el text quedi totalment en vertical o en horitzontal.

    - Reduir el "soroll": Eliminant els punts extrems d'intensitat, tant negatius com positius, suavitzant les vores i eliminant altres artefactes no desitjats.

    - Binarització: Consisteix en transformar la imatge en binària, és a dir, cada píxel només pot tenir dos colors, o blanc o negre. Fixant un *thereshold*, els píxels es classifiquen en fons o primer pla, això ajuda a separar el text del fons.

    - Esqueletització: Cada imatge pot tenir un text amb diferent tipologia de lletra, l'esqueletització és una tècnica que converteix tota la lletra de la mateixa gruixudesa.

2. *Segmentació*: Aquest pas consisteix en dividir la imatge en parts significatives per diferenciar el text del fons. Per un costat es separen les diverses línies, també es divideixen amb paraules i per últim s'identifiquen cada un dels caràcters. Tot això identificant els espais en blanc entre elles.

3. *Feature extraction:* Un cop s'han dividit en parts, s'extreuen diverses característiques com poden ser la forma, mida, orientació, patrons... . Aquestes característiques serveixen més endavant per classificar el text. Avui en dia molts models utilitzen arquitectures com CNN, RNN i LSTM, per identificar els *features*.

4. *Classificació*: L'etapa de classificació consisteix en utilitzar les característiques extretes anteriorment i assignar-les a una classe o lletra ja vista anteriorment. Per fer la classificació, hi ha diverses tècniques per exemple Xarxes Neuronals, SVM (*Support Vector Machine*)..., que busquen un caràcter ja vist en un *dataset* d'entrenament que coincideixi amb el text a reconèixer.

5. *Post-processing:* Aquest pas és per millorar els resultats i reduir petits errors que s'han comès a les etapes anteriors. Utilitza el context del text per corregir i acabar de concretar cada paraula i frase.

Hi ha tècniques de *Natural Language Processing* que permeten fer-ho o a través de diccionaris per comprovar l'existència de les paraules.

5.2 Models existents

Actualment, hi ha molts sistemes creats de reconeixement de text, alguns ja venen amb models entrenats que permeten utilitzar-los directament i que a més són de codi obert. Aquests sistemes estan formats de diversos algoritmes, uns per detectar i localitzar el text i altres per reconèixer els caràcters. A continuació s'expliquen els tres models que s'han utilitzat per a aquest projecte.

Tesseract OCR:

Tesseract [19] és un motor de OCR desenvolupat per HP en el 1994 i que actualment és de codi obert i pertany a Google. Està entrenat per una varietat ampla d'idiomes, 116, però permet a l'usuari entrenar un model propi i a més té una bona precisió. Utilitza el reconeixement dels caràcters, la coincidència de patrons i el context per identificar el text. La seva arquitectura és la següent:

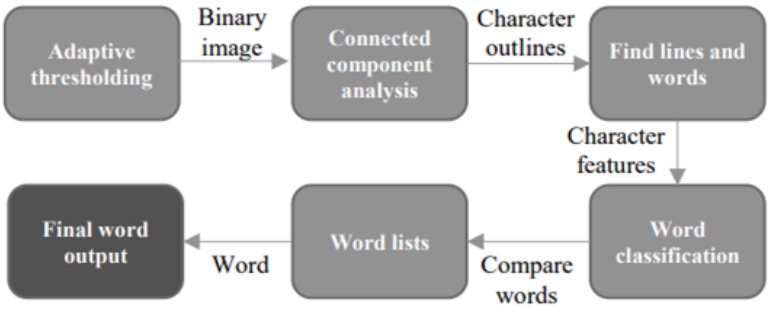


Figura 7: Arquitectura Tesseract OCR. [20]

Com es pot observar en la **Figura 7**, Tesseract OCR segueix una arquitectura semblant a la que s'explica anteriorment. Inicialment, s'aplica un preprocés propi que transforma la imatge a una binària. Llavors es divideixen els inputs en regions de text, seguidament en línies, paraules i finalment en caràcters. Un cop hi ha el text localitzat, s'ha de classificar, Tesseract ho fa amb dos passos. En un primer intent, el model classifica cada una de les paraules i les que tenen una satisfacció suficient es passen a un classificador adaptatiu per millorar-lo. Les paraules que no estan ben reconegudes, tornen a passar pel classificador. [20]

Paddle OCR:

El PaddleOCR [21] és un sistema desenvolupat per PaddlePaddle d'accés obert i fàcil d'utilitzar per a l'usuari. Utilitza dos tipus d'algorismes diferents, els de *two-stage* i els *end-to-end*. Generalment, els *two-stages*, es caracteritzen per dividir-se en dos algoritmes, la detecció de text i el reconeixement del text. Els *end-to-end*, combinen els dos per unir en un sol algoritme, entrena les dues parts conjuntament. En el cas de la detecció de text, PaddleOCR ofereix diversos algoritmes com EAST (*Efficient and Accurate Scene Text Detector*), DB (*Differentiable Binarization*), SAST (*Single-SHot Arbitrarily-Shaped*)… . Pel reconeixement de text utilitza les CRNN (*Convolutional Recurrent Neural Network*), SRN(*Semantic Reasoning Networks*)... . De tots els algoritmes mencionats anteriorment, les CRNN [22] són de les que més s'utilitzen en els OCR, es caracteritzen per combinar les CNN amb les RNN. Utilitzen les CNN per extreure les característiques i detectar patrons, contorns... de l'input i també les RNN (*Recurrent Neural Network*) que s'encarreguen de modelar les dependències seqüencials i mantenir informació passada. Això permet al model poder identificar millor el text en casos on hi ha canvis en els fons o la seqüència de text.

EasyOCR:

EasyOCR [23] és un model OCR de Python que es caracteritza per la facilitat que soposa utilitza-lo per a l'usuari i que conté models preentrenats en més de 80 idiomes. Té la següent estructura:

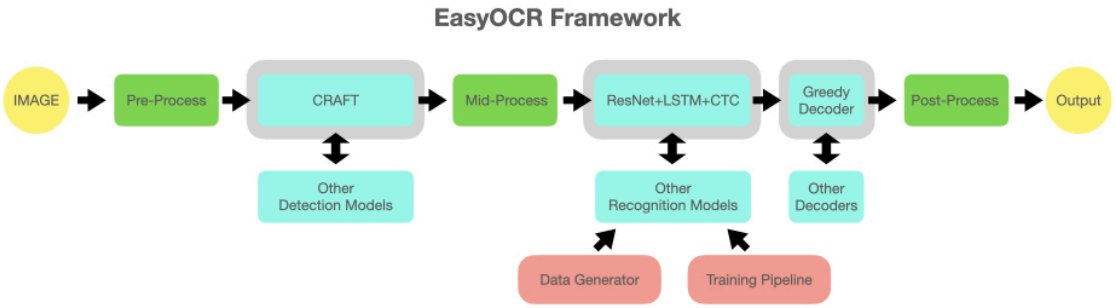


Figura 8: Arquitectura EasyOCR. [23]

Com mostra la figura anterior, després de fer el preprocessament, hi ha la pas de detectar el text. Actualment, EasyOCR utilitza CRAFT (Character-Region Awareness For Text detection), de *PyTorch*, que detecta les regions amb text de forma efectiva i utilitzant l'afinitat entre caràcters. A través de les CNNs, CRAFT genera una puntuació de regió (per localitzar caràcters individuals) i puntuació per afinitat (agrupar els caràcters en instàncies).

Un cop s'ha detectat el text, CRAFT utilitza 3 components. El primer és *feature extraction* que transforma l'input en valors numèrics redundants pel procés de classificació. En aquest cas s'utilitzen Resnet (*Residual Networks*) i VGG (*Visual Geometry Group*). Com a segon pas, hi ha *sequence labeling* que assigna una etiqueta a cada un dels elements de la seqüència utilitzant l'eficiència de LSTM (*Long Short-Term Memory*). Finalment, hi ha el procés de *decoding* a través de CTC (*Connectionist Temporal Classification)*.

Capítol 6

Marc pràctic detecció d'objectes

A la part pràctica de detecció d'objectes, s'aprofita l'arquitectura YOLOv8 per localitzar i classificar els logos. Per tal d'adaptar el model a les necessitats i condicions del projecte, s'ha d'entrenar el model amb imatges dels productes i logos que es vulguin considerar. Tot i que YOLOv8 proporciona models preentrenats, per obtenir resultats precisos, cal entrenar-lo amb un conjunt d'imatges personalitzat. Aquest procediment va des de la creació del *dataset*, anotació, entrenament i avaluació dels resultats.

6.1 Eines utilitzades

- YOLOv8, repositori *ultralytics*: Proporcionat per *ultralytics*, YOLOv8 [24] és de codi obert i s'hi pot accedir des del seu repositori Github. El codi permet entrenar i testejar models de detecció d'objectes.
- CVAT: Per fer l'anotació d'objectes en imatges, CVAT [25] permet penjar imatges, indicar els logos que apareixen i extreure els resultats en labels, format apte per YOLOv8.
- Repositori data_augmentation_yolov7: El repositori de Github "Data Augmentation on YOLO" [26], permet dividir el conjunt d'imatges i labels per entrenar, validar i testejar.
- Google Colab Pro: Amb la versió gratuïta de Google Colab [27] es poden executar codis amb uns recursos molt limitats. Google Colab Pro, es pot llogar el servei per un mes i proporciona unitats computacionals que es poden executar amb GPUs potents.

6.2 Procediment

Per la part pràctica de detecció d'objectes, s'utilitza YOLOv8 que és un model de codi obert. L'empresa *ultralytics* permet descarregar el repositori de YOLOv8 [24] i entrenar el model des de zero o utilitzar models preentrenats. Com que aquest projecte es centra en reconèixer marques que es puguin trobar a les escombraries en ciutats de Catalunya, hi ha marques locals que no són conegudes a la resta del món. Aquest fet dificulta trobar un *dataset* que compleixi els requisits i, per tant, s'ha optat per crear un conjunt d'imatges i entrenar un model de YOLOv8. A continuació, s'explica tot el procediment:

1. Creació del conjunt d'imatges:

Inicialment per crear el *dataset*, es fa una llista de marques d'envasos més comunes i que es poden trobar a les escombraries de les ciutats. D'aquesta llista inicial, se n'acaben seleccionant 22 (vegeu Apèndix-1).

Un cop seleccionades les marques amb les quals es vol treballar, es fa una recol·lecta de productes i es comencen a tirar fotos dels productes en diverses posicions, orientacions, il·luminacions i amb fons variats. Les fotografies s'han tirat enfocant els productes del terra des d'una altura similar a la que qualsevol usuari podria fer. Algunes imatges contenen únicament un producte, i altres en tenen més d'un. A la **Figura 9** es poden observar tres exemples d'imatges d'aquest *dataset* inicial.

A part de les imatges que s'han tirat per crear el conjunt, per les marques més internacionals com Coca Cola, Nestle, Pepsi que s'acostumen a trobar més per carrer, s'afegeixen algunes imatges dels logos trobats a diversos *datasets* online *LogoDet-3K* [34] i *Logos: BK KFC McDonald Starbucks Subway* [35]. Amb la unió dels dos conjunts, s'obtenen 137 imatges i 145 logos.

Figura 9: Exemple d'imatges del *dataset* inicial.

2. Anotació de les dades:

Un cop es tenen les imatges, s'utilitza l'eina CVAT [25] per anotar manualment els logos que apareixen a cada imatge. Aquest procediment consisteix en dibuixar un rectangle indicant en quatre coordenades la posició del logo i quina marca li pertany. Es repeteix el procés per cada imatge i cada logo. A la **Figura 10** s'observa un cas d'anotació, amb un producte de la marca *Milka*, es forma un rectangle que indica la posició del logo.

Un cop es tenen les imatges anotades a la interfície de CVAT, s'extreuen els resultats en format de YOLO, les etiquetes o també anomenades *labels*. Aquest format consisteix en un fitxer txt per cada una de les imatges i per cada logo hi ha una línia formada per 5 xifres. La primera representa el nombre associat a cada marca i les altres 4 són les coordenades dels logos.

Figura 10: Exemple anotació del logo de Milka.

3. Augment del *dataset*:

Amb l'actual *dataset* format per les 137 imatges i *labels* no és suficient per entrenar de forma satisfactòria un model, així doncs, es fa un augment del conjunt d'imatges. Per fer-ho s'utilitza el codi donat pel repositori de Github, data_augmentation_yolov7 [26] que per cada una de les imatges d'input, en genera més aplicant-li diverses tècniques com la saturació, translació, afegeix soroll, canvi de brillantor i contrast... (vegeu Apèndix-2). Un cop aplicat aquest augment, s'obtenen 1233 imatges i 1305 logos anotats. A la **Taula 1**, s'analitza la distribució del total de productes per cada una de les marques. Per exemple Nestle és la que té més instàncies amb 117 i Artiach la que menys amb 18.

Totes les imatges i les seves corresponents etiquetes estan pujades al Github, en un repositori del projecte (vegeu Apèndix-3).

Marca	Inicial	Augmentat		Inicial	Augmentat		Inicial	Augmentat
Grefusa	5	45	**McDonalds**	11	99	**Artiach**	2	18
Bezoya	4	36	**Lays**	4	36	**Oreo**	4	36
Font Vella	9	81	**Nestle**	13	117	**Chips Ahoy**	6	54
CocaCola	10	90	**Fritz ravitz**	11	99	**Milka**	3	27
7up	10	90	**Haribo**	10	90	**Bonpreu**	7	63
Pepsi	8	72	**Gallo**	4	36	**Colgate**	4	36
Estrella Damm	6	54	**ChupaChups**	6	54	**Pantene**	4	36
Alpro	4	36						

Taula 1: Nombre de productes per cada marca en el *dataset* inicial i augmentat.

4. Entrenament del model:

Un cop ja es té el *dataset* complet, es comencen a fer diversos entrenaments de YOLOv8. Per fer aquest pas s'utilitza el repositori de YOLOv8 *ultralytics* [24] i es fan un total de quatre entrenaments modificant dos paràmetres:

·Els *epochs* o etapes, és un hiperparàmetre que representa el nombre de vegades que el model recorre tot el conjunt de les imatges.

·El batch size és un altre hiperparàmetre que defineix el nombre de mostres que el model a de recórrer abans d'actualitzar els paràmetres interns.

El primer entrenament es duu a terme amb el conjunt d'imatges augmentat a 1233 imatges i 20 epochs. Tarda unes 14 hores aproximadament, ja que es va s'executa amb la CPU local.

Pels següents tres entrenaments al voler augmentar el nombre d'*epochs*, a part que triga molt, executant en local, surt un error relacionat amb la GPU i el CUDA. Per tant, com a alternativa es decideix provar-ho amb Google Colab. Com que la versió gratis té els recursos limitats i no permet executar tot l'entrenament per complet, es contracta la versió Google Colab Pro durant un mes per poder utilitzar 100 unitats computacionals de GPUs potents.

El segon entrenament es fa amb 50 *epochs* i una *batch size* de 32, 986 imatges per entrenar el model, 123 per la validació i 124 per el test. Triga uns 30 minuts aproximadament.

El tercer és amb el mateix conjunt d'imatges que el segon però amb 100 *epochs* i *batch size* de 64. També triga uns 30 minuts aproximadament.

Pels primers tres entrenaments s'entrena el model des de zero, en canvi, pel quart s'utilitza el model ja preentrenat, yolov8n.pt, que proporciona *ultralytics*.

A continuació, es mostren els resultats de cada entrenament.

6.3 Resultats

Les funcions d'entrenament de YOLOv8, generen els resultats de l'entrenament utilitzant les imatges de validació que permeten avaluar cada model amb mètriques com la precisió i la matriu de confusió. Per executar el codi de YOLOv8 es requereixen poques línies de codi (vegeu Apèndix-4).

<u>Primer entrenament, 20 epochs:</u>

Com a primer entrenament, s'utilitzen les 1.233 imatges i 20 *epochs*. Utilitzant les mateixes imatges per l'entrenament i per la validació, s'obtenen resultats molt millorables. Mentre que s'observen bastantes deteccions, hi ha molts logos que no detecta o que detecta erròniament.

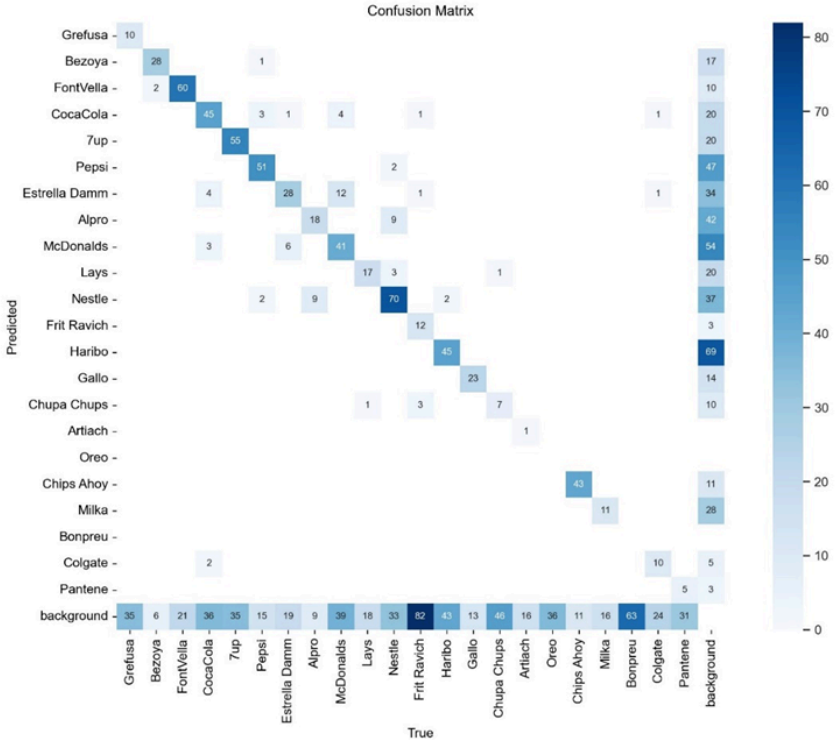


Figura 11: Matriu de confusió del primer entrenament.

A la **Figura 11** es pot observar que Fritz Ravich és una de les marques que menys reconeix l'algoritme. De tots els productes, només en reconeix 12 correctament, 82 no els identifica i 5 que els confon amb altres logos. Per altres marques com Nestle, Font Vella, 7up, els resultats són millors, ja que en detecta més del 50%.

<u>Segon entrenament, 50 epochs:</u>

Per fer aquest segon entrenament s'utilitzen 986 imatges per entrenar el model, 123 per la validació i 124 pel test. Els paràmetres utilitzats son els que venen per defecte en excepció del nombre de *epochs* que són 50 i el *batch size*, 32.

Observant el csv dels resultats, per l'últim epoch es veuen diverses mètriques d'avaluació, (vegeu Apèndix-5) on s'expliquen cada una de les mètriques.

train/box_loss	train/cls_loss	val/box_loss	val/cls_loss	precision	recall	mAP50	mAP50-95
1,331	1,5091	1,214	1,0866	0,77782	0,816	0,87235	0,6303

Taula 2: Mètriques de l'últim *epoch* del segon entrenament.

Com es pot observar a la **Taula 2** i la **Figura 12**, al llarg dels 50 *epochs*, les pèrdues de localització i classificació va disminuint a mesura que augmenta el nombre d'*epochs*. Això s'observa tant pel procés d'entrenament com en el de validació, i indica que el model va millorant quant a pèrdues i millora la predicció d'imatges no vistes. Tot i això, les pèrdues a l'últim *epoch*, superen els 1.3 d'error i deixa marge de millora.

Llavors en quant a la precisió i el recall veiem que van en augment com era d'esperar, a l'últim *epoch* tenen valors de 0.778 i 0.816 respectivament. Tot i ser lleugerament alts, podrien millorar. El mateix cas amb mAP50 i mAP50-95.

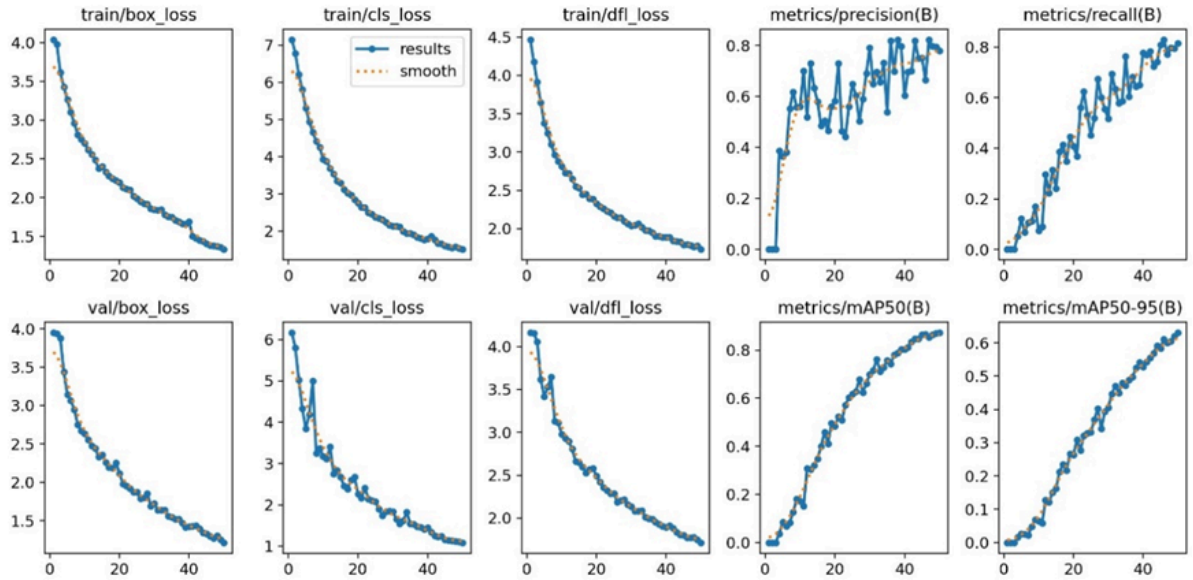


Figura 12: Evolució de les mètriques al llarg dels epochs del segon entrenament.

A partir de la confusion matrix de la **Figura 13** s'observa que hi ha unes marques com Chips Ahoy, Bezoya, 7up... que el model té 100% d'encert. En canvi altres marques com McDonalds, no es tan eficient, confon moltes imatges de McDonalds amb Estrella Damm. Tot i així, hi ha una millora quant a l'últim entrenament.

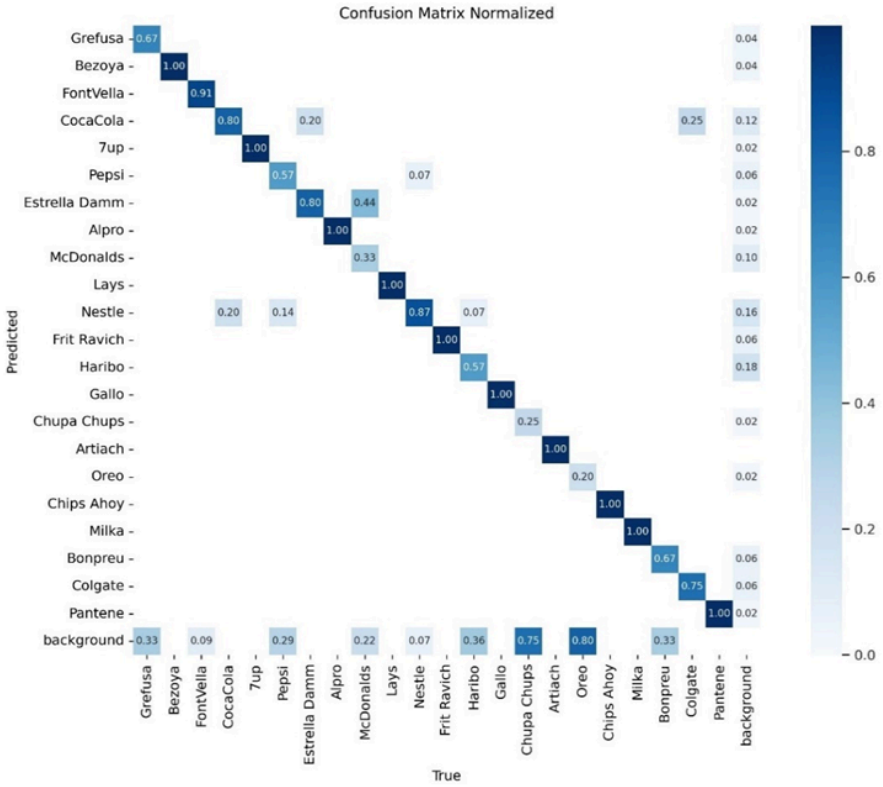


Figura 13: Matriu de confusió normalitzada segon entrenament.

A més dels resultats de l'entrenament i la validació, hi ha les imatges del conjunt de test que el model no ha vist. Amb el model entrenat, es fa un test amb les imatges i observem els resultats variats, algunes marques les detecta bé, algunes les confon amb d'altres i n'hi ha que no les detecta pràcticament mai.

Figura 14: Exemples de detecció correctes.

Figura 15: Exemples de deteccions errònies.

A la **Figura 14** s'observen dos casos on la detecció és correcte i a més amb una puntuació de confiança alta. Per altra banda, a la **Figura 15** s'observen dos casos de deteccions errònies, el primer cas detecta un logo equivocat i el segon no en detecta cap.

Per veure la resta de deteccions, el següent enllaç porta a una carpeta de *drive*:

https://drive.google.com/drive/folders/1r6ol94vlyZBSwSVEf7xJdJWAhTFPQVJK?usp=sharing

Tercer entrenament 100 epochs:

Per aquest entrenament, s'utilitza el mateix *dataset* que l'anterior però amb 100 *epochs* i un *batch size* de 64.

train/box_loss	train/cls_loss	val/box_loss	val/cls_loss	precision	recall	mAP50	mAP50-95
0,82613	0,7658	0,78685	0,63229	0,92783	0,91737	0,97083	0,80802

Taula 3: Mètriques de l'últim *epoch* del tercer entrenament.

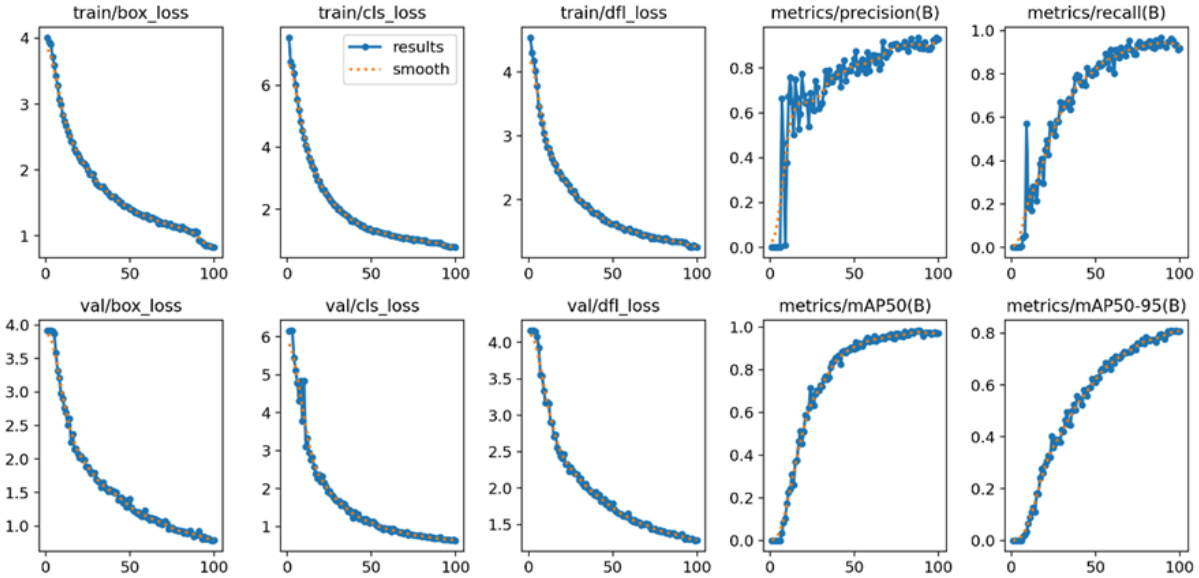


Figura 16: Evolució de les mètriques al llarg dels *epochs* del tercer entrenament.

Analitzant la **Taula 3** i la **Figura 16**, veiem que les pèrdues de localització i classificació van disminuint al llarg dels *epochs* i a l'últim, tenen un valor menor que 1, fet que no passava en el segon entrenament. L'alta precisió de 0.928, ens indica que del total d'objectes que detecta el model, la majoria els classifica en la marca correctament. El recall de 0.917, indica que el model també té bona capacitat de trobar els logos.

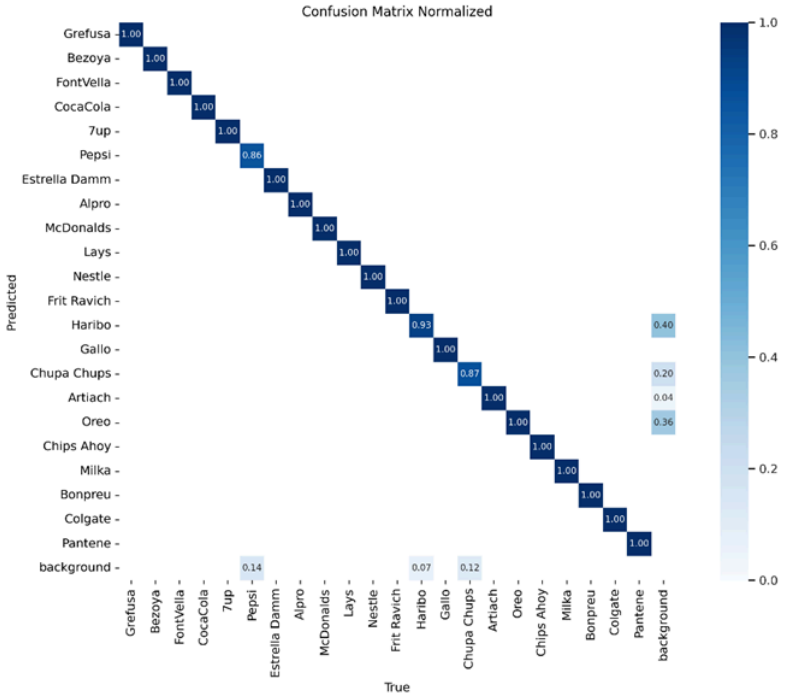


Figura 17: Matriu de confusió normalitzada tercer entrenament.

A través de la *confusion matrix,* **Figura 17**, s'observa que el model ha millorat molt respecte l'anterior entrenament, en la majoria de les marques, tenen un 100% de precisió.

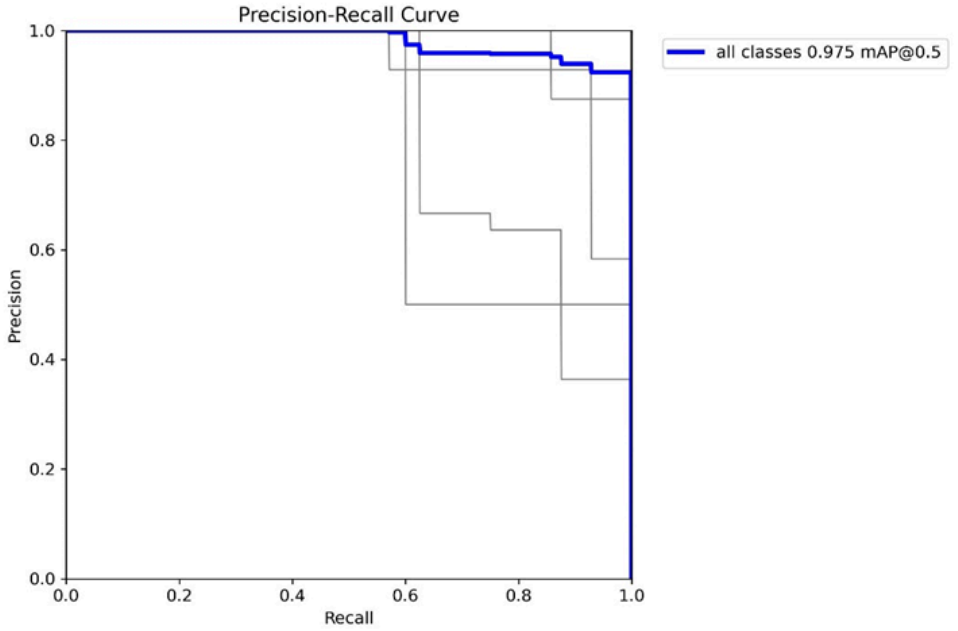


Figura 18: Corba de precisió i sensibilitat

Amb la corba de precisió i sensibilitat o també anomenat *recall*, **Figura 18**, observem bons resultats, ja que al augmentar el *recall*, la precisió tarda a disminuir i a més no decreix gaire.

Al provar el model amb les imatges del conjunt de test s'han detectat correctament els logos dels productes. Tot i això, en alguns casos detecta el mateix logo diverses vegades. Per veure les deteccions, el següent enllaç porta a una carpeta de *drive*:

https://drive.google.com/drive/folders/1SKLcerhnXtGIdhccuydT0BInKaTonHuv?usp=sharing

Quart entrenament, 100 *epochs* + model pre-entrenat:

Per aquest últim entrenament s'utilitzen els mateixos paràmetres que l'anterior però aquest cop partint d'un model ja preentrenat.

train/box_loss	train/cls_loss	val/box_loss	val/cls_loss	precision	recall	mAP50	mAP50-95
0,27	0,30402	0,27388	0,23606	0,984	0,99755	0,995	0,97307

Taula 4: Mètriques de l'últim *epoch* del quart entrenament.

Observant la **Taula 4**, s'observa que són els millors resultats obtinguts fins ara. Les pèrdues d'entrenament i validació són valors molt baixos, en canvi, les mètriques de precisió i sensibilitat són resultats pràcticament perfectes.

Comparant amb els entrenaments anteriors, la **Figura 19** mostra aquest cas a l'utilitzar ja un model preentreat, la precisió augmenta i convergeix molt més ràpid. Pel mateix motiu, les pèrdues disminueixen amb menys *epochs*.

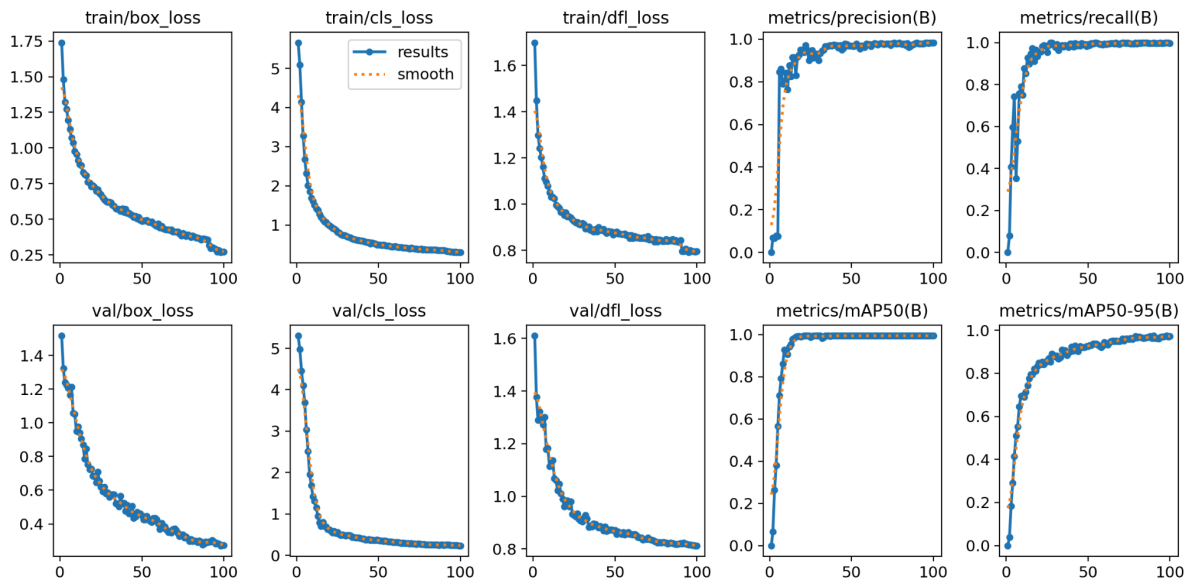


Figura 19: Evolució de les mètriques al llarg dels *epochs* del quart entrenament.

Observant la *confusion matrix*, **Figura 20**, veiem com amb el *dataset* de validació s'obté un 100% de precisió.

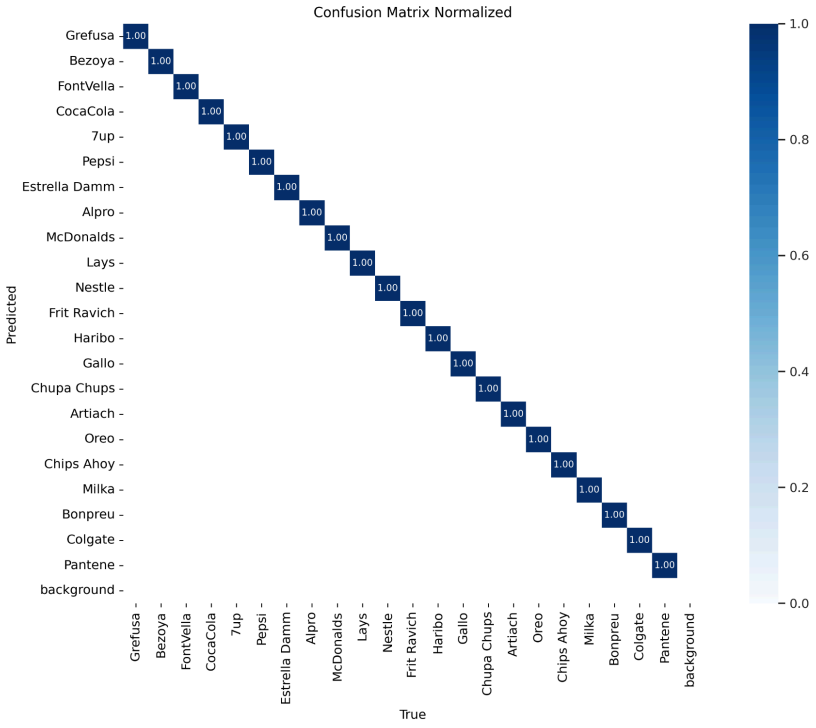


Figura 20: Matriu de confusió normalitzada quart entrenament.

Per tant, amb les imatges de validació, el model detecta tots els logos de les imatges, a la **Figura 21** es mostren exemples de les prediccions i les puntuacions de confiança.

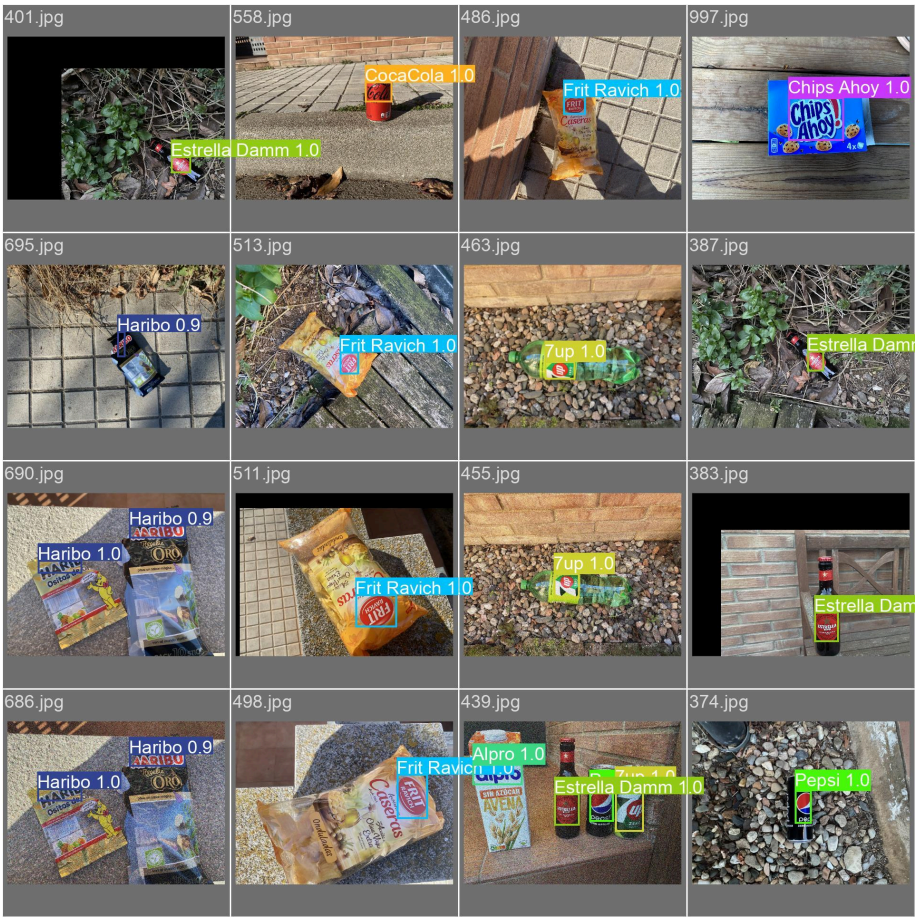


Figura 21: Exemples de les deteccions del quart entrenament.

En el següent enllaç s'observen les diferents deteccions de les dades de test del *dataset* passades amb el model entrenat.

https://drive.google.com/drive/folders/1sGwd9D4-v00dDgFcUOEy59adi2sagBmn?usp=sharing

6.4 Conclusions

El model que obté millors resultats, és el quart, que està entrenat amb 100 *epochs* i parteix d'un model ja preentrenat. Els resultats amb el conjunt de dades de validació, surt perfecte, detecta cada un dels logos i, per tant, obté una precisió i *recall* molt alts. Tot i això, cal destacar que les imatges de validació i les d'entrenament són força semblants, ja que han estat obtingudes amb un procés d'augment del *dataset*. Al capítol 8, es torna a avaluar el model amb un conjunt d'imatges completament nou i veurem que tot i que els resultats són bons, no són tan perfectes com els resultats d'aquest apartat. Com a avantatges, utilitzar un model de detecció d'objectes concretament YOLO, és l'alta precisió i rapidesa que proporciona. Tot i això, com a desavantatge, és que s'ha d'entrenar el model amb un conjunt ampli d'imatges per cada una de les marques que es volen reconèixer. I per tant, afegir una marca requereix tornar a entrenar el model.

Capítol 7

Marc pràctic reconeixement de text

Per la part pràctica de Text Recognition, s'utilitzen tres models d'OCR per extreure text de les imatges. Com que aquests models ja estan entrenats en diverses fonts, no s'ha d'entrenar cap OCR. Per dur a terme la detecció de marques, no només fa falta detectar el text que apareix a la imatge sinó netejar-lo i extreure'n les parts rellevants per poder identificar marques.

7.1 Eines utilitzades

- Tesseract OCR: Consisteix en un motor de reconeixement de caràcters. La llibreria de *Python pytesserract* [28], és de codi obert i permet executar el model de detecció de text Tesserract OCR gestionat per Google.
- PaddleOCR: El repositori de Github de paddlepaddle [21], permet baixar la llibreria *paddleocr* per tal d'utilitzar aquest altre mètode de reconeixement de text .
- EasyOCR: El repositori de Github de EasyOCR [23] i, permet utilitzar la llibreria *easyocr* per fer ús de les seves funcions i per tal de reconèixer text.
- Llibreria spaCy: És una llibreria de codi obert que proporciona eines de processament de llenguatge natural. En aquest projecte s'utilitza per processar el text en brut a extreure entitats. [29]
- NER Text Annotator: És una web que permet anotar text per tal d'entrenar models de NER (*Named Entity Recognition*). [30]
- Llibreria fuzzy wuzzy: És una llibreria de *Python* que permet comparar text, calcular la similitud entre *strings* i retornar un valor segons la distància de Levenshtein entre les dues seqüències. [31]

7.2 Procediment

1. Extreure text

Inicialment per extreure el text de cada una de les imatges, es proven tres models diferents, Tesserract OCR, Paddle OCR i Easy OCR. Tots tres es poden executar utilitzant llibreries de *Python*. Per veure quin funciona millor en les imatges d'escombraries, s'utilitza el conjunt de les 137 imatges del *dataset* inicial. En els tres models, es crea un codi que per cada una de les imatges, genera com a output, un fitxer txt amb el text detectat.

Primera prova: Tesseract OCR

En un primer intent per reconèixer els logos a través de Text Recognition, s'utilitza Tesseract OCR a través la llibreria de pytesseract [28].

Primerament, observant les imatges, els resultats semblen bastant dolents, al tenir logos amb noms amb fonts diverses, el model no és capaç de reconèixer gaires logos. En la majoria d'imatges reconeix text però no de logo, sinó d'altres textos de l'envàs, per exemple, com s'observa a la **Figura 22**, Tesseract OCR detecta vàries paraules, però del logo només detecta "pro".

Figura 22: Exemple de detecció Tesseract OCR.

Segona prova: PaddleOCR

En una segona prova, s'utilitza PaddleOCR que proporciona models preentrenats per diversos idiomes. Descarregant el repositori de *paddleocr*, s'aconsegueixen resultats molt millors que Tesserract OCR.

Figura 23: Exemple 1 de detecció PaddleOCR.

Figura 24: Exemple 2 de detecció PaddleOCR.

Com mostra la **Figura 23**, hi ha casos que el nom del logo es detecta perfectament i és l'únic text que detecta. En altres casos com la **Figura 24,** detecta més text i a més, s'observa com registre la paraula "Haribo" dues vegades, ja que apareix vàries vegades a l'envàs.

Tercera prova: EasyOCR

Per últim s'utilitza EasyOCR amb models ja entrenats per fer una última prova. En aquest cas els resultats són millors que Tesserract, però no tan bons com PaddleOCR.

A la **Figura 25** es mostra un exemple en què el model detecta pràcticament tots els textos inclòs el logo.

Figura 25: Exemple de detecció EasyOCR.

2. Netejar el text i extreure'n marques

Un cop es tenen els fitxers txt per cada una de les imatges, cal netejar el text per extreure'n només les parts rellevants pel projecte, les marques. Per fer aquest pas, es fa ús la llibreria *spaCy* per detectar possibles entitats a partir de tot el text extret. Llavors comparar-les amb la llista de marques del *dataset,* utilitzant la llibreria *fuzzy wuzzy*.

2.1. Entrenar spaCy

La llibreria spaCy [29], proporciona models NER (*Named Entity Recognition*) per reconèixer entitats en text no estructurat. Ja hi ha alguns models entrenats en diferents idiomes, en el cas d'aquest projecte, les marques que es tenen en compte, són de diferents països i no són solament en un idioma. Utilitzant el model en anglès, no detecta marques com Grefusa, Font Vella... i utilitzant el de castellà, no reconeix altres com Fritz Ravich. Per això, es decideix entrenar un model propi únicament amb les marques del *dataset*. Per entrenar el model, s'agafen un conjunt de 28 fitxers txt obtinguts al processar les imatges amb el mètode PaddleOCR, ja que és el que dóna millors resultats. Amb aquest conjunt de text en brut, es fan les anotacions, utilitzant el repositori de GitHub, ner-annotator [30] on per tot el text, s'ha d'indicar quines paraules o conjunt de paraules, han de ser reconegudes com a marca. L'output d'aquesta eina, és un fitxer JSON, que llavors s'utilitza en un script de *Python* i s'entrena el model spaCy.

2.2. Comparar marques

Un cop es té el model de spaCy entrenat, es redacta un codi de *Python* per tal de, a partir de tot el text reconegut, processar-lo i extreure'n únicament el nom de les marques. El codi complet està al Github (vegeu Apèndix-3) però la idea general és que primer de tot, passa el text a la funció *nlp* de Natural Processing Language i es queda únicament amb la llista d'entitats. Llavors utilitza la llibreria *fuzzy wuzzy* [31] per comparar cada entitat amb la llista de les marques que s'han establert. *Fuzzy wuzzy* permet comparar *strings* amb una funció que retorna un valor de la similitud entre dos textos. El codi l'utilitza, per juntament amb un threshold determinat, decidir si considerar l'entitat com a marca o no. L'output d'aquest codi és un fitxer txt amb les marques reconegudes per cada imatge. Seguidament, amb un altre codi, es comparen els fitxers predits pel model de reconeixement de text, amb el *groundtruth*. El codi es pot trobar al Github (vegeu Apèndix-3), el que fa, és calcular els *True Positives*, *False Positives* i *False Negatives* per llavors trobar la precisió, el recall i el valor F1.

7.3 Resultats

		spaCy	trained Spacy
TesseractOCR	True Positives	5	28
	False Positives	130	458
	False Negatives	140	117
	Precision	0.037	0.058
	Recall	0.344	0.193
	F1-score	0.036	0.089
EasyOCR	True Positives	14	31
	False Positives	4	12
	False Negatives	131	114
	Precision	0.778	0.721
	Recall	0.096	0.214
	F1-score	0.171	0.330
PaddleOCR	True Positives	41	92
	False Positives	0	18
	False Negatives	104	53
	Precision	1.0	0.836
	Recall	0.283	0.635
	F1-score	0.441	0.712

Taula 5: Taula de resultats pels diferents models provats.

De la **Taula 5** veiem els resultats per cada un dels algoritmes provats. En la primera columna es mostren els resultats utilitzant el model preentrenat de spaCy en castellà, la segona conté els resultats del model spaCy entrenat amb les marques. Per cada un dels mètodes, el canvi de model de spaCy té efectes diferents, pel cas de Tesseract OCR, utilitzant el model entrenat, s'obté més precisió, però a la vegada, redueix el *recall*. En el cas de EasyOCR i PaddleOCR, passa el contrari, la precisió disminueix, però el *recall* augmenta. Tot i això, mirant el valor F1, sempre dona valors més alts que amb el model entranat. De tots el sis casos, el que s'utilitzarà per els següents pasos, és el Paddle OCR i el model spaCy entrenat ja que és el que mostra un rendiment més alt, encara que la precisió no sigui la millor, el balanç amb el *recall* i la puntuació F1, donen els resultats més complets.

7.4 Conclusions

Un cop es tenen els tres models avaluats, veiem per les condicions d'aquest projecte, on el reconeixement de text es fa a productes escampats per terra amb fonts de diferents colors, mides, formats… , el model que dona millors resultats és el PaddleOCR. Analitzant la precisió, veiem que en els casos que no és 1.0, és a causa dels *False Positives*, és a dir, marques que s'han detectat quan no hi són. Això passa sobretot en els casos on l'envàs dels productes, conté el nom de la marca vàries vegades i, per tant, es suma com si fos un altre producte.

El reconeixement de text, a diferència de la detecció d'objectes, té l'avantatge que no s'ha d'entrenar el model per cada una de les marques. Encara que si s'utilitza spaCy, es pot entrenar per extreure les entitats, no és tan costós com l'entrenament d'imatges. Tot i això, té el desavantatge de que només pot reconèixer marques on el seu nom aparegui al Logo, o en algun lloc de l'envàs, això limita el model en alguns casos.

Capítol 8

Sistema final

Arribats aquest punt, s'han avaluat dos mètodes per identificar les marques de productes que apareixen a una imatge. Per un costat s'ha desenvolupat un model de YOLOv8 entrenat per reconèixer 22 marques a partir dels seus logos. Per altra banda, s'ha analitzat l'algoritme PaddleOCR que detecta tot el text i amb uns scripts, extreu les marques que apareixen. Com ja s'ha comentat, cada un té uns avantatges i desavantatges i hi ha casos que un dels models detecta una marca que no detecta l'altre i, per tant, ajuntant els dos models de manera que es cobreixin les carències, pot resultar en un sistema més complet i amb millors resultats. A continuació s'explica el procés i resultats del sistema que uneix els dos mètodes.

8.1 Creació d'un nou conjunt d'imatges

El conjunt d'imatges amb el que s'ha treballat fins ara, s'ha utilitzat moltes vegades per analitzar cada mètode i a més, com ja s'ha comentat a l'apartat de detecció d'objectes, al passar per un procés d'augment del conjunt, hi ha imatges que són bastant semblants i, per tant, pot donar una falsa impressió de resultats perfectes. Per això, s'ha creat un altre conjunt d'imatges, per tal d'avaluar els dos mètodes i el nou sistema amb imatges plenament noves.
El procés de creació del *dataset* és el mateix que el que està explicat anteriorment. Està format per 51 imatges i 95 logos anotats i les seves corresponents *labels*. Les imatges contenen productes de les marques ja vistes i busquen emmarcar productes amb diferents fons, il·luminacions i orientacions. Aquest conjunt d'imatges extra està penjat al Github del projecte (vegeu Apèndix-3).

Marca		Marca	
Font Vella	6	Fritz ravitz	7
CocaCola	9	Haribo	10
7up	15	Gallo	4
Pepsi	10	Artiach	2
Estrella Damm	6	Oreo	4
Alpro	3	Chips Ahoy	2
McDonalds	5	Milka	4
Lays	5	Bonpreu	3
Nestle	0	**Total**	**95**

Taula 6: Nombre de productes per cada marca en el *dataset* extra.

8.2 Redacció codi del nou sistema

Per unificar els dos mètodes, detecció d'objectes i reconeixement de text, es redacta un *script* de *Python* que es pot trobar a Github (vegeu Apèndix-3) i que ajunta el model entrenat de YOLOv8 i l'algoritme PaddleOCR. Aquest codi utilitza la funció find_brands() que s'ha redactat per l'apartat del reconeixement de text i que de text no estructurat, n'extreu les marques. Primer de tot, processa la imatge amb PaddleOCR i obté una llista de les marques que ha reconegut, seguidament fa el mateix però amb YOLOv8. Llavors, a partir de les dues llistes, se'n crea una de nova que conté la intersecció de les dues llistes més les diferències entre les dues llistes. És a dir, agafa les marques detectades pels dos models i les que només un dels dos detecta, la **Figura 26** mostra un exemple. Aquest mètode busca reduir el nombre de *False Negatives*, les marques que no són detectades.

El resultat és extret en fitxers txt que permet calcular les mètriques resultants amb el mateix codi que la part de reconeixement de text.

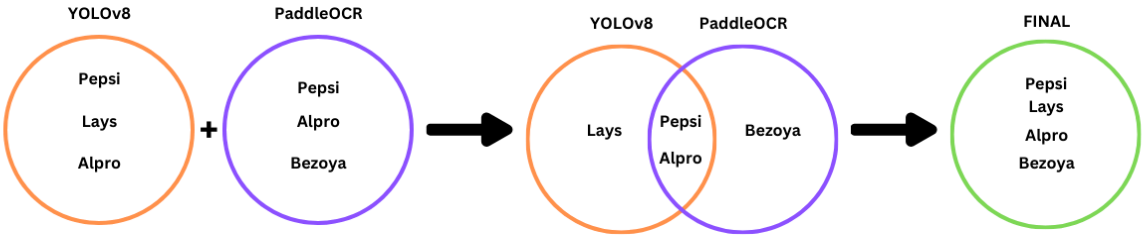


Figura 26: Exemple de combinació dels dos mètodes per obtenir el resultat final.

8.3 Resultats

Un cop fet el codi, s'avaluen els tres mètodes, únicament utilitzant YOLOv8, únicament utilitzant PaddleOCR i el sistema que combina els dos. En aquest cas només es centra en que l'algorisme sigui de capaç d'identificar les marques que apareixen a la imatge, i no té en compte la precisió quant a la localització i les *bounding boxes*.

	Només YOLOv8	Només PaddleOCR	Combinació dels dos
True Positives	45	39	59
False Positives	3	9	12
False Negatives	50	56	36
Precision	0.938	0.813	0.831
Recall	0.474	0.411	0.621
F1-score	0.629	0.546	0.711

Taula 7: Mètriques per cada un dels mètodes estudiats.

Analitzant la **Taula 7**, veiem com l'algorisme que combina els dos mètodes, obté més nombre de *True Positives*, és a dir, identifica més marques correctament. També obté més *False Positives*, ja que al final acaba sumant les dels dos casos, però a la vegada redueix el nombre de *False Negatives*, marques que no identifica però hi són.
Pel que fa la precisió, YOLOv8 és el que en té més amb un 0.938, però al mateix moment, té un *recall* baix. Si mirem la combinació dels dos, no s'aconsegueix una precisió tan alta però si que tant la F1-score com el *recall* són valors severament més alts.
Per tant, dels tres mètodes, pel cas d'aquest projecte, es podria dir que el que dona millors resultats, és la combinació de detecció d'objectes i el reconeixement de text.

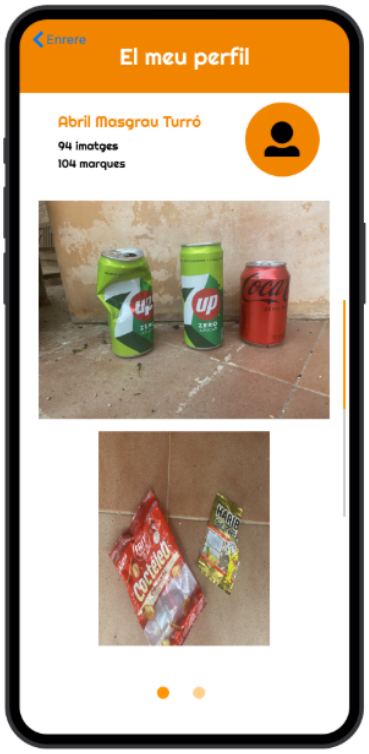
Capítol 9

Prototip d'aplicació per registrar la distribució d'escombraries i marques

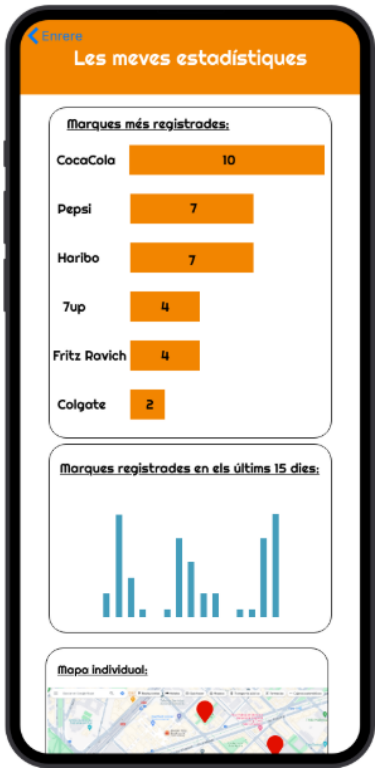
Un cop es té el sistema de reconeixement dels logos, s'ha fet un prototip de l'aplicació resultant que consisteix en disseny de la interfície. Aquest prototip s'ha fet amb l'eina de proto.io [32] que permet crear un prototip amb diverses pantalles i a més té l'opció de mostrar les interaccions amb botons, i altres elements d'una forma poc complexa.
El nom de l'aplicació és Brand Waste Tracker i consta de diverses pantalles amb diferents funcionalitats. La idea general de l'aplicació, és que cada usuari tingui un perfil, on afegeixi imatges i la mateixa aplicació, reconegui quines marques hi apareixen. A mesura que els usuaris vagin generant fotos i dades, l'aplicació anirà recollint la informació per fer unes estadístiques. Hi ha estadístiques per cada usuari, per cada marca i ciutat que implementi aquesta aplicació. A continuació es mostren les diverses pantalles de l'aplicació.

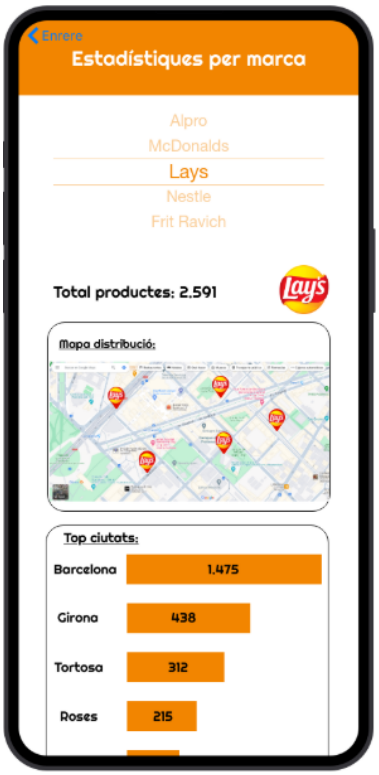
La primera pantalla mostra l'inici de l'aplicació, amb el seu nom de *Brand Waste Tracker,* i el logo de l'app. La segona mostra un menú que permet a l'usuari anar a les diferents pantalles i a la configuració, primer de tot, hi ha el botó per anar al perfil de l'usuari, llavors el mapa amb les marques registrades i els tres següents són estadístiques on l'usuari es pot informar de les dades que recull l'aplicació. S'ha gravat un vídeo que mostra amb més detall l'aplicació i també s'aprecien les interaccions amb els botons, més semblant a una aplicació real (vegeu Apèndix-6).

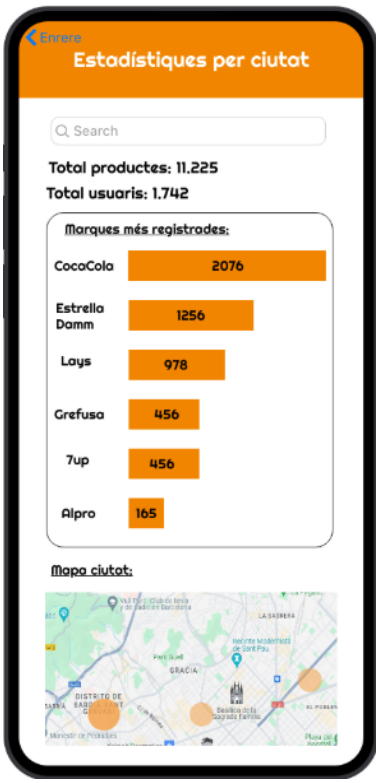
A continuació, es mostren dues pantalles més, a l'esquerra hi ha el perfil de l'usuari. Indica el total d'imatges pujades, i el nombre de marques que ha identificat l'aplicació, a més, es mostren les imatges de cada usuari i dona l'opció de veure la imatge amb les *bouding boxes* per cada logo detectat (vegeu Apèndix-6). A la dreta, hi ha el mapa on amb la ubicació de l'usuari, mostra les marques i les localitats pròximes que han estat registrades per qualsevol usuari. A més, hi ha un buscador que permet buscar altres localitats d'una forma similar al Google Maps.



En les tres pròximes captures, es mostren estadístiques, un resum analític de les dades recollides de tots els usuaris. La primera mostra les estadístiques per cada usuari, com per exemple les marques més habituals, un registre de l'activitat dels darrers dies, un mapa individual, les localitats on l'usuari ha estat i un top d'usuaris més actius. Seguidament, hi ha una pantalla amb les estadístiques de cada marca. L'usuari pot seleccionar-ne una de la llista de marques que cobreix l'aplicació, i veure la distribució de registres en un mapa i també un top de les ciutats amb més productes. Finalment, hi ha una pàgina amb un resum per cada ciutat, que mostra el total de productes i d'usuaris, les marques més freqüents en aquell lloc i també un mapa amb els llocs on més rebuig hi ha.







Capítol 10

Conclusions i treball futur

10.1 Conclusions

En aquest projecte s'ha analitzat quin és el millor mètode per reconèixer logos en imatges de productes d'escombraries. Principalment, s'han valorat dos mètodes, detecció d'objectes i reconeixement de text, on per cada un s'han observat avantatges i inconvenients.
Pel reconeixement de text, s'ha conclós que el millor algoritme pel projecte, és PaddleOCR amb una precisió i sensibilitat bastant millorable. L'avantatge d'utilitzar el reconeixement de text, és que no s'ha d'entrenar el model per cada una de les marques, simplement requereix un bon sistema per netejar tot el text i extreure'n només els noms de les marques. Això fa que sigui menys costos afegir noves marques, però per altra banda, té la limitació que han de ser marques, on en els seus logos aparegui el nom. També s'ha observat que el reconeixement de text genera bastants de *False Positives*, ja que hi ha envasos que contenen el nom de la marca vàries vegades.
Pel que fa a la detecció d'objectes, utilitzant YOLOv8, els resultats són millors tant en la precisió com la sensibilitat. Al mateix moment, utilitzar aquests mètodes, requereixen un entrenament del model bastant complex, amb moltes imatges i temps d'entrenament, per tant, afegir marques noves, no és senzill.
Com ja s'ha analitzat a l'apartat 8.3, utilitzant la combinació dels dos mètodes, s'aconsegueix compensar els *False Negatives* de cada mètode i encara que la precisió no millora la de YOLOv8, tant la sensibilitat com la F1-score, surten valors més alts i, per tant, es conclou que és el millor sistema per utilitzar a l'aplicació.
Amb la creació de l'aplicació *Brand Waste Tracker* i amb la col·laboració de tots els usuaris, es pot proporcionar informació per tal de no només concienciar a la població de les restes d'escombraries a llocs públics, però també a les empreses responsables de les deixalles.

Pel que fa a la valoració del treball, ha sigut molt útil per aprendre les diferents branques del reconeixement d'imatges. Sobre la detecció d'objectes, informar-me dels algorismes actuals i aprendre el procés d'entrenar un model, des de la recopilació d'imatges, anotar-les, entrenar, avaluar-ho… . Quant a reconeixement de text, ha sigut molt útil comparar els mètodes, escriure un codi per treure la part necessària, utilitzar la llibreria spaCy… . Apart de tot l'aprenentatge de la part pràctica, ha sigut molt enriquidor aprendre els conceptes teòrics que envolten el reconeixement d'imatges i veure la magnitud de les xarxes neuronals convolucionals.

10.2 Treballs futurs

Un cop finalitzat el projecte, es poden identificar millores o extensions que es poden fer per tal de perfeccionar-lo.

- Expandir model YOLOv8: Entrenar el model amb més marques. En aquest projecte, es treballa amb 22 marques, una proposta de millora, podria ser augmentar el nombre.
- Solucionar els *False Positives* de reconeixement de text: Com ja s'ha comentat en els apartats anteriors, hi ha productes que contenen el nom de la marca vàries vegades i, per tant, es detecta com a diferents productes. Una millora seria trobar la manera d'identificar aquests casos i comptar-los com a un únic producte.
- Plantejar el problema de si a la imatge apareixen logos que no són d'escombraries: És a dir, si a la imatge surt de fons un local o un anunci de McDonalds, el model podrà detectar el logo, però no forma part del conjunt de deixalles.
- Desenvolupar l'app: En l'extensió d'aquest projecte, l'aplicació s'ha quedat a nivell de prototip, una continuació del projecte, podria ser fer-la funcional.

Bibliografia

[1] Greenpeace. (s.d.). *¿Cómo llega el plástico a los océanos y qué sucede entonces?* Recuperat 20 de febrer del 2024, des de https://es.greenpeace.org/es/trabajamos-en/consumismo/plasticos/como-llega-el-plastico-a-los-oceanos-y-que-sucede-entonces/

[2] Oi, B. M. (2023). *PepsiCo sued by New York state for plastic pollution*. BBC News. Recuperat 20 de febrer del 2024, des de https://www.bbc.com/news/business-67435329

[3] Orti, O., Tous, R., Gomez, M., Poveda, J., Cruz, L., & Wust, O. (2019). Real-Time Logo Detection in Brand-Related Social Media Images. In I. Rojas, G. Joya, & A. Catala (Eds.), *Advances in Computational Intelligence. IWANN 2019.* Lecture Notes in Computer Science (vol. 11507). Springer, Cham. Recuperat de https://doi.org/10.1007/978-3-030-20518-8_11

[4] Wang, J., Min, W., Hou, S., Ma, S., Zheng, Y., & Jiang, S. (2022). Logodet-3k: A large-scale image dataset for logo detection. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, *18*(1), 1-19. Recuperat de https://doi.org/10.48550/arXiv.2008.05359

[5] clickworker. (2023). *Image Recognition - Term Explanation in the AI Glossary*. clickworker.com. Recuperat 21 de febrer del 2024, des de https://www.clickworker.com/ai-glossary/image-recognition/

[6] O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*. Recuperat de https://doi.org/10.48550/arXiv.1511.08458

[7] GeeksforGeeks. (2024). *Introduction to Convolution Neural Network*. Recuperat 27 de febrer del 2024 des de https://www.geeksforgeeks.org/introduction-convolution-neural-network/

[8] IBM (2024) *What are Convolutional Neural Networks?*. Recuperat 27 de febrer del 2024 des de https://www.ibm.com/topics/convolutional-neural-networks

[9] Yani, M., S, S. M. B. I., & ST, M. C. S. (2019). Application of Transfer Learning Using Convolutional Neural Network Method for Early Detection of Terry's Nail. *Journal Of Physics. Conference Series*, *1201*(1), 012052. Recuperat de https://doi.org/10.1088/1742-6596/1201/1/012052

[10] MATLAB & Simulink (s.d.). *What is object detection?* Recuperat 28 de febrer del 2024 des de https://es.mathworks.com/discovery/object-detection.html

[11] Boesch, G. (2024). *Object Detection in 2024: The Definitive Guide*. viso.ai. Recuperat 28 de febrer de 2024 des de https://viso.ai/deep-learning/object-detection/

[12] Park, S. (2021). A guide to Two-stage Object Detection: R-CNN, FPN, Mask R-CNN. Medium. Recuperat 28 de febrer de 2024 des de https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c

[13] Kim, E. (2023). Understanding Single Shot Detector (SSD) - Elven Kim - *Medium*. Recuperat el 28 de febrer del 2024 des de https://medium.com/@elvenkim1/understanding-single-shot-detector-ssd-782e4b93aec3

[14] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788). Recuperat de https://doi.org/10.48550/arXiv.1506.02640

[15] Kundu, R. (2024b, abril 10). YOLO: Algorithm for Object Detection Explained [+Examples]. *V7*. Recuperat 4 de febrer de 2024 des de https://www.v7labs.com/blog/yolo-object-detection

[16] Keita, Z. (2022). *YOLO Object Detection Explained*. *Datacamp*. Recuperal el 4 de febrer del 2024 des de https://www.datacamp.com/blog/yolo-object-detection-explained

[17] Amazon Web Services, Inc. *¿Qué es el OCR? - Explicación del reconocimiento óptico de caracteres*. Recuperat el 25 de febrer del 2024 des de https://aws.amazon.com/es/what-is/ocr/

[18] Reddy, S. (2021). *What is an OCR ?? - Towards Data Science*. *Medium*. Recuperat el 25 de febrer del 2024 des de https://towardsdatascience.com/what-is-ocr-7d46dc419eb9

[19] Smith, R. (2007, September). An overview of the Tesseract OCR engine. In *Ninth international conference on document analysis and recognition (ICDAR 2007)* (Vol. 2, pp. 629-633). IEEE. Recperat de 10.1109/ICDAR.2007.4376991

[20] Gjoreski, M., Zajkovski, G., Bogatinov, A., Madjarov, G., Gjorgjevikj, D., & Gjoreski, H. (2014). Optical character recognition applied on receipts printed in Macedonian Language. Recuperat de 10.13140/2.1.1632.4489

[21] PaddlePaddle. (s.d.). *PaddleOCR/README_en.md at main · PaddlePaddle/PaddleOCR*. GitHub. https://github.com/PaddlePaddle/PaddleOCR/blob/main/README_en.md

[22] Gill, N. S. (2023). *Convolutional recurrent neural network for text recognition*. *XenonStack*. Recuperat el 26 de febrer del 2024 des de https://www.xenonstack.com/insights/crnn-for-text-recognition

[23] JaidedAI. (s. d.). *GitHub - JaidedAI/EasyOCR: Ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic and etc.* GitHub. https://github.com/JaidedAI/EasyOCR

[24] Ultralytics. (s. f.). *GitHub - ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite*. GitHub. https://github.com/ultralytics/ultralytics

[25] *CVAT*. (s. d.). Recuperat el 20 de gener del 2024 des de https://www.cvat.ai/

[26] MinoruHenrique. (s. f.). *GitHub - MinoruHenrique/data_augmentation_yolov7: Apply data augmentation techniques on YOLO v7 format dataset.* GitHub. Recuperat el 27 de gener del 2024 des de https://github.com/MinoruHenrique/data_augmentation_yolov7

[27] Google Colab. (s. d.). *Colab.google*. colab.google. Recuperat el 20 de febrer del 2024 des de https://colab.google/

[28] Tesseract-Ocr. (s. d.). *GitHub - tesseract-ocr/tesseract: Tesseract Open Source OCR Engine (main repository)*. GitHub. https://github.com/tesseract-ocr/tesseract

[29] *spaCy · Industrial-strength Natural Language Processing in Python*. (s. d.). Recuperat el 5 de març del 2024 des de https://spacy.io/

[30] *NER Annotator for SpaCy*. (s. d.). Recuperat el 5 de març del 2024 des de https://tecoholic.github.io/ner-annotator/

[31] *fuzzywuzzy*. (2020). Recuperat el 5 de març del 2024 des de PyPI. https://pypi.org/project/fuzzywuzzy/

[32] Proto.io. (s. d.). *Proto.io - Prototyping for all*. Recuperat el 4 d'abril del 2024 des de https://proto.io/

[33] White, G., Cabrera, C., Palade, A., Li, F., & Clarke, S. (2020). WasteNet: Waste classification at the edge for smart bins. *arXiv preprint arXiv:2006.05873*. Recuperat de https://doi.org/10.48550/arXiv.2006.05873

[34] *LogoDET-3K*. (2020). Kaggle. Recuperat el 20 de gener del 2024 des de https://www.kaggle.com/datasets/lyly99/logodet3k

[35] *Logos: BK KFC McDonald Starbucks Subway None*. (2019). Kaggle. Recuperat el 20 de gener del 2024 des de https://www.kaggle.com/datasets/kmkarakaya/logos-bk-kfc-mcdonald-starbucks-subway-none

[36] K. Rezkiani, I. Nurtanio and Syafaruddin, "Logo Detection Using You Only Look Once (YOLO) Method," *2022 2nd International Conference on Electronic and Electrical Engineering and Intelligent System (ICE3IS)*, Yogyakarta, Indonesia, 2022, pp. 29-33, Recuperat de 10.1109/ICE3IS56585.2022.10010121

Apèndix

Apèndix 1: Llista de marques del conjunt d'imatges

Num	Marca
0	Grefusa
1	Bezoya
2	FontVella
3	CocaCola
4	7up
5	Pepsi
6	Estrella Damm
7	Alpro
8	McDonalds
9	Lays
10	Nestle
11	Frit Ravich
12	Haribo
13	Gallo
14	Chupa Chups
15	Artiach
16	Oreo
17	Chips Ahoy
18	Milka
19	Bonpreu
20	Colgate
21	Pantene

Apèndix 2: Imatges del conjunt augmentat

A continuació es mostren alguns exemples d'imatges originals, amb les imatges transformades a través d'aplicar, una translació, canvi de saturació, contrast, brillantor, afegir soroll….

Imatge original i imatge amb soroll afegit:

Imatge original i imatge amb augmentar la saturació:

Imatge original i imatge amb més contrast:

Apèndix 3: Github projecte

Per guardar i mostrar tant els *datasets* utilitzats pel projecte com els codis, s'ha creat un repositori del Github. Conté:

- Carpeta "images": Imatges del *dataset* utilitzat per entrenar el model YOLOv8.
- Carpeta "labels": Etiquetes del *dataset* utilitzat per entrenar el model YOLOv8.
- extract_brands_spacy.py: Codi que a partir del text reconegut, el neteja i extreu únicament el nom de les marques.
- metrics.py: Codi que avalua els resultats predits pels mètodes de reconeixement de text en imatges. Calcula la precisió, sensibilitat i el F1-score.
- text_and_object_detection.py: Codi que s'utilitza per a combinar la detecció d'objectes amb YOLOv8 i el reconeixement de text amb PaddleOCR.
- Carpeta "Extra": Conté tant les imatges com les etiquetes del conjunt d'imatges creat per avaluar el model final.

https://github.com/AbrilMasgrau/DataLogos01

Apèndix 4: Entrenament amb YOLOv8

L'empresa *ultralytics*, permet descarregar el repositori de YOLOv8 i entrenar un model a través un conjunt d'imatges personalitzat utilitzant models ja preentrenats o des de 0. Per dur a terme aquest projecte, he entrenat un model des de 0 i també a partir d'un model preentrenat utilitzant el *dataset* personalitzat. El procés d'entrenament d'un model amb YOLO és bastant simple quant a codi i passos, a través de Google Colab, només es necessiten quatre línies de codi:

¡pip install ultralytics

import ultralytics from YOLO

model = YOLO("yolov8n.yaml") #model = YOLO("yolov8n.pt") pel model preentrenat.

results = model.train(data=os.path.join(ROOT_DIR, "file.yaml"), epochs=50, batch=32)

On file.yaml és un fitxer on s'especifiquen els *paths* a les imatges i *labels* d'entrenament i les de validació. A més s'especifiquen el nombre de classes i s'assigna a cada una de les marques, un número del 0 al 21. Queda de la següent manera:

path: /content/gdrive/My Drive/TFG/YOLO

train: train/images

val: val/images

nc: 22

names:

0: Grefusa

...

21: Pantene

Apèndix 5: Mètriques d'avaluació

Un cop entrenat el model, es crea una carpeta on es guarden els resultats de l'entrenament i la validació. És important saber a què es refereixen les mesures d'avaluació que proporciona YOLOv8.

·**box_loss**: cuantifica la diferència entre les *bounding box* que ha predit el model, amb les reals.

·**cls_loss**: quantifica la diferència entre les *labels*, etiquetes, predites i les reals.

·**Precisió**: mesura la proporció de logos detectats correctament del total de logos detectats pel model:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

·***Recall*** o sensibilitat: mesura l'habilitat del model en detectar la presència de logos.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

·**AP:** representa la mitjana de la precisió a mesura que augmenta el *recall*.

·**mAP50:** mesura la mitjana dels valors de AP pel valor límit de IoU a 0.5.

·**mAP50-95:** mesura la mitjana dels valors de AP pels diversos valors límit de IoU, des de 0.5 a 0.95.

Apèndix 6: Prototip d'aplicació

En el següent enllaç, hi ha un vídeo on es mostra el prototip de l'aplicació i simula l'experiència de l'usuari:

https://drive.google.com/file/d/1NF4IiK9C2vUs0ReuT6Rp1LC1YjbEIq-9/view?usp=sharing